Architectures for Language Processing that Leverage Observable Structure in Language

Vicky Zayats

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Mari Ostendorf, Chair

Hannaneh Hajishirzi

Kristina Toutanova

Noah Smith

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Architectures for Language Processing that Leverage Observable Structure in Language

Vicky Zayats

Chair of the Supervisory Committee:
Prof. Mari Ostendorf
Department of Electrical & Computer Engineering

Structural components are an inherent part of both written and spoken language. While a lot of research has been put into studying the latent structure of language, such as the organization of phonemes into words using morphology, and words into sentences using syntax, higher-level explicitly marked observable structure has not been extensively used. Some examples of observable structure in written text include rows and columns of tables, web-page structure outlined through HTML tags, or conversational history structure in a multi-party discussion. This thesis aims to address the problem of leveraging observable structures that naturally co-occur with language. Specifically, we explore two types of approaches, multimodal integration and architecture adaptations, to learn a structure-aware representation of language for three different scenarios, where each has a unique structure very different from the others. In our first scenario, we concentrate on Reddit discussion forums that contain conversation structure as part of the metadata, with the task of predicting the most influential comments in a thread. In our second scenario, we look at spoken language where we use acoustic cues as an observable structure for a disfluency detection task. Finally, in our third scenario, we work with Wikipedia articles that contain both tables and text in a question answering task. We compare explicit modeling of the structure with multimodal methods that use features extracted from the structure. In the popularity prediction task, we use a graph of the tree structure of conversation history in a multi-party discussion as a

way to propagate information about each one of the elements to the rest of the graph. This is done by introducing a novel graph-LSTM architecture that summarizes and propagates information about the discussion happening at different branches of the tree. In a disfluency detection task we are dealing with structure observed through acoustic-prosodic cues, which contain information associated with the specific word sequence in addition to signaling disfluencies. To address this problem we propose a novel approach that reduces the irrelevant aspects of a prosody observation in order to incorporate only relevant information about the structure, making the signal complementary to that found in the textual component. In the question answering task, we are looking at the ways to represent a table and enhance the limited amount of unstructured text associated with some of the table entries. This is done by two novel contributions. First, we generalize the BERT [22] transformer architecture to capture table representations, while pretraining new relations using table corpus extracted from Wikipedia. Secondly, we introduce a novel approach for updating table representations based on the text of an article surrounding the table in order to enrich table entries with broader context. Our findings from all of the tasks suggest that having architectural adaptations that explicitly model observable structure can be more powerful than feature-based methods.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am incredibly grateful to my adviser Mari Ostendorf who was a true inspiration throughout my PhD, supporting me in research and personal matter, helping to keep a bigger picture in mind, providing knowledgeable feedback, believing in me and my ideas, caring and filling our collaboration with indescribable energy and passion for research. Whether in work or in personal life, I knew I could always rely on Mari's kind words and advice, and for that I remain eternally grateful and indebted. I could not have imagined having a better advisor and mentor for my Ph.D.

I have also been lucky to be work with Kristina Toutanova who motivated me to never give up, broadened my horizons, brainstormed new ideas and taught me the power of technical details. I want to thank Hanna Hajishirzi for her guidance during my first years in Ph.D., her endless motivation and encouragement, and being always ready to help and brainstorm new ideas. I am thankful to Richard Wright for his advice, knowledge and insightful conversations. I would also like to thank the rest of my thesis committee, Noah Smith and Cecilia Aragon for their insightful comments and advice on research.

I am thankful to Joo-Kyung Kim, Young-Bum Kim, Jaimee Teevan, Shamsi Inqbal, Dan Liebling, and Mark Yarvis for their invaluable guidance during my internships at Amazon, Microsoft, and Intel.

I would like to thank all my co-authors, collaborators, lab mates, and friends: Farah Nadeem, Trang Tran, Kevin Lybarger, Sara Ng, Roy Lu, Ellen Wu, Kevin Everson, Michael Lee, Sitong Zhou, Yi Luan, Aaron Jaech, Hao Fang, Hao Cheng, Yuzhong Liu, Ji He, Sining Sun, Jingyong Hou, Shobhit Hathi, Alex Marin, Julie Medero, Nicole Nichols, Courtney Mansfield, and Esther Le Grezause for their company, support, and insightful discussions

throughout my years at school.

I am thankful to my undergraduate advisers Karen Livescu, Israel Cohen, and Nir Tessler, who have first introduced me to research and launched me on this wonderful path.

I am thankful for DARPA, NSF, and Google for funding, without which this thesis would not exist.

Last and foremost, I am truly thankful to my family. I am grateful to my parents who always supported and encouraged me throughout the way, to my husband Artiom for traveling half of the world with me for my passion, being there everyday and held my hand through the whole journey of Ph.D, and to my son Ben for helping to put things into perspective.

DEDICATION

To my dad who always was dreaming to become a researcher and contribute to science.

Chapter 1

INTRODUCTION

1.1 Language and Structure

Structural components are an inherent part of both written and spoken language. One example of such structure is the organization of phonemes using morphology into words, and words into sentences using syntax and grammar. This type of structure can be characterized as a latent, or *unobservable structure*, since it does not have an explicit marking in the text or speech. In the last few decades there has been a lot of research on studying and modeling the latent language structure, with most of the efforts made towards sentence-level representations. On the other hand, explicitly marked, higher-level *observable structure* that is often present has not been extensively used. Some examples of observable structure in written text include paragraph and section structure in documents, conversational history in a multi-party discussion, rows and columns of tables, or web-page structure outlined through HTML tags. Additional information in speech, which includes emphasis, focus, contrast, and phrase structure plays an important role and affects the meaning of the spoken utterance. This sort of structure is communicated through acoustic-prosodic cues, which represent another type of observable information that can be used in spoken language processing. Because a lot of attention has already been put into studying and modeling latent linguistic structure of language, this dissertation aims to address the problem of leveraging observable structures that naturally co-occur with language.

Different genres of language (written articles, social media discussions, lectures, spontaneous conversations, etc.) vary greatly in terms of form, style and observable structure. Form differences include the origin of the signal, such as audio, web-based text, images, or videos. One of the challenges associated with working with forms other than text is word

errors that are introduced while extracting words from audio or images. These errors introduce noise and present additional challenges for language processing. This thesis will ignore this issue by using hand transcripts when working with spoken language. Style differences include disfluencies, hesitations, latent sentence boundaries, and overlapping turn taking in speech comparing to written text, or large amount of out-of-vocabulary words, grammaticality, and domain-specific jargon in social media comparing to more formal genres like news. This thesis will use style-matched training data to avoid challenges associated with genre mismatch. Finally, differences in observable structure include the amount of the structure (sparse structure of article defined by paragraphs and sections vs. dense structure of tables) and noisiness of the structural components (well-defined structure of multi-party thread discussion vs. context-sensitive acoustic cues extracted from speech). Incorporating observable structure across different genres that vary in terms of structure properties will be the focus of this dissertation.

The main goal of this thesis is to look at the problem of leveraging observable structure for language processing across problems that have different properties associated with observable structure. Since different scenarios require different strategies, here we concentrate on three tasks, each of which has a unique set of properties. In a popularity prediction task we use a well-defined structure as a way to propagate information about each one of the elements to the rest of the graph. In a disfluency detection task we are dealing with structure observed through acoustic cues, which in addition to disfluency-related signal contains other sources of information, requiring mechanisms to filter out irrelevant parts of signal. In the question answering task, we are looking at the ways to represent a table and enhance the limited amount of unstructured text associated with some of the table entries.

1.2 Overview of the Solutions

There are two general ways that structure can be incorporated with popular neural language architectures: a) extracting features of the structure and treating these as a separate "modality" in multimodal integration, and b) modifying the neural net architecture to re-

flect the structure of the problem. In the first approach, research concentrates on ways to effectively represent and combine the two modalities (textual and structural). In the second approach, the common neural architectures used for language processing can be adapted to incorporate a structural component, similar to how graph convolutional neural networks [40] adapted traditional convolution neural network to work on graphs, or how hierarchical recurrent neural networks can be adapted to summarize different layers of hierarchy in a document [112].

In this thesis we explore the above mentioned two types of approaches - multimodal integration and architecture adaptations, and how they can be used to learn a structure-aware representation of language. Specifically, we are focusing on three types of structures: thread discussion structure in social media, speech prosodic structure for spoken language, and table structure for documents that contain both tables and unstructured text. For all of these problems we compare explicit modeling of the structure with multimodal methods that use structure as a set of features. Our experiments show that having architectural adaptations that explicitly model observable structure can be more powerful than feature-based methods.

1.3 Tasks

This dissertation focuses on three tasks, each one associated with a different type of a structured context: discussion structure in the task of predicting popularity of comments on social media, phrase structure observable through prosody in a disfluency detection task, and table structure in a question answering task. In this section we will give a general overview of each of those problems; further details are provided in the relevant chapters.

Popularity Prediction. Social media provides a convenient and widely used platform for discussions among users. Popularity prediction of comments in social media, which can also be referred to as community endorsement, is a task of growing interest with applications in advertisement and social dynamics modeling. In this work, our task was in identifying influential comments on Reddit, measured in number of upvotes and downvotes, using the

text of the comment given the structure of the discussion. When the comment-response links are preserved, those discussions can be represented in a tree structure where comments represent nodes, the root is the original post, and each new reply to a previous comment is added as a child of that comment. When learning a discussion representation on social media such as Reddit, it is important to capture the overall context of the discussion. The main focus of this line of work is to incorporate a structure of discussion on Reddit in order to improve the prediction of popularity of an individual comment.

Disfluency detection. Disfluencies are hesitations, repetitions, and self-corrections in spontaneous speech that break the usual flow of the sentence. They are important to account for, both because of the challenge that the disrupted grammatical flow poses for natural language processing of spoken transcripts and because of the information that they provide about the speaker. Automatic disfluency detection involves identifying the unwanted sequence of words that precede the point where the flow breaks. To identify disfluencies, we need to understand a structural component of a sentence, in particular where the flow breaks. Prosody provides a view of sentence structure, parallel to the text, so it can contribute information about the structure of a phrase. Most algorithms for disfluency detection use only word transcripts due to many sources of variability affecting the acoustic correlates. The goal of this work is to improve disfluecy detection by accounting for a structural component from two different angles. First, we want to account for the pattern-based structure that originates from a frequent similarity between reparandum and repair of the disfluency. Secondly, we want to incorporate an acoustic view that can serve as a noisy observation of phrase structure.

Question Answering on Tables. Tables in web documents are pervasive and can be directly used to answer many queries searched on web, leading to question answering on tables being an important recent research direction. In order to effectively answer questions involving tables, it is important to have a good table representation. A table is an example of text represented in a structured way. Text in tables have special relations, such as relations to a row or column header, or nearby table cells. On one hand, using some standard language representation architectures, such as BERT [22], gives an advantage of using pretrained

representations for unstructured component of the table contents. On the other, in order to use those representations efficiently we need to adapt them to account for the table structure. In addition, very often information presented in tables is compact and limited. Finally, tables are often appear within a context, such as as an article describing the table. Thus, including the information from an article as an additional context can enrich table representation. In this work we try to improve question answering on tables by improving table representation and enriching the table entries with additional information from the surrounding text.

1.4 Contributions

At high level, the main contribution of this thesis is to highlight the utility of leveraging observable structure in natural language processing explicitly in the model vs. implicitly as a contextual feature. In addition, the main application contributions of this dissertation are as follows:

Popularity Prediction. This work introduces a novel approach for representing social media discussions with a graph-LSTM, where each comment associated with a single LSTM node. Our approach provides a method for summarizing and propagating information about the discussion happening at different branches of the thread, thus improving popularity prediction for individual comments. Our analysis suggests that having such information propagation across the discussion thread more clearly highlights the role of language context for controversial comments and jokes.

Disfluency detection. Much of information that is present in an acoustic signal is irrelevant to the disfluency detection task. Due to high variability in acoustic correlates, using simple multimodal fusion approaches do not usually perform well. In order to address this problem we propose a novel approach that effectively cleans a prosody observation incorporating only relevant information about the structure, making the signal complementary to the one found in the textual component. This method requires architecture modifications to extract a better prosody representation and in this line of work we compare it with more conventional multimodal fusion approaches. In addition, we present a novel method called

pattern match networks that exploit a parallel unobserved structure between the reparandum and repair often presented in disfluencies. This method is presented and used for disfluency detection task, and serves as a strong baseline for the text-based disfluency detection framework, as well as the foundation that the prosody-based architecture builds upon.

Question Answering on Tables. The main contribution of this work is two-fold. First, we propose a novel BERT-based table representation architecture which makes the use of a pretained BERT model. Secondly, we propose a new approach that propagates information from the article surrounding the table to the relevant parts of the table, further updating the table representation. Those improvements lead to improvements in question answering on tables task using the Natural Questions dataset.

1.5 Outline

The rest of the dissertation is organized as follows.

In Chapter 2 we provide a general overview of related recent work on neural models of language. Then, we give survey of ways that structure has be characterized for language processing tasks, including hierarchical neural models, graph-structured neural architectures, and multimodal approaches. Task-specific related literature will be reviewed in the corresponding chapter.

Chapter 3 focuses on discussion thread structure, where the task is to predict popularity of the comments on Reddit. Here we introduce a graph-LSTM architecture to model both hierarchical and time components of conversation history structure. Experiments on multiple subreddits compare characterization of structure using both feature-based techniques and architectural adaptations, showing improvements of the proposed model compared to a node-independent architecture with feature-based representation of structure. Analyses show a benefit to the model over the full course of the discussion, improving detection in both early and late stages.

In Chapter 4 we present the disfluency detection task and two novel methods associated with using structural components in spoken language. First, we present a novel contribution

of pattern-match networks that leverage latent parallel structure between the reprarndum and repair in disfluencies, which is used as a main building block and to provide a strong text-based disfluency detection system. Next, we introduce a novel multimodal representation learning approach for combining speech prosodic structure with the text-based approach in scenario where structural components consist of a high variance signal.

Chapter 5 introduces two types of architectural changes aimed at improving question answering on tables. First, we introduce a BERT modification specifically targeted for table encoding. Secondly, we introduce a novel mechanism to enrich text representation of cells by linking it to a relevant external information in the article. We evaluate both of the approaches on the Natural Questions dataset, showing improvements of both proposed methods on the subset of Natural Questions dataset that contains answers in tables.

Finally, in Chapter 6 we conclude the thesis by summarizing the main contributions of the dissertation, including the methods developed and experimental findings. Further, we discuss the future directions of research and broader impact of the methods developed in this thesis in terms of potential applications.

Chapter 2
BACKGROUND

This chapter provides a general overview of related recent work on neural models used in natural language processing and gives a survey of ways that structure has been characterized for language processing tasks, including hierarchical neural models, graph-structured neural architectures and multimodal approaches.

2.1 Neural Models of Language

Neural networks play a central role in modeling for natural language processing tasks. There are multiple reasons for this. First of all, with the abundance of data, neural networks are complex enough to find patterns in language. Secondly, neural networks make it possible to map discrete words into a continuous vector space to obtain word embeddings, which can be efficiently calculated using a large amount of unlabeled data. By initially obtaining a good representation of language through pretraining, models for downstream tasks can incorporate general language related knowledge with the task-specific approach. In this section I will give a brief overview of the main architectures this dissertation builds upon.

2.1.1 Recurrent Neural Networks

Language is often represented with sequential neural models. One of the most popular types of architectures used for NLP tasks is a Recurrent Neural Network (RNN), which is a sequential generalization of feedforward neural network with internal memory and connections between nodes that form a directed graph. The two most most frequently used modifications of RNNs are Long-Short Term Memory (LSTM) [34] and Gated Recurrent Units (GRU) [18]. Both models include gating mechanisms that control the proportions of information to forget

vs. to pass on to the next time step in order to overcome the vanishing gradient problem in training. In case of the LSTM, which is used in this thesis, there are input, output, and forget gates, and cell states in addition to the hidden state. Below is a set of equations describing LSTM unit, with the subscripts $i,\ f,\ g,\ c$, and o for the input gate, temporal forget gate, hierarchical forget gate, cell, and output, respectively, σ is a sigmoid function, and $\circ$ indicates the Hadamard product.

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
\tilde{c}_t &= W_c x_t + U_c h_{t-1} + b_c \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
h_t &= o_t \circ tanh(c_t)
\end{aligned}
$$

For many sequence labeling tasks it is beneficial to have access to both past (left) and future (right) contexts. In order to combine the prediction from both of the contexts, the Bidirection LSTM (BiLSTM) first encodes the sequence in both the forward and backward directions and then at each timestep the forward and backward outputs are concatenated. When LSTMs are used for tagging problems, one of the most popular hybrid approaches is to add a CRF layer [43] on top of the LSTM predictions [57]. This is done in order to jointly decode labels for the whole sentence thus overcoming a problem of inconsistencies between output labels, which can result in label sequences that are not valid paths in the state space [119].

Soon after the initial success of RNN models in NLP, an attention mechanism was introduced as part of an RNN-based encoder-decoder architecture in the context of Neural Machine Translation (NMT) [5]. It was developed to solve two main issues with RNNs - the forgetfulness of RNNs, especially for long sequences, and no explicit word alignments during decoding in NMT. The main idea of an Attention layer is to obtain explicit alignments between each of the input tokens $\mathbf{x}_i$ and a hidden state $\mathbf{h}_j^o$ at time j, which is done

by calculating normalized attention scores α_{ji}, such that $\sum_i \alpha_{ji} = 1$, where α_{ji} measures pairwise association between input and output representation. Then, the whole input sentence can be summarized in one vector using a weighted average of input representations with attention scores as weights. There are multiple extensions of the original attention mechanism, including self-attention [83, 100] where attention scores are calculated between a pair of input representations and hierarchical attention [112, 39] with attention calculated at multiple levels.

2.1.2 Transformer

The transformer [100] is another type of architecture for sequence modeling that relies exclusively on self-attention and does not use any recurrent connections. The main idea behind the transformer is in using multiple self-attention layers, one on top of the other, to encode the input sentence. At a high level, the transformer's multi-head self-attention, which calculates the attention multiple times in parallel rather than only computing it once, can be represented in the following way. Given an input sequence of discrete symbols (e.g. words or word pieces), $(x_1 \dots x_n)$, the transformer first maps them to embedding space, $(z_1 \dots z_n)$ where it encodes both the symbol identity and its positional encoding. Then, a set of queries Q corresponding to each of the inputs $(z_1 \dots z_n)$, and a set of key-value pairs (K, V), also corresponding to the inputs $(z_1 \dots z_n)$ are used in order to compute the multi-head attention, where query Q^i and key K^j are used in order to calculate an association between z_i and z_j. Both query and key have dimension d_k, the attention head h in layer l is computed as

$$Att_{h,l}(Q, K, V) = softmax(\frac{(W^Q_{h,l}Q)^T(W^K_{h,l}K)}{\sqrt{d_k}})W^V_{h,l}V \tag{2.1}$$

where $W^Q_{h,l} \in \mathbb{R}^{d_{model} \times d_k}$, $W^K_{h,l} \in \mathbb{R}^{d_{model} \times d_k}$ and $W^V_{h,l} \in \mathbb{R}^{d_{model} \times d_v}$ are learned parameters. By calculating the dot product between query and key projections, the transformer captures the interaction between each pair of inputs at positions i and j. Then, the heads $h = 1, \dots, H$

are concatenated and projected to get a representation that can be used in the next layer:

$$MultiHead_l(Q, K, V) = Concat(Att_{1,l}(Q, K, V), \ldots, Att_{H,l}(Q, K, V))W_l^O \tag{2.2}$$

with the parameter $W_l^O \in \mathbb{R}^{Hd_v \times d_{model}}$.

2.1.3 Contextualized Word Embeddings

In the last few years contextualized word embeddings showed a lot of success. First introduced as ELMo by Peters *et al.* [72], the idea of calculating word embeddings that account for the whole context of a word during inference, rather than using a table look-up, achieved a significant gain on an array of tasks. In the ELMo architecture, stacked bidirectional LSTMs are used together with the language modeling objective, and were pretrained on a large 1B word corpus. Later, the LSTM architecture was replaced by a transformer in GPT [76]. After that, the transformer-based BERT model [22] model was introduced which incorporated information from both preceding and following contexts due to a novel masked LM objective. Since BERT many transformer-based variations of contextualized word embeddings have been introduced [111, 52, 45]. In this thesis, in our experiments on question answering use RoBERTa [52], which has an architecture almost identical to BERT but was pretrained on larger amounts of data and achieves better results in a variety of tasks.

2.2 Characterizing Structure in Language

Structure can be represented in a lot of different ways. This section gives a survey on some methods that can be used for representing structure in language processing to provide context for the work in this thesis.

2.2.1 Hierarchical Models

Hierarchical models are usually used to capture multiple levels of abstraction in data with natural hierarchical structure. For example, lower layers of a neural network can be used to summarize words into a sentence vector, and upper layers to summarize sentences into

a document vector, thus becoming a popular way to model documents [112, 63, 49]. Hierarchical models are also commonly used in dialog modeling [80, 97, 15], where words form sentences, and sentences form dialog acts. There are multiple common ways to represent a structure using neural networks. Earlier approaches concentrated on stacking multiple RNNs to represent different levels of abstraction [49, 80]. Later on, the attention mechanism was used in order to summarize the lower layers of hierarchical model [112, 58]. More recently, transformer-based hierarchical models were introduced to extend the BERT architecture [126].

2.2.2 Graph-Structured Neural Networks

Graph-based neural networks are widely used for a range of problems with a structural representation, ranging from modeling social networks [29, 124, 14] to modeling protein interaction [67, 96] to knowledge graphs [13] and many other research areas. Recent surveys on graph-based neural networks [129, 108] give a good overview and comparison of different graphical neural network architectures. In this section I will give a brief general overview of the methods used for modeling graphs, with the emphasis on the architectures that have the most relevance to the applications studied in this dissertation.

The transformer can also be considered as a graph-based architecture with attention being used to propagate information between nodes.

Graph Convolutional Neural Networks

Convolutional-based approaches are the most common types of graph-based neural networks that are motivated by pattern extracting ability of Convolutional Neural Networks (CNNs), generalizing the operation of convolution from grid data to graph data. Convolution-based graph approaches can be divided into two main streams: spectral-based approaches [11, 32, 40, 48] and spatial-based approaches [3, 101, 124, 14], with the latter being most commonly used due to their efficiency, flexibility, and generality. The main idea behind those

approaches is to define graph convolutions by information propagation, where they formulate graph convolutions as aggregating feature information from the node and its neighbours. The method uses approximations to directly compute convolution by using neighbor nodes' information and iterative updates, thus allowing such methods to be used efficiently on large graphs with billions of nodes and edges (e.g. social network graphs). Some of the later more efficient graph-based models [40, 101], have been widely used in NLP for information extraction [75, 66], knowledge base completion [79], machine translation [8], and multi-hop reading comprehension [109].

Graph Recurrent Neural Networks

Another type of graph-based neural networks is based on the information propagation idea used in RNN, with one popular branch of methods that modify LSTM/GRU architectures to represent tree-like structures. Early work on tree LSTMs have been proposed for a variety of sentence-level NLP tasks including sentiment analysis [93, 46], relation extraction [61, 70], parsing [23], and sentence completion [125]. Those works used various ways to integrate information coming from multiple children. Some options included adding a separate forget gate for each child [93, 70], recurrent propagation among siblings [125], or use of stack LSTMs [23].

Our work on conversation modeling for Reddit (Chapter 3) uses an N-ary Tree-LSTM architecture similar to [93], but applied it to a graph instead of a tree. Our approach mainly differs from the previous works in two respects: the tree structure in our model characterizes a discussion rather than a single sentence; and our architecture incorporates both hierarchical and temporal recursions in one LSTM unit.

After the publication of our work, new graph recurrent neural architectures have been introduced. Zhang *et al.* [127] proposed a variant of gating for a graph-LSTM for sentence representation. Another modification of [127] was also used for paragraph representation with the modification of using partially-connected graphs [109].

2.2.3 Multimodal Approaches

Another way of combining structural and language components is using multimodal approaches, where both structural and language components can be treated as two separate sets of features. In this section we will give a broad overview of some methods used in multimodal representation learning and fusion techniques.

Multimodal Representation Learning

Multimodal problems vary a lot with respect to the amount of information shared across two (or more) modalities, and how much shared vs. complementary information from one or two modalities is contributing to the task. Thus, multimodal methods would highly depend on the initial assumptions about the data and underlying tasks, and could be classified into three categories: 1) approaches that do not have any constraints with respect to the overlap between information carried by different modalities; 2) approaches that are explicitly interested in representing only shared information between two modalities; and 3) approaches that aim to represent complementary information between the modalities. In our work on prosody-aware disfluency detection (Chapter 4) we introduce a novel multimodal representation learning technique that falls under the third category, where we explicitly extract complementary information from the prosodic cues.

The biggest branch of multimodal representation learning focuses on explicit common representation between multiple modalities. While this type of method suits tasks where one modality can be absent at training or inference time (*e.g.*, image captioning), it is not well suited for other tasks where one modality carries important information with respect to the task that is not present in the other modality, thus leading to a loss of information when using a shared representation exclusively. Ngiam *et al.* [65] first introduced deep autoencoders for the multi-modal setting where the architecture can be seen as separate autoencoders for each modality where the autoencoders share one hidden layer to learn a joint representation. Following a similar intuition, Srivastava *et al.* [90] used Deep Restricted

Boltzmann Machines (DBM) in order to create a shared representation between text and image modalities. Methods based on Canonical Correlation Analysis (CCA) [30] explicitly constrain correlations across modalities with the objective function. Given two aligned modalities (views), CCA finds pairs of linear projections of the views which are maximally correlated. The CCA family of algorithms include linear and Kernel CCA [30], and Deep CCA [2]. Another modification of the objective function was proposed by Sohn *et al.* [89], where the new objective function maximizes conditional log-likelihood of each modality given the other, thus emphasizing optimizing for cross-modality correlations. Methods based on Variational Autoencoders [88, 92] were also introduced for joint modeling of images and text.

Some of the more recent work on multimodal learning is concerned with finding complementary information between the modalities, rather than just concentrating on the shared one. One example of this is an extension of CCA [105] that separates information into three channels - one shared and two modality dependent complementary channels, which aims towards explicit representation of both shared and complementary information. Our work on innovation features (Chapter 4) also separates information from prosodic view into shared and complementary information found in transcript, while keeping the transcript modality (the most informative one for disfluency detection task) unchanged.

Fusion

Modalities can be combined in various ways. The most straightforward approach would be early fusion, e.g. concatenating features from two or more modalities. Another option can be combining predictions made by separate modality models, which is often referred to as late fusion. Most of the other methods concentrate on developing models that fuse different modality representations at an intermediate (internal) stage of the model. Having an additional input often adds to a complexity of the model. Since for different tasks input variables can be in a different form, supervised algorithms are often more application-specific. Xu *et al.* [110] used an attention mechanism in order to learn the correspondence between a part of the image and the corresponding text. Similarly, an attention mechanism was used

to learn the modality interaction at each time step for videos [115]. Ma *et al.* [56] used an additional multimodal layer on top of an RNN before the output which would combine the language representation from the RNN and the current word embedding with the image embedding. Recent work on sentiment analysis from videos proposed to use an outer product between all modality representations [114]. The main motivation for this approach is the observation that often information across multiple modalities is complementary, thus the approach makes it possible to capture second order interactions.

Chapter 3

DISCUSSION THREAD STRUCTURE

Discussion thread structure is an observable structure that is inherit to discussions on social media and can be easily extracted from the metadata. This chapter concentrates on characterizing such conversational structure and combining it with a neural representation of language, evaluated on a popularity prediction task on Reddit. Work described in this chapter was published in Transactions of the Association for Computational Linguistics [120].

3.1 Comment Popularity Prediction

Social media provides a convenient and widely used platform for discussions among users. When the comment-response links are preserved, those conversations can be represented in a tree structure where comments represent nodes, the root is the original post, and each new reply to a previous comment is added as a child of that comment. Some examples of popular services with tree-like structures include Facebook, Reddit, Quora, and StackExchange. Figure 3.1 shows an example conversation on Reddit, where bigger nodes indicate higher upvoting of a comment.[1] In services like Twitter, tweets and their retweets can also be viewed as forming a tree structure. When time stamps are available with a contribution, the nodes of the tree can be ordered and annotated with that information. The tree structure is useful for seeing how a discussion unfolds into different subtopics and showing differences in the level of activity in different branches of the discussion.

Predicting popularity of comments in social media is a task of growing interest. Popularity has been defined in terms of the volume of the response, but when the social media platform has a mechanism for readers to like or dislike comments (or, upvote/downvote), then

[1]The tool `https://whichlight.github.io/reddit-network-vis` was used to obtain this visualization.

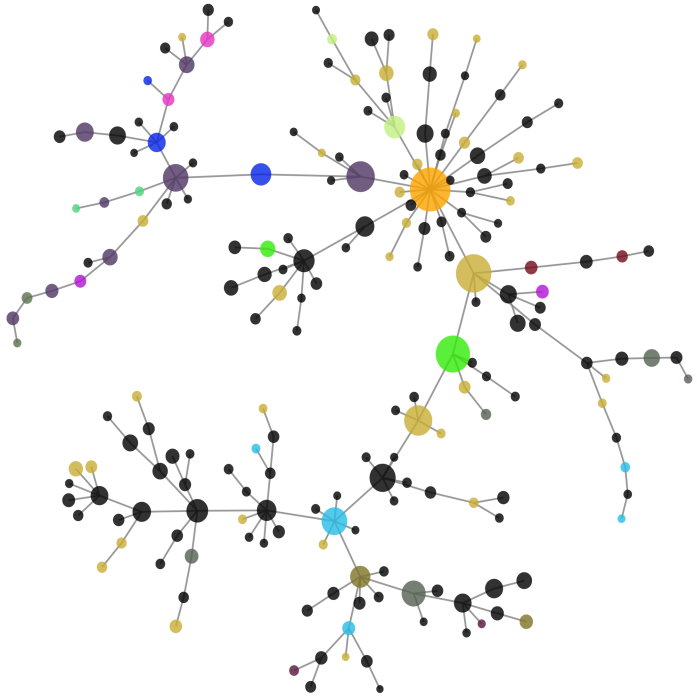
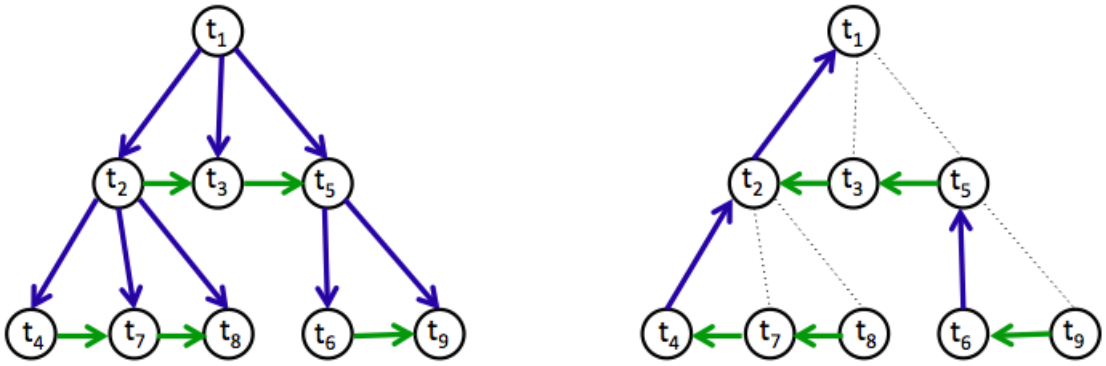


Figure 3.1: Visualization of a sample thread on Reddit. Vertices are individual comments, while edges indicate a reply structure in the thread.

the difference in positive/negative votes provides a more informative score for popularity prediction. This definition of popularity, which has also been called community endorsement [24], is the task of interest in our work on tree-structured modeling of discussions.

Previous studies found that the time when the comment/post was published has a big impact on its popularity [44]. In addition, the number of immediate responses can be predictive of the popularity, but some comments with a high number of replies can be either controversial or have a highly negative score. Language should be extremely important for distinguishing these cases. Indeed, community style matching is shown to be correlated to comment popularity in Reddit [98]. However, learning useful language cues can be difficult due to the low frequency of these events and the dominance of time, topic and other factors. Thus, in several prior studies, authors constrained the problem to reduce the effect of those factors [44, 94, 38]. In this study, we have no such constraints, but attempt to use the tree



(a) Forward hierarchical and timing structure

(b) Backward hierarchical and timing structure

Figure 3.2: An example of model propagation in a graph-structured LSTM. Here, the node name are shown in a chronological order, e.g. comment t_1 was made earlier than t_2. 3.2(a) Propagation of graph-structured LSTM in the forward direction. Blue arrows represent hierarchical propagation, green arrows represent timing propagation. 3.2(b) Backward hierarchical (blue) and timing (green) propagation of graph-LSTM. Dash lines indicate hirachical structure that is not directly used in the propagation.

structure to capture the flow of information in order to better model the context in which a comment is submitted, including both the history it responds to as well as the subsequent response to that comment.

To capture discussion dynamics, we introduce a novel approach to modeling the discussion using a bidirectional graph-structured LSTM, where each comment in the tree corresponds to a single LSTM unit. In one direction, we capture the prior history of contributions leading up to a node, and in the other, we characterize the response to that comment. Motivated by prior findings that both response structure and timing are important in predicting popularity [24], the LSTM units include both hierachical and temporal components to the update, which distinguishes this work from prior tree-structured LSTM models. We assess the utility of the model in experiments on popularity prediction with Reddit discussions, comparing to a neural network baseline that treats comments independently but leverages information about the graph context and timing of the comment. We analyze the results to show that

the graph LSTM provides a useful summary representation of the language context of the comment.

As in [24], but unlike other work [31], our model makes use of the full discussion thread in predicting popularity. While knowledge of the full discussion is only useful for post-hoc analysis of past discussions, it is reasonable to consider initial responses to a comment, particularly given that many responses occur within minutes of someone posting a comment. Comments are often popular because of witty analogies made, which requires knowledge of the world beyond what is captured in current models. Responses to these comments, as well as to controversial comments, can improve popularity prediction. Responses of others clearly influence the likelihood of someone to like or dislike a comment, but also whether they even read a comment. By introducing a forward-backward tree-structured model, we provide a mechanism for leveraging early responses in predicting popularity, as well as a framework for better understanding the relative importance of these responses.

The main contributions of this chapter include: a novel approach for representing tree-structured language processes (e.g., social media discussions) with LSTMs; evaluation of the model on the popularity prediction task using Reddit discussions; and analysis of the performance gains, particularly with respect to the role of language context.

This work has been published in the Transactions of the Association for Computational Linguistics [120] and presented at EMNLP, 2017.

3.2 Related Work

The problem of predicting popularity in social media platforms has been the subject of several studies. Popularity as defined in terms of volume of response has been explored for shares on Facebook [17] and Twitter [6] and Twitter retweets [94, 128, 10]. Studies on Reddit predict karma as popularity [44, 38, 31] or as community endorsement [24]. Popularity prediction is a difficult task where many factors can play a role, which is why most prior studies control for specific factors, including topic [94, 106], timing [94, 38], and/or comment content [44]. Controlling for specific factors is useful in understanding the components of a successful post,

but it does not reflect a realistic scenario. Studies that do not include such constraints have looked at Twitter retweets [10] and Reddit karma [31, 24].

The work in [31] uses reinforcement learning to identify popular threads to track given the past comment history, so it is learning language cues relevant to high karma but it does not explicitly predict karma. In addition, it models relevance via an inner-product of past and new comment embeddings, and uses an LSTM to model inter-comment dependencies among a collection of comments irrespective of their sibling-parent relationship, whereas the LSTM in our work is over a graph that accounts for this relationship.

The work most closely related to our study is by Fang *et al.* [24]. The node-independent baseline implemented in our study is equivalent to their feedforward network baseline, but the results are not directly comparable because of differences in training (we use more data) and input features. The most important difference in our approach is the representation of textual context using a bidirectional graph-LSTM, including the history behind and responses to a comment. Other differences are: i) Fang *et al.* use an LSTM to characterize comments, while our model uses a simple bag-of-words approach, and ii) they learn latent submission context models to determine the relative importance of textual cues, while our approach uses a submission context SVM to prune low karma comments (ignoring their text). Allowing for differences in baselines, we note that the absolute gain in performance from using text features is larger for our model, which represents language context.

3.3 Methods

The proposed model is a bidirectional graph LSTM that characterizes a full threaded discussion, assuming a tree-structured response network and accounting for the relative order of the comments. Each comment in a conversation corresponds to a node in the tree, where its parent is the comment that it is responding to and its children are the responding comments that it spurs ordered in time. Each node in the tree is represented with a single recurrent neural network (RNN) unit that outputs a vector (embedding) that characterizes the interim state of the discussion, analogous to the vector output of an RNN unit which characterizes

the word history in a sentence. In the forward direction, the state vector can be thought of as a summary of the discussion pursued in a particular branch of the tree, while in the backward direction the state vector summarizes the full response subtree that followed a particular comment. The state vectors for the forward and backward directions are concatenated for the purpose of predicting comment karma. The RNN updates – both forward and backward – incorporate both temporal and hierarchical (tree-structured) dependencies, since commenters typically consider what has already been said in response to a parent comment. Hence, we refer to it as a graph-structured RNN rather than a tree-structured RNN. Figures 3.2(a) and 3.2(b) show an example of the state connections associated with hierarchical and timing structures for the forward and backward RNNs, respectively.

The supervision signal in training will impact the character of the state vector, and the forward and backward state sub-vectors are likely to capture different phenomena. Here, the objective is to predict quantized comment karma. We anticipate that the forward state will capture relevance and informativeness of the comment, and the backward process will capture sentiment and richness of the ensuing discussion.

The specific form of the RNN used in this work is an LSTM. The detailed implementation of the model is described in the sections to follow.

3.3.1 Graph-structured LSTM

Each node in the tree is associated with an LSTM unit. The input x_t is an embedding that can incorporate both comment text and local submission context features associated with thread structure and timing, described further in section 3.3.2. The node state vector h_t is generated using a modification of the standard LSTM equations to include both hierarchical and timing structures for each comment. Specifically, we use two forget gates - one for the previous (or subsequent) hierarchical layer, and one for the previous (or subsequent) timing layer.

In order to describe the update equations, we introduce notation for the hierarchical and timing structure. In Figure 3.2, the nodes in the tree are numbered in the order that the

comments are contributed in time. To characterize graph structure, let $\pi(t)$ denote the parent of t and $\kappa(t)$ its first child. Time structure is represented only among a set of siblings: $p(t)$ is the sibling predecessor in time, and $s(t)$ is the sibling successor. The pointers $\kappa(t)$, $p(t)$ and $s(t)$ are set to $\emptyset$ when t has no child, predecessor, or successor, respectively. For example, in Figure 3.2(a), the node t_2 will have $\pi(t_2) = t_1$, $\kappa(t_2) = t_4$, $p(t_2) = \emptyset$ and $s(t_2) = t_3$, and the node t_3 will have $\pi(t_3) = t_1$, $\kappa(t_3) = \emptyset$, $p(t_3) = t_2$ and $s(t_3) = t_5$.

Below we provide the update equations for the forward process, using the subscripts i, f, g, c, and o for the input gate, temporal forget gate, hierarchichal forget gate, cell, and output, respectively. The vectors i_t, f_t, and g_t are the weights for new information, remembering old information from siblings, and remembering old information from the parent, respectively. σ is a sigmoid function, and $\circ$ indicates the Hadamard product. If $p(t) = \emptyset$, then $h_{p(t)}$ and $c_{p(t)}$ are set to the initial state value.

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{p(t)} + V_i h_{\pi(t)} + b_i) \\
f_t &= \sigma(W_f x_t + U_f h_{p(t)} + V_f h_{\pi(t)} + b_f) \\
g_t &= \sigma(W_g x_t + U_g h_{p(t)} + V_g h_{\pi(t)} + b_g) \\
\tilde{c}_t &= W_c x_t + U_c h_{p(t)} + V_c h_{\pi(t)} + b_c \\
c_t &= f_t \circ c_{p(t)} + g_t \circ c_{\pi(t)} + i_t \circ \tilde{c}_t \\
o_t &= \sigma(W_o x_t + U_o h_{p(t)} + V_o h_{\pi(t)} + b_o) \\
h_t &= o_t \circ tanh(c_t)
\end{aligned}
$$

When the whole tree structure is known, we can take advantage of the full response subtree to better represent the node state. To that end, we define a backward LSTM that has a similar set of update equations except that only the first child will pass the hidden state to its parent. Specifically, the update equations are the same except that $\pi(t)$ is replaced with $\kappa(t)$, $p(t)$ is replaced with $s(t)$, and a different set of weight matrices and bias vectors are learned.

Let $+$ and $-$ indicate forward and backward embeddings respectively. On top of the LSTM unit, the forward and backward state vectors are concatenated and passed to a softmax

layer to predict 8 quantized karma levels:

$$P(y_t = j|x, h) = \frac{\exp(W_s^j[h_t^+; h_t^-])}{\sum_{k=1}^{8} \exp(W_s^k[h_t^+; h_t^-])}$$

where x and h correspond to the set of input features and state vectors (respectively) for all nodes in the discussion.

3.3.2 Input Features

The full model includes two types of features in the input vector, including non-textual features associated with the submission context and the textual features of the comment at that node.

The **submission context** features are extracted from the graph and metadata associated with the comment, motivated by prior work showing that context factors such as the forum, timing and author of a post are very useful in predicting popularity. The submission context features include:

- *Timing*: time since root, time since parent (in hours), number of later comments, and number of previous comments
- *Author*: a binary indicator as to whether the author is the original poster, and number of comments made by the author in the conversation
- *Graph-location*: depth of the comment (distance from the root), and number of siblings
- *Graph-response*: number of children (direct replies to the comment), height of the subtree rooted from the node, size of that subtree, number of children normalized for each thread (2 normalization techniques), subtree size normalized for each thread (2 normalization techniques).

Two methods are used to normalize the subtree size and number of children to compensate for variation associated with the size of the discussion, specifically: i) subtract the mean

feature value in the thread, and ii) divide by the square root of the rank of the feature value in the thread.

These features are a superset of those used in [24]. The subvector including all these features is denoted x_t^s.

The **comment text** features, denoted x_t^c, are generated using a simple average bag-of-words representation learned during the training:

$$x_t^c = \frac{1}{N} \sum_{i=1}^{N} W_e^i$$

where W_e^i is an embedding of the i-th word in the comment, and N is the number of words in the comment. Comments longer than 100 words were truncated to reduce noise associated with long comments, assuming that the early portion carries the most information. The percentage of the comments that exceed 100 words is around $11\% - 14\%$ for the subreddits used in the study. In all experiments, the word embedding dimension is $d = 100$, and the vocabulary includes only words that occurred at least 10 times in the dataset.

The input vector x_t is set to either x_t^s or $[x_t^s; x_t^c]$, depending on whether the experiment uses text.

3.3.3 Pruning

Often the number of comments in a single subtree can be large, which leads to high training costs. A large percentage of the comments are low karma and minimally relevant for predicting karma of neighbors, and many can be easily identified with simple graph and timing features (e.g. having no replies or contributed late in the discussion). Therefore, we introduce a preprocessing step that identifies comments that are highly likely to be low karma to decrease the computation cost. We then assign these nodes to be level 0 and prune them out of the tree, but retain a count of nodes pruned for use in a count-weighted bias term in the update to capture information about response volume.

For detecting low karma comments, we train a simple SVM classifier to identify comments at the 0 karma level based on the submission context features. If a pruned comment leads

to a disconnected graph (e.g., an internal node is pruned but not its children), then the comment is retained in the tree. In testing, all pruned comments are given a predicted level of 0 and accounted for in the evaluation.

The state updates have an additional bias term for any nodes that have subsequent sibling or children comments pruned. For example, consider Figure 3.2, if nodes $\{t_5, t_6, t_7, t_9\}$ are pruned, then t_8 will have a modified forward update, and t_3, t_4 will have a modified backwards update. At node t, define M_t^κ to be the number of levels pruned below it, M_t^p as the number of immediately preceeding comments pruned in its subgroup (responding to the same parent), and M_t^s as the number of subsequent comments pruned in its subgroup plus the non-initial comments in the associated subtrees. In the example above, $M_3^\kappa = 1,\ M_3^s = 2,\ M_4^s = 1,\ M_8^p = 1$, and all other $M_t^* = 0$. The pointers are updated reflect the structure of the pruned tree, so $p(8) = 4,\ s(4) = 8,\ s(3) = \emptyset$. The bias vectors r_κ, r_p and r_s are associated with the different sets of nodes pruned.

Let $+$ and $-$ indicate forward and backward embeddings respectively. The forward update has an adjusted predecessor contribution $(h^+_{p(t)} + M_t^p r_p)$. The backward update adds $M_t^s r_s + M_t^\kappa r_\kappa$ to either $h^-_{s(t)}$ or $h^-_{\kappa(t)}$, depending on whether it is a time or hierarchical update, respectively.

3.3.4 Training

The objective function is minimum cross-entropy over the quantized levels. All model parameters are jointly trained using the adadelta optimization algorithm [123]. Word embeddings are initialized using word2vec skip-gram embeddings [59] trained on all comments from the corresponding subreddit. The code is implemented in Theano [95] and is available at `https://github.com/vickyzayats/graph-LSTM`.We tune the model over different dimensions of the LSTM unit, and use the performance on the development set as a stopping criteria for the training.

subreddit	**comments**	**threads**	**vocab size**
askwomen	0.8M	3.5K	32K
askmen	1.1M	4.5K	35K
politics	2.2M	4.9K	55K

Table 3.1: Data statistics.

subreddit	Prec	Rec	% pruned
askwomen	67.9	72.4	36.9
askmen	60.1	75.3	36.1
politics	49.6	60.3	47.5

Table 3.2: Precision and recall of the pruning classifier and percentage of comments pruned.

3.4 Experiments

3.4.1 Data

Reddit[2] is a popular discussion forum platform consisting of a large number of subreddits focusing on different topics and interests. In our study, we experimented with 3 subreddits: askwomen, askmen, and politics. All the data consists of discussions made in the period between January 1, 2014 and January 31, 2015. Table 3.1 shows the total amount of data used for each of the subreddits. For each subreddit, the threads were randomly distributed between training, development (dev) and test sets with the proportions of 6:2:2. The performance of the pruning classifier on the dev set is presented in Table 3.2.

[2] `https://reddit.com`

3.4.2 Task and Evaluation Metrics

Reddit karma has a Zipfian distribution, highly skewed toward the low-karma comments. Since the rare high karma comments are of greatest interest in popularity prediction, [24] proposes a task of predicting quantized karma (using a nonlinear head-tail break rule for binning) with evaluation using a macro average of the F1 scores for predicting whether a comment exceeds each different level. Experiments reported here use this framework.

Specifically, all the comments with karma lower than 1 are assigned to level 0, and each subsequent level corresponds to karma less than or equal to the median karma in the rest of the comments based on the training data statistics. Each subreddit has 8 quantized karma levels based on its karma distribution. There are 7 binary subtasks (does the comment have karma at level j or higher for $j = 1, \ldots, 7$), and the scoring metric is the macro average of $F1(j)$. For tuning hyperparameters and as a stopping criterion, we use a linearly weighted average of F1 scores to increase the weight on high karma comments, which gives slightly better performance for the high karma cases but has only a small effect on the macro average.

3.4.3 Baseline and Contrast Systems

We compare the graph LSTM to a node-independent baseline, which is a feedforward neural network model consisting of input, hidden and softmax layers. This model is a simplification of the graph-LSTM model where there is no connection between nodes. The node-independent model characterizes a comment without reference to either the text of the comment that it is responding to or the comments reacting to it. However, the model does have information on the size of the response subtree via the submission context input features. Both node-independent and graph-structured models are trained with the same cost function and tuned over the same set of hidden layer dimensions.

We contrast performance of both architectures with and without using the text of the comment itself. As shown in [24], simply using submission context features (graph, timing, author) gives a strong baseline. In order to evaluate the role of each direction (forward or

backward) in the graph-structured model, we also present results using only the forward direction graph-LSTM for comparison to the bidirectional model. In addition, in order to evaluate the importance of the language of the comment itself vs. the language used in the rest of the tree, we perform an interpolation between the graph-LSTM with no language features and the node-independent model with language features. The relative weight for the two models is tuned on the development set.

3.4.4 Karma Level Prediction

The results for the average F1 scores on the test set are presented in Table 3.3. In experiments for all the subreddits, graph-structured models outperform the corresponding node-independent models both with and without language features. Language features also give a greater performance gain when used in the graph-LSTM models. The fact that the forward graph improves over the interpolated models shows that it is not simply the information in the current node that matters for karma of that node. Finally, while the full model outperforms the forward-only version for all the subreddits, the gain is smaller than that obtained by the forward direction alone over the node-independent model, so the forward direction seems to be more important.

The karma prediction results (F1 score) at the different levels is shown in Figure 3.3. While in askmen and askwomen subreddits the overall performance decreases for higher levels, the politics subreddit has an opposite trend. This may be due in part to the lower pruning recall in the politics subreddit, but [24] also observe higher performance for high karma levels in the politics subreddit.

3.5 Analysis

Here, we present analyses aimed at better understanding the behavior of the graph-structured model and the role of language in prediction. All analyses are performed on the development set. The analyses are motivated by considering possible scenarios that are exceptions to the easy cases, which are: i) comments that are contributed early in the discussion and spawn

Model	Text	askwomen	askmen	politics
indep	no	53.2	48.3	46.6
graph	no	54.6	52.1	47.9
indep	yes	52.8	50.7	47.4
interp	mix	54.7	52.1	48.2
graph(f)	yes	55.0	53.3	49.9
graph	yes	**56.4**	**54.8**	**50.4**

Table 3.3: Average F1 score of karma level prediction for node-independent (indep) vs. graph-structured (graph) models with and without text features; interp corresponds to an interpolation of the graph-structured model without text and the node-independent model with text; and graph(f) corresponds to a graph-structured model contains forward direction only.

large subtrees, likely to have high karma, and ii) comments with small subtrees that typically have low karma. We hypothesized three scenarios where the bidirectional graph-LSTM with text might be useful. One case is controversial comments, which have large subtrees but do not have high karma because of downvotes; these tend to have *overprediction* of karma when using only submission context. The other two scenarios involve *underprediction* of karma when using only submission context. Early comments associated with few children and a more narrow subtree (see the downward chain in Figure 3.1) may spawn popular new threads and benefit from the popularity of other comments in the thread (more readers attracted), thus having higher popularity than the number of children suggests. Lastly, comments that are clever or humorous discussion endpoints might have high popularity but small subtrees. These two cases tend to differ in their relative timing in the discussion.

3.5.1 Karma Prediction vs. Time

The first study looked at where the graph-LSTM provides benefits in terms of timing. We plot the average F1 score as a function of the contribution time in Figure 3.4. As an approximation

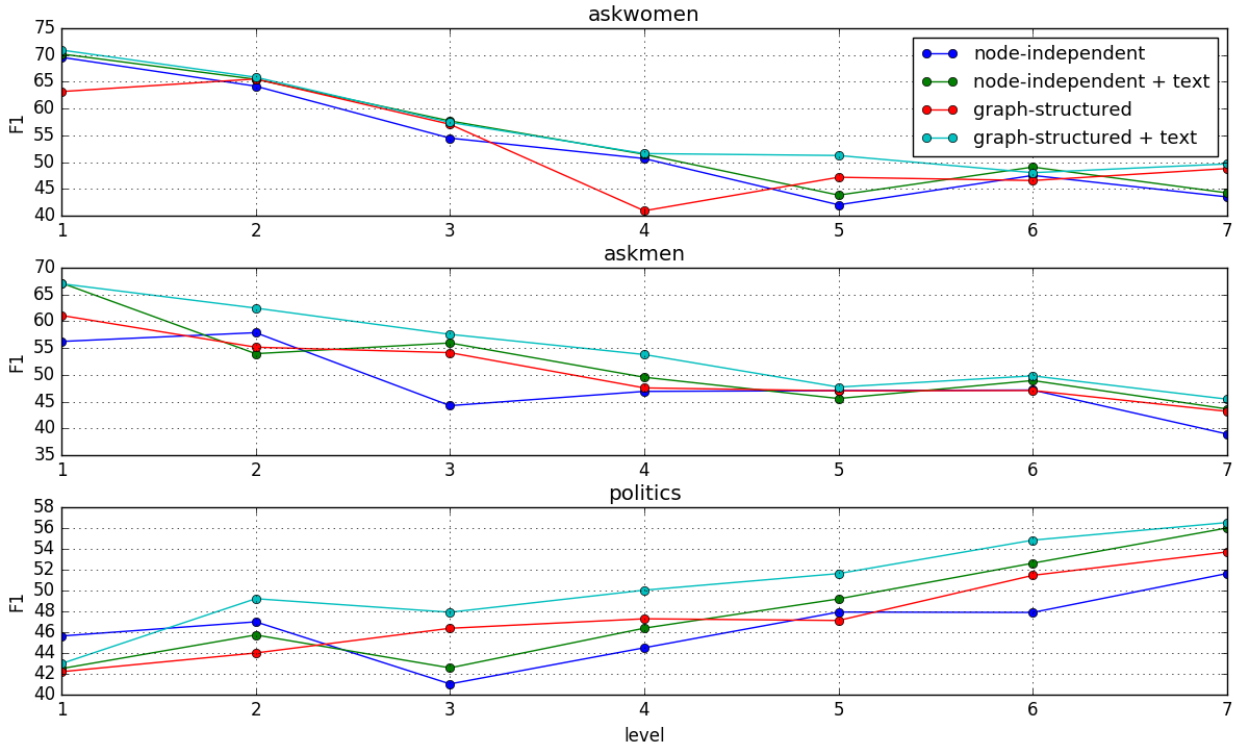


Figure 3.3: F1 scores as a function of the quantized levels for different model configuration.

for time, we use the quantized number of comments made prior to the current comment. The plots show that the graph-structured model improves over the node-independent model throughout the discussion. Relative gains are larger towards the end of discussions where the node-independent performance is lower. A similar trend is observed when plotting average F1 as a function of depth in the discussion tree.

While the use of text in the graph-LSTM seems to help throughout the discussion, we hypothesized that there would be different cases where it might help, and these would occur at different times. Indeed, 93% of the comments that are overpredicted by more than 2 levels by the node-independent model without text (controversial comments) occur in the first 20% of the discussion. Comments that are underpredicted by more than 2 occur throughout the discussion and are roughly uniform (13-19%) over the first half of the discussion, but then quickly ramp down. High-karma comments are rare at the end of the discussion; less than

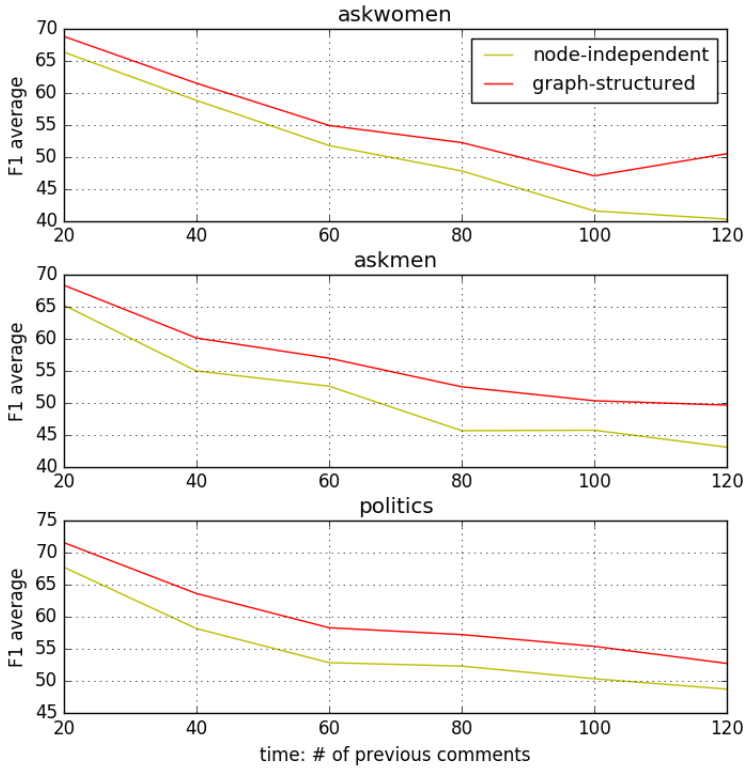


Figure 3.4: Average F1 scores as a function of time, approximated using the number of previous comments quantized in increments of 20.

5% of the underpredicted comments are in the last 30%.

3.5.2 Importance of Responses

In order to see how the model benefits from using the language cues in *underpredicted* and *overpredicted* scenarios, we look at the size of errors made by the graph-LSTM model with and without text features. In Figure 3.5, the x-axis indicates the error between the actual karma level and the karma level predicted by the graph-LSTM using submission context features only. The negative errors represent the *overpredicted* comments, and the positive errors represent the *underpredicted* comments. The y-axis represents the average error between the actual karma level and the karma level predicted by the model using both

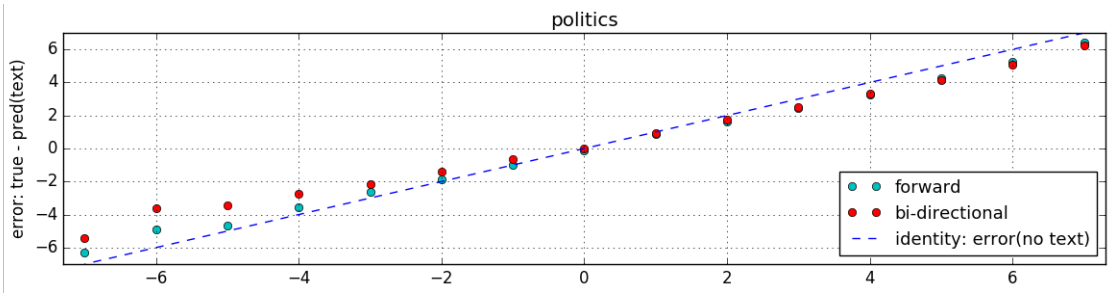


Figure 3.5: The error between the actual karma level and the karma level predicted by the model using both submission context and language features. Negative errors correspond to over-prediction; positive errors correspond to under-prediction.

submission context and language features.The x=y identity line corresponds to no benefit from language features. Results are presented for the politics subreddit; other subreddits have similar trends but smaller differences for the underpredicted cases.

We compare two models – bidirectional and forward direction graph-structured LSTM – in order to understand the role of the language of the replies vs. the comment and its history. We find that, for the bidirectional graph-LSTM model, language is helping identify overpredicted cases more than underpredicted ones. The forward direction model also outperforms the node-independent model, but has less benefit in overpredicted cases, consistent with our intuition that controversy is identifiable based on the responses. Although the comment text input is simply a bag of words, it can capture the mixed sentiment of the responses.

While it is not represented in the plot, larger errors are much less frequent. Looking at average F1 as a function of the number of children (direct responses), we found that the graph-LSTM mainly benefits nodes that have a small number of children, consistent with the two underprediction scenarios hypothesized. However, many underpredicted cases are not impacted, since errors due to pruning contribute to 15-40% of the underpredicted cases, depending on the subreddit (highest for politics). This explains the smaller gains for the positive side of Figure 3.5.

3.5.3 Language Use Analysis

To provide insights into what the model is learning about language, we looked at individual words associated with different categories of comments, as well as examples of the different error cases.

For the word level analysis, we classified words in two different ways, again using the politics subreddit. First, we associate words in comments with zero or positive karma. For each word in the vocabulary, we calculate the probability of a single-word comment being level zero using the trained model with a simplified graph structure (a post and a comment) where all the inputs were set to zero except the comment text. The lists of positive-karma and zero-karma correspond to the 300 words associated with the lowest and highest probability of zero-karma, respectively. We identified 300 positive-karma and zero-karma reply words in a similar fashion, using a simplified graph with individual words us as inputs for the reply while predicting the comment karma.

Second, we identified words that may be indicative of comments that are over- and underpredicted by the graph-structured model without text and for which the graph-LSTM model with text reduced the error by more than 2 levels. Specifically, we choose those words w in comments having the highest ratio $r = p(w|t)/p(w)$, where t indicates an over- or underpredicted comment, subject to minimum occurrence constraints (5 for overpredicted comments, 15 for underpredicted comments). The 50 words with the highest ratio were chosen for each case and any words in both over- and underpredicted sets were eliminated, leaving 47 words. Again, this was repeated for words in replies to over vs. underpredicted comments, but with a minimum count threshold of 20, resulting in 45 words.

The lists are noisy, similar to what is often found with topic model, and colored by the language of the subreddit community, but a few trends can be observed. Looking at the list of words associated with replies to positive-karma comments we noticed words that indicate humor ("LOL", "hilarious"), positive feedback ("Like", "Right"), and emotion indicators ("!!", swearing). Words in comments and replies associated with overpredicted (controversial)

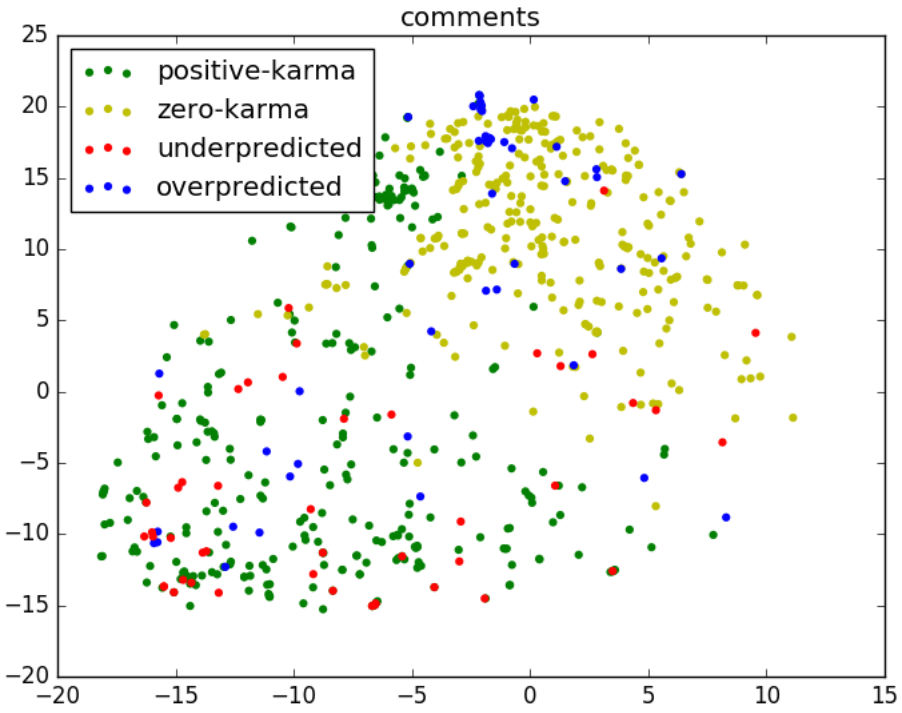


Figure 3.6: The mapping of the words in the comments to the shared space using t-SNE in politics subreddit. Shown are the words that are highly associated with positive-karma, negative-karma, underpredicted and overpredicted comments.

cases are related to controversial topics (sexual, regulate, liberals), named political parties, and mentions of downvoting or indication that the comment has been edited with the word "Edit."

Since the two sets of lists were generated separately, there are words in the over/under-predicted lists that overlap with the zero/non-zero karma lists (12 in the reply lists, 20 in the comment lists). The majority of the overlap (26/32 words) is consistent with the intuition that words on the underpredicted list should be associated with positive-karma, and words on the overpredicted list might overlap with the zero-karma list.

Rather than providing word lists, many neural network studies illustrate trends using word embedding visualization. The embeddings of the words from the union of lists for positive-karma, zero-karma, underpredicted and overpredicted comments and replies were

together used to learn a t-SNE mapping. The results are plotted for comments in Figure 3.6, which shows that the words that are associated with underpredicted comments (red) are aligned with positive-karma words (green) for both comment text and text in replies. Words associated with overpredicted comments (blue) are more scattered, but they are somewhat more like the zero-karma words (yellow). The trends for words in replies are similar.

Table 4.4 lists examples of the different error scenarios with the reference karma and predictions of different models (node-independent without text, feedforward graph-LSTM with text, and the full biLSTM). The first two examples are overpredicted (controversial) cases, where ignoring text leads to a high karma prediction, but the reference is zero. In the first case, the forward model incorrectly predicts high karma because "Republican" tends to be associated with positive karma. The model leveraging reply text correctly predicts the low karma. In the second case, the forward model captures reduces the prediction, but again having the replies is more helpful. The next two cases are examples of underprediction due to small subtrees. Example 3 is incorrectly labeled as level 0 by the forward and no-text models, but because the responses mention "nice joke" and "accurate analogy," the bidirectional model is able to identify it as level 7. Example 4 has only one child, but both models using language correctly predict level 7, probably because the model has learned that references to "Colbert" are popular. The next two examples are underpredicted cases from early in the discussion, many of which expressed an opinion that in some way provided multiple perspectives. Finally, the last two examples represent instances where neither model successfully identifies a high karma comment, which often involve analogies. Unlike the "titanic" analogy, these did not have sufficient cues in the replies.

3.6 Conclusion

In summary, this chapter presents a novel approach for modeling threaded discussions on social media using a graph-structured bidirectional LSTM which represents both hierarchical and temporal conversation structure. The propagation of hidden state information in the graph provides a mechanism for representing contextual language, including the history

Ex	karma				Comment
1	0	7	7	0	Republicans are fundamentally dishonest. (politics, id:1x9pcx)
2	0	7	4	0	That is rape. She was drunk and could not consent. Period. Any of the supposed "evidence" otherwise is nothing but victim blaming. (askwomen, id:2h8pyh)
3	7	0	0	7	The liberals keep saying the titanic is sinking but my side is 500 feet in the air. (politics, id:1upfgl)
4	7	3	7	7	I miss your show, Stephen Colbert. (askmen, id:2qmpzm)
5	7	3	7	7	that is terrifying. they were given the orders to bust down the door without notice to the residents, thereby placing themselves in danger. and ultimately, placing the lives of the residents in danger (who would be acting out of fear and self-defense) (politics, id:1wzwg6)
6	7	0	5	6	It's something, and also would change the way that Police unions and State Prosecutors work. I don't fundamentally agree with the move, since it still necessitates abuse by the State, but it's something. (politics, id:27chxr)
7	6	0	0	0	Chickenhawks always talk a big game as long as someone else is doing the fighting. (politics, id:1wbgpd)
8	6	0	0	0	[They] use statistics in the same way that a drunk uses lampposts: for support, rather than illumination. -Andrew Lang. (politics, id:1yc2fj)

Table 3.4: Example comments and karma level predictions: reference, no text, graph(f), graph.

that a comment is responding to as well as the ensuing discussion it spawns. Experiments on Reddit discussions show that the graph-structured LSTM leads to improved results in predicting comment popularity compared to a node-independent model. Analyses show that the model benefits prediction over the extent of the discussion, and that language cues are particularly important for distinguishing controversial comments from those that are very positively received. Responses from even a small number of comments seem to be useful, so it is likely that the bidirectional model would still be useful with a short-time lookahead for early prediction of popularity.

While we evaluate the model on predicting the popularity of comments in specific forums on Reddit, it can be applied to other social media platforms that maintain a threaded structure or possibly to citation networks. In addition to popularity prediction, we expect the model would be useful for other tasks for which the responses to comments are informative, such as detecting topic or opinion shift, influence or trolls. With the more fine-grained feedback increasingly available on social media platforms (e.g. laughter, love, anger, tears), it may be possible to distinguish different types of popularity as well as levels, e.g. shared sentiment vs. humor.

In this study, the model uses a simple bag-of-words representation of the text in a comment; more sophisticated attention-based models and/or feature engineering may improve performance. In addition, performance of the model on underpredicted comments appears to be limited by the pruning mechanism that we introduced. It would be useful to explore the tradeoffs of reducing the amount of pruning vs. using a more complex classifier for pruning. Finally, it would be useful to evaluate performance using a short window lookahead for responses, rather than the full discussion tree.

After the publication of this work, the follow up work used the graph-LSTM approach for integrating discussion structure to classify discourse acts on Reddit [60].

Chapter 4

SPEECH PROSODIC STRUCTURE

In this chapter we present the disfluency detection task and two novel methods associated with using structural components in spoken language. First, we present a novel contribution of pattern-match networks that leverage latent parallel structure between the reprarndum and repair in disfluencies, which is used as a main building block and to provide a strong text-based disfluency detection system. Next, we introduce a novel multimodal representation learning approach for combining speech prosodic structure with the text-based approach in scenario where structural components consist of ambiguous signal. While the work included in this thesis focuses on neural approaches, it is informed by my earlier work (not included here), including Zayats *et al.* [121], Zayats *et al.* [122], and Zayats *et al.* [119]. This thesis focuses only on the latest work with neural approaches for disfluency detection, which are published on arxiv [117] and in Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies [118].

4.1 Disfluency Detection

In this chapter we present two perspectives on integrating phrase-level structural components of language into standard NLP architectures, where both of the components are done on disfluency detection task. In the first scenario we leverage a parallel structure of many disfluencies to replace hand-crafted features,[1] while in the second scenario we incorporate prosody as a noisy observation of phrase-level structure of an utterance.[2]

---

[1]This work was published on arxiv [117].

[2]This work was published and presented at NAACL, 2019 [118].

4.1.1 Disfluency Structure

Speech disfluencies are hesitations and self corrections in spontaneous speech, including filled pauses, repetitions, repairs and false starts. The rate of disfluencies varies with the speaker and context; one study observed disfluencies once in every 20 words, affecting up to one third of utterances [84]. Disfluencies are important to account for, both because of the challenge that the disrupted grammatical flow poses for natural language processing of spoken transcripts and because of the information that they provide about the speaker.

Most work on disfluency detection builds on the framework that annotates a disfluency in terms of: a reparandum (words which the speaker intends to replace or ignore); followed by an interruption point (+); an optional interregnum ({ }), which are words such as filled pauses, discourse markers, etc.; and then the repair (corrections), if any. Systems are usually evaluated on the ability to correctly identify the reparandum. A few simple examples are given below:

```
[ it's + {uh} it's] almost...
[ was it, + {I mean} , did you ] put...
[by + ] it was attached to...
[he is + our clients are] subject to ...
you want [it + just something] that is ...
```

Based on the similarity/differences between the reparandum and the repair, disfluencies are often categorized into three types: repetition (the first example), rephrase (the second, forth and fifth examples), and restart (the third example).

The interruption point is associated with a disruption in the realization of a prosodic phrase, as will be discussed in Section 4.4.2. The interruption point is not observed in the word transcript.

Pattern Structure in Disfluencies

Previous studies on disfluency detection observe that a repair is often a "rough copy" of a reparandum [12, 130]; thus, hand-crafted pattern match features play an important role in many disfluency detection approaches. They have been shown to be helpful to both sequential and parsing based approaches [104, 119, 25, 107, 74]. In the examples above, "he" resembles "clients", "is" resembles "are" and "it's" is a repetition. However, in many cases, the pattern match is not simple, if present at all, as in the last example.

In Section 4.3 we present a novel architecture that allows automated discovery of the patterns. We first calculate a similarity score between neighboring words in a sentence. Then, we use those scores directly to identify multi-token patterns with convolutional neural networks (CNN). Experiments show that our proposed architecture has an in-domain performance comparable to using hand-crafted pattern match features, and it outperforms baselines in cross-domain settings.

4.1.2 Prosody as an Observable Structure

Prosody is information in speech that affects how a sequence of words is understood. Prosodic elements of speech include pausing, loudness, rhythm, and intonation. Prosody may reflect things like emotional state of the speaker, intent of a speaker, irony or sarcasm. In addition, prosody indicates emphasis, focus and contrast of the utterance, and provides cues about syntactic structure of the utterance that help a listener resolve syntactic ambiguities. Thus, prosody can be used as an indicator of structure of the utterance for spoken language, being especially helpful in parsing disfluent speech. For example, an interruption point in a disfluency is associated with a prosodic disruption and could involve cutting words off or elongation associated with hesitation, followed by a prosodic reset at the start of the repair. There may also be emphasis in the repair to highlight the correction.

The first efforts to integrate prosody with word cues for disfluency detection [7, 87] found gains from using prosody, but word cues played the primary role. In subsequent

work [74, 35, 104], more effective models of word transcripts have been the main source of performance gains. The success of recent neural network systems raises the question of what the role is for prosody in future work.

In terms of prosody modeling, this thesis makes three main contributions. First, our analysis of a high performance disfluency detection algorithm confirms hypotheses about contexts where text-only models have high error rates. Second, we introduce a novel representation of prosodic cues, i.e. the innovation vector resulting from predicting prosodic cues given the whole sentence context. Analyses of the innovation distributions show expected patterns of prosodic cues at interruption points. Finally, we demonstrate improved disfluency detection performance on the Switchboard corpus by integrating prosody and text-based features in a neural network architecture, while comparing early and late fusion approaches.

4.2 Related Work

Most work on disfluency detection falls into three main categories: sequence tagging, noisy-channel and parsing-based approaches. Sequence tagging approaches include conditional random fields (CRF) [27, 68, 121], Max-Margin Markov Networks (M^3N) [74], Semi-Markov CRF [25], and Those approaches were recently replaced by recently recurrent neural networks [36, 119, 103]. The main benefit of sequential models is the ability to capture long-term relationships between reparandum and repairs. Noisy channel models operate on a relationship between the reparandum and repair for identifying disfluencies [12, 130]. Lou *et al.* [54] used a neural language model to rerank sentences using the noisy channel model. Approaches that combine parsing and disfluency removal tasks include [77, 35, 99]. Recently a transition-based neural model architecture was proposed for disfluency detection [104]. The current state-of-the-art in disfluency detection [102] uses a neural machine translation framework with a transformer architecture and additional simulated data. All of the models mentioned above rely heavily on pattern match features, hand-crafted or automatically extracted, that help to identify repetitions and disfluencies with parallel syntactic structure. Our work on pattern-match networks (Section 4.3) that automatically extract pattern-match features is

similar to the approach in Lou *et al.* [53] that was developed concurrently to ours, while we provide more extensive experiments and analysis of the model. After the publication of our work, several approaches have improved the performance, including usage of BERT [4] and self-training a self-attentive constituency parser [55].

While prosodic features are useful for detecting interruption points [64, 86, 85, 51], recent methods on disfluency detection predominantly rely on lexical information exclusively. An exception is [25], which showed some gains using a simple concatenation of pause and word duration features. Similar to disfluency detection, parsing has seen little use of prosody in recent studies. However, Tran *et al.* [99] recently demonstrated that that a neural model using pause, word and rhyme duration, f0 and energy helps in spoken language parsing, specifically in the regions that contain disfluencies. We found that the approach described in [99] did not lead to improved disfluency detection performance, and hence developed a new approach.

4.3 Pattern Structure in Disfluencies

4.3.1 Methods

The main motivation behind our approach is to allow the model to automatically learn and find patterns in sentences without defining them via hand-crafted features. Our proposed model uses two levels to automatically find patterns in sentences. In the first level we calculate similarities for each word in a sentence with words in the surrounding window, which we refer to as neighbor similarity. After calculating the single-token similarity weights, in the second level, we use those weights as features to extract local patterns using a convolutional neural network. The schematic diagram of the model is presented in Figure 4.1.

Neighbor Similarity

The hand-crafted pattern match features used in disfluency detection are usually in the form of "*does the exact word/POS/bigram appeared previously in a fixed length window?*" In our

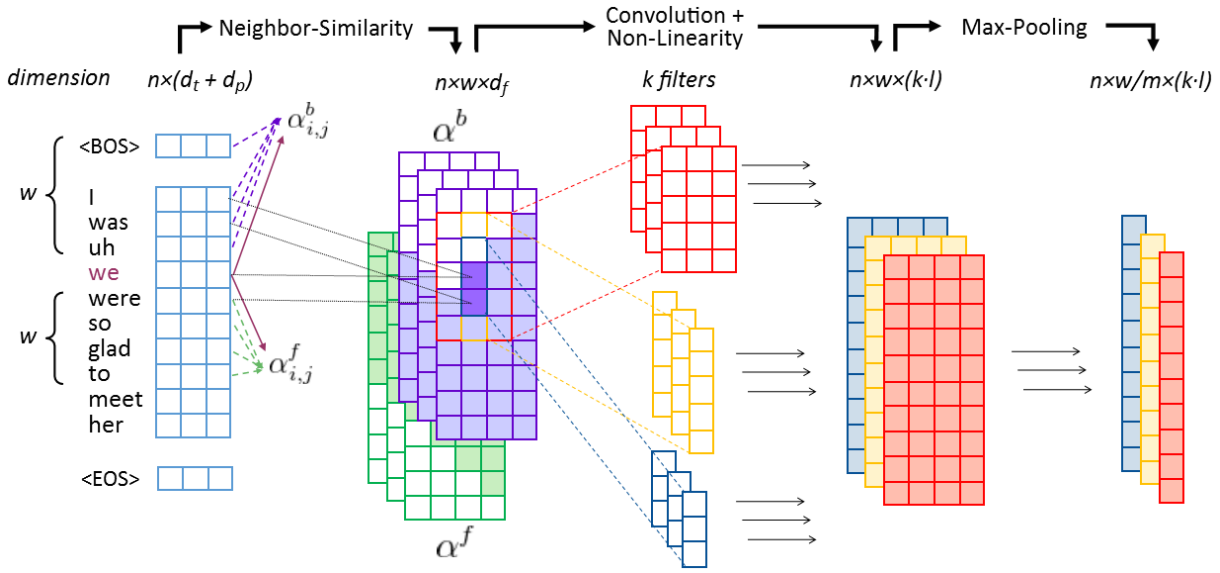


Figure 4.1: An illustration of the model. In this example, the backward neighbor-similarity layer α^b identifies that *"we"* has high similarity with *"I"* and *"were"* has high similarity with *"was"*. Both *"we"* and *"were"* are at a distance of 3 from the corresponding *"I"* and *"was"*. A convolutional filter can catch the horizontal pattern in row 3, thus indicating the presence of a bigram pattern match between *"we were"* and *"I was"*.

work, instead of manually defining similarity functions (e.g. exact match of the word/POS), we learn similarity functions between individual words in a sentence. For each word in a sentence of length n, we calculate a similarity between the given word x_i and each of the words x_j in the preceding/following window of size w, for $j \in [i \pm 1, \ldots, i \pm w]$. In our task we used cosine similarity sim to calculate the alignment score between a pair of words due to the straightforward resemblance of words in the reparandum and repair:

$$\alpha_{i,j}^{\{f,b\}} = sim(W^1 x_i, W^2 x_j) \tag{4.1}$$

where $W^1, W^2 \in \mathbb{R}^{d_f \times d_g \times (d_t + d_p)}$ are learned.

We refer to similarity scores in the preceding/following windows as α^b (backward) and α^f (forward), respectively. For the cases when the x_j is outside of the sentence boundaries, we set the similarity score to be zero. In order to capture multiple types of similarities between two word representations, we concatenate token and part-of-speech (POS) embeddings and

learn multi-dimensional similarity scores $\alpha_{i,j} \in \mathbb{R}^{d_f}$. In our experiments, we set $d_g = (d_t + d_p)$, where d_t and d_p are the dimensions of token and POS embeddings, respectively. The overall dimension of the similarity matrix is $\alpha \in \mathbb{R}^{w \times n \times d_f}$, where n is the sentence length and w is the size of the window.

Convolution over Similarity Features

While neighbor-similarity features can be useful, they do not exploit all the information about repeating patterns. A simple example can be a bigram pattern match feature: the model can find a similarity between closely related words on the unigram level, but it is unable to directly identify cases where the bigram would be repeated. To capture temporal patterns presented in neighbor-similarity scores, we apply convolutional filters on the output of the neighbor-similarity layer, followed by a non-linearity (tanh). For example, in Figure 4.1, the neighbor-similarity layer would identify similarity between individual tokens *"we"* and *"I"*, and *"were"* and *"was"*. A convolutional layer on top would capture the horizontal bigram pattern between *"we were"* and *"I was"*. The output of the convolutional layer is $f_{conv}(\alpha) \in \mathbb{R}^{w \times n \times kl}$, where k is number of different filter shapes and l is the number of output filters for each filter shape. We apply the max-pooling layer with a downsample rate m on top to summarize the convolutional layer output at each time i. The output of the max-pooling layer is $g(\alpha) \in \mathbb{R}^{w/m \times n \times kl}$. We flatten the outputs of the max-pooling layer and concatenate with the input feature embeddings, and input the resulting vector to an LSTM.

4.3.2 Experiments

Our experiments target both in-domain and cross-domain scenarios. In addition, we analyze the differences in errors made by the models.

Data

Switchboard (SWBD) [28] is the standard and largest corpus used for disfluency detection. The state-of-the-art in disfluency detection at the time of publication achieved F1 score of 91.1 on the SWBD test set [102]. In our experiments we found that tokenization and sentence segmentation differences can affect the performance significantly. In addition to Switchboard, we test our models on three out-of-domain publicly available datasets annotated with disfluencies [121]:

Switchboard: phone conversations between strangers on predefined topics;

CallHome: phone conversations between family members and close friends;

SCOTUS: transcribed Supreme Court oral arguments between justices and advocates;

FCIC: two transcribed hearings from Financial Crisis Inquiry Commission.

Model Comparisons

We train the CRF[3] and bidirectional LSTM-CRF[4] models as baselines, both with and without pattern match features. For simplicity, we refer to bidirectional LSTM-CRF model as just LSTM. We initialize word embeddings using 300 dimensional Glove embeddings [71], and a hidden dimension of LSTM unit of 150. In all models we use **identity features**, including word, POS tag, whether the word is a filled pause, discourse marker, edit word or fragment. The **hand-crafted pattern match features** include: distance to the repeated {word, bigram, POS, word+next POS, POS bigram, POS trigram} in the {preceding, following} window; whether word bigram is repeated in the {preceding, following} window allowing some words in between the two words; and distance to the next conjunction word. Following [119], we use 8 BIO states: BE (beginning of the reparandum), IE (internal reparandum state), IP (last word before an interruption point), BE_IP (single word reparandum), C (repair), C_IE (for nested disfluencies, a word is being in intermediate state of a reparandum

[3] `https://taku910.github.io/crfpp`

[4] `https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf`

Model	pattern	SWBD test	CallHome	SCOTUS	FCIC
CRF	–	71.3	58.1	70.8	53.2
	✓	82.5	63.2	79.2	63.8
LSTM	–	82.9	54.1	57.9	36.9
	✓	**86.8**	58.7	66.6	48.9
LSTM + sim	–	85.9	64.8	78.8	65.0
LSTM + sim + conv	–	86.7	**65.2**	**79.9**	**66.1**

Table 4.1: F1 scores on cross-domain disfluency detection; "pattern" stands for hand-crafted pattern match features.

and repair simultaneously), C_IP (same as C_IE but for the last word before an interruption point), and O (other, non-disfluency). For each experiment, we average the performance of 15 randomly initialized models.

For our proposed model we use the following parameters: window size $w = 10$, neighbor-similarity dimension $d_f = 100$, $k = 5$ different filter shapes: $[1, 1], [3, 1], [3, 3], [5, 1]$ and $[5, 3]$, with output filter dimension $l = 16$, and downsample rate $m = 3$. In our initial experiments we have tuned the parameters d_f, k, l and m to the values mentioned above using the Switchboard development set.

Results

The cross-domain experiment results are presented in Table 4.1. In general, for the in-domain data (Switchboard), the pattern match networks achieve performance comparable to the LSTM model with hand-crafted pattern match features, and significantly outperforms the CRF model. In addition, our model is robust compared to the baselines when applied to out-of-the-domain data, with a consistent improvement over the CRF. The LSTM performs poorly on out-of-the-domain data. To better understand the model differences, in the next section we conduct error analysis and discuss the findings.

Error analysis

The difficulty in applying the model on out-of-domain data lies in both difference in corpora and underlying model. There is substantial variation in vocabulary, conversational style, disfluency types, and sentence segmentation criteria across corpora. CallHome is more casual than SWBD; SCOTUS and FCIC are formal high stakes discussions with vocabularies highly dissimilar to SWBD. SCOTUS, FCIC, and CallHome contain 2, 5 and 7 times more restarts token-wise, respectively, than Switchboard. Also, on average, disfluencies in all three out-of-domain corpora tend to be longer, especially in CallHome and FCIC.

To further study the effect of pattern match features, we trained models with identity features only. When comparing models with identity features only, the CRF performs poorly compared to the LSTM on in-domain data. On the other hand, the LSTM with no pattern match features performs considerably well on in-domain Switchboard. By looking at cross-domain results, we see that the CRF is more stable across the domains, compared to the LSTM. We hypothesize that the LSTM is more powerful in learning specific data structure, compared to the CRF, and overfit the models to match Switchboard style. On the other hand, pattern match networks are better at capturing patterns that are more general across domains.

By looking closer at the results we found that there is a significant drop in precision for LSTMs with hand-derived features. In particular, it "hallucinates" a lot of disfluencies, longer ones in particular. This might be due to the long memory of the LSTM, as opposed to the CRF, which tends to be more local. Table 4.2 presents some examples with false positives made by the LSTM with hand-engineered features, but were correctly identify by our model.

The difficulty in applying the model on out-of-domain data lie in both difference in corpora and underlying model. There is substantial variation in vocabulary, conversational style, disfluency types, and sentence segmentation criteria across corpora. CallHome is more casual than SWBD; SCOTUS and FCIC are formal high stakes discussions with vocabularies

Corpus	Ex	Sentence
CallHome	1	Oh he [looks like + John Travolta but he has like] curly blond hair.
	2	[I do n't think + [I know her + but I 've]] heard of her
SCOTUS	3	What is your [authority + for that proposition, Mr. Guttentag, your case authority]?
	4	... as to permit review in [the court + of appeals, then the district court] habeas corpus procedure need ...
FCIC	5	Thank you for the opportunity [to + contribute to the commission 's work to] understand the causes of ...
	6	... counter parties were unaware [of + the full extent of] those vehicles and therefore could not ...

Table 4.2: Example sentences wrongly predicted as disfluent by LSTM model with hand-crafted pattern match features. The brackets indicate predicted disfluency regions, where the respective gold annotation is "non-disfluent".

highly dissimilar to SWBD. SCOTUS, FCIC, and CallHome contain 2, 5 and 7 times more restarts token-wise, respectively, than Switchboard. Also, on average, disfluencies in all three out-of-the-domain corpora tend to be longer, especially in CallHome and FCIC.

4.3.3 Analysis

By looking at the precision vs. recall of the models we noticed that the main drop in performance in LSTM with hand-derived features comes from low precision (Table 4.3). In particular, the LSTM with pattern match features "hallucinate" a lot of disfluencies, longer ones in particular. On the other hand if we look at the relative drop in performance between LSTMs with PMN vs. hand-crafted features, we can see that the relative difference in precision in CallHome and SCOTUS is similar, while FCIC is the most affected by the domain shift. SCOTUS and FCIC are two domains that are most affected by higher rates of

		CallHome	SCOTUS	FCIC
Model (Prec)	CRF	71.5	90.9	88.7
	LSTM	54.7	71.8	58.7
	LSTM PMN	68.4	88.0	87.2
Statistics	OOV in disfl	3%	6.4%	7.6%
	Repetitions	30.2%	59.5%	28.5%

Table 4.3: **Top:** Precision scores for CRF, LSTM with hand-engineered pattern match features and LSTM with PMN. **Bottom:** Statistics on percentage of OOV words that are part of a disfluency ($<$ 10 words in training set) and percentage of tokens in disfluencies that are repetitions.

out-of-vocabulary words (OOV)s in disfluencies (here defined as words that occur less than 10 times in the training set). While the OOV rates for FCIC and SCOTUS are similar, the drop in performance on SCOTUS is smaller, which can be explained by a higher number of repetitions which often are easier to predict.

Because low precision (high false detection rate) is the main problem for the LSTM with hand-engineered features, we look at examples of sentences (Table 4.4) with false positives that were correctly identify by the LSTM PMN model. Many examples here are exact match between the last word in the reparandum and the repair. In example (2), there is an "I" + verb pattern match. The analysis supports our hypothesis that the LSTM relies too much on the heuristic feature.

4.3.4 Conclusions

In this work we introduce a novel neural network architecture which allows automatic discovery of patterns and directly uses similarity scores as input features to a CNN. We show that our approach can be as effective as using carefully designed, hand-engineered pattern match features in a disfluency detection task, eliminating the need for feature engineering,

Corpus	Ex	Sentence
CallHome	1	Oh he [looks like + John Travolta but he has like] curly blond hair.
	2	[I do n't think + [I know her + but I 've]] heard of her
SCOTUS	3	What is your [authority + for that proposition, Mr. Guttentag, your case authority] ?
	4	. . . as to permit review in [the court + of appeals, then the district court] habeas corpus procedure need . . .
FCIC	5	Thank you for the opportunity [to + contribute to the commission 's work to] understand the causes of . . .
	6	. . . counter parties were unaware [of + the full extent of] those vehicles and therefore could not . . .

Table 4.4: Example sentences wrongly predicted as disfluent by LSTM model with hand-crafted pattern match features. The brackets indicate predicted disfluency regions, where the respected gold annotation is "non-disfluent".

and show that it is robust in cross-domain testing.

4.4 Prosody as an Observable Phrase Structure

4.4.1 How Might Prosody Help?

In this section, we hypothesize where prosody might help and look at the relative frequency of these cases and the performance of a high accuracy disfluency detection algorithm in these contexts.

Disfluency detection algorithms based on text alone rely on the fact that disfluencies often involve parallel syntactic structure in the reparandum and the repair, as illustrated in the previous examples. In these cases, pattern match provides a strong cue to the disfluency. In addition, ungrammatical function word sequences are frequently associated with disfluencies, and these are relatively easy for a text-based model to learn. In some cases, an interregnum

	Reparandum Length				% in
Type	**1-2**	**3-5**	**6-8**	**8+**	**type**
repetition	1894	419	12	1	46%
rephrase	794	585	66	–	28%
restart	196	14	–	–	4%
nested*	149	262	158	118	13%

Table 4.5: Total word counts associated with reparanda of different lengths and types of disfluencies. *Counts for nested disfluencies exclude repetition tokens.

	Reparandum Length				
Type	**1-2**	**3-5**	**6-8**	**8+**	**overall**
repetition	0.99	0.99	1	1	0.99
rephrase	0.75	0.66	0.44	–	0.70
restart	0.41	0	–	–	0.39
nested*	0.79	0.66	0.62	0.21	0.62

Table 4.6: Percent of reparandum tokens that were correctly predicted as disfluent. *Statistics for nested disfluencies exclude repetition tokens.

word (or words) provides a word cue to the interruption point. In the Switchboard corpus, only 15% of interruption points are followed by an interregnum, but it can provide a good cue when present. Prosody mainly serves to help identify the interruption point. Thus, for these types of disfluencies, it makes sense that prosodic cues would not really be needed.

Because disfluencies with a parallel syntactic structure do represent a substantial fraction of disfluencies in spontaneous speech, text-based algorithms have been relatively effective. The best models achieve F-scores of 86-92%[5] [54, 117, 104, 102]. We hypothesize that many

[5]It is difficult to directly compare published results, because there are different approaches to tokeniza-

	Reparandum Length	
Type	**1-2**	**3-5**
content-content	0.61 (30%)	0.58 (52%)
content-function	0.77 (20%)	0.66 (17%)
function-function	0.83 (50%)	0.80 (32%)

Table 4.7: Relative frequency of rephrases correctly predicted as disfluent for disfluencies that contain a content word in both the reparandum and repair (content-content), either the reparandum or repair (content-function) or in neither. Percentages in parentheses show the fraction of tokens belong to each category.

errors are associated with contexts where we expect that prosodic cues are useful, specifically the five cases below, with examples from the development set.

Restarts: Some disfluencies have no repair; the speaker simply restarts the sentence with no obvious parallel phrase.

```
[ it would be + ] I think it's clear...
well [the +] uh i think what changed...
```

Long disfluencies: These include distant pattern match or substantial rephrasing.

```
[there is + for people who don't want to do the military service it would be
neat if there were]
[what they're basically trying to do + i don't know up here in massachusetts
anyhow what they're basically trying to do]
```

tion that have a non-trivial impact on performance but are not well documented in the literature. Those differences include handling of fragment words, turn boundaries, and tokenization. For example, some studies use fragment features explicitly, while others omit them because speech recognition systems often miss them. Turn boundaries that do not end with a slash unit pose an ambiguity during speaker overlap: cross-turn 'sentences' can either be combined into a longer sentence or separated based on the turn boundary, which impacts what can be detected. Lastly, there are differences in whether contractions and possessives are split into two tokens, and whether conversational terms such as "you know" are combined into a single token.

Complex (nested) disfluencies: Disfluencies can occur within other disfluencies.

```
[really + [[i + i] + we were really]...
[[to + to try to] + for two people who don't really have a budget to] ]...
```

Non-trivial rephrasing: Rephrasing does not always involve a simple "rough copy" of a repair.

```
[can + still has the option of]...
to keep them [in + uh quiet ]...
```

Fluent repetitions: Contexts with fluent repetitions often include expressing a strong stance.

```
a long long time ago...
she has very very black and white...
```

In order to confirm that there is potential for prosody to help in these contexts, we first categorize the disfluencies. To avoid hand-labeling of categories, we distinguished disfluencies based on surface forms (repetition, rephrase, restart) and length of the disfluency reparandum. Word counts for the different categories are given in Table 4.5.

Conditioning on the different contexts, we analyze errors in the development set made by the high accuracy text-based disfluency detection system that is the baseline for this study [117]. For this model, trained on Switchboard, the performance is 87.4 F-score (P=93.3, R=82.2) on the development set and 87.5 (P=93.1, R=82.5) on the test set. For each class, we measured the disfluency detection recall (relative frequency of reparandum tokens that were predicted correctly), as well as the percentage of tokens associated with each class. The results in Table 4.6 confirm that error rates are higher for restarts, longer rephrasings, and complex disfluencies.

Rephrase disfluencies include both short lexical access errors, as well as non-trivial rewordings, which tend to be longer and involve content words. Table 4.7 breaks down performance

for different lengths and word class to explore this difference. We found that rephrase disfluencies that contain content words are harder for the model to detect, compared to rephrases with function words only, and error increases for longer disfluencies.

Finally, the relative frequency of false positives in fluent repetitions is 0.35. Since fluent repetitions account for only 4% of all repetitions, the impact on overall performance is small.

The ultimate goal of a disfluency detection system is to perform well in domains other than Switchboard. Other datasets are likely to have different distributions of disfluencies, often with a higher frequency of those that are hard to detect, such as restarts and repairs [121]. In addition, due to the differences in vocabulary, disfluencies with content words are more likely to get misdetected if there is a domain mismatch. Thus, we hypothesize that prosody features can have a greater impact in a domain transfer scenario.

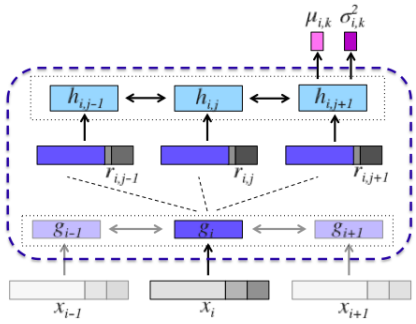
4.4.2 Methods

Integrating prosodic cues has proved difficult because of the many sources of variability affecting the acoustic correlates, while systems that only use text achieve high performance. In this work, we propose a new approach that operates on differences in information found in text and prosody. In order to calculate such differences, we introduce innovation features, similar to the concept of innovations in Kalman filters. The key idea is to predict prosodic features based on text information, and then use the difference between the predicted and observed prosodic signal (innovations) as a new feature that is additionally used to predict disfluencies.

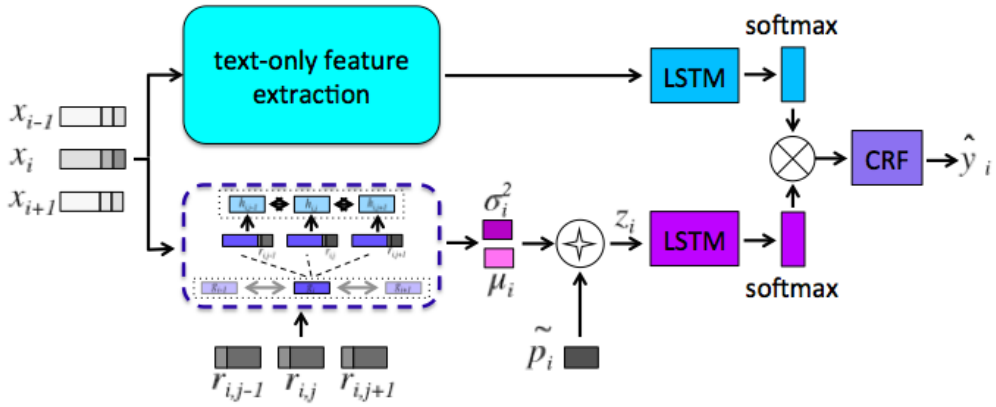
Let a prosody cue, p_i at time i be an observation associated with a sentence transcript containing n word tokens, $x_0 \dots x_n$. This observation can be modeled as a function of the sentence context $H(x_0 \dots x_n)$ perturbed with Gaussian noise $v_i \sim \mathcal{N}(0, \sigma_i^2)$:

$$p_i = H(x_0 \dots x_n) + v_i \tag{4.2}$$

v_i can be viewed as a difference in information found between text and prosody. This difference can be measured using a z-score, which is a measure of how many standard deviations



(a) Prosody prediction model



(b) Late fusion model

Figure 4.2: Prosody prediction (**top**) and late fusion (**bottom**) models. x_i is a contcatenation of token, POS and identity features embeddings at time i; r_i, j is a concatenation of stress and phone embeddings for phone j in token i; $\widetilde{p}_i$ is a vector of prosodic cues; g_i and h_i are hidden states of token level and phone level LSTMs, correspondingly.

below or above the population mean an observation is. This framework can be viewed as a non-linear extension of a Kalman filter, where both H and σ_i^2 are parametrized using a neural network. Since disfluencies are irregularities in spoken language, they can be considered anomalies to fluent speech flow. A prosody flow that is unusual for a given word sequence, such as one that happens at interruption points, will likely have higher deviation from the predicted distribution. This anomaly in speech flow provides a strong signal when extracted using innovations, which is complementary to the text cues. In the next sections we give more details about the neural network architecture for text encoding, prosodic cues

and innovation features, as well as an overview of the whole system.

Text Encoding for Prosody Prediction

We use both context around a word as well as subword information in text encoding for prosody prediction. Our text encoding consists of two bidirectional LSTMs: one on the token level and another on the phone level. First, we use pretrained word embeddings [47], part-of-speech tags embeddings, and identity features (whether the word is a filled pause, discourse marker, or incomplete) as inputs to a word-level bidirectional LSTM. Then, for each phone in a word we concatenate the phone embedding, its stress embedding, and the hidden state of the word-level LSTM for the corresponding token. The resulting phone feature vector is used as input to the second bidirectional LSTM. The last hidden state h_i of this second LSTM for token i summarizes the phone, stress and context information of that token, which we use to predict word-level prosodic cues. We use 3 categories of stress features in our experiments: primary, secondary and a non-stress phone.

Prosodic Cues

Our prosodic cues include:

Pause. Given a pause before a word, r_i, our pause cues are scaled as follows:

$$\widetilde{r_i} = \min(1, \ln(1 + r_i)) \tag{4.3}$$

Pause information is extracted on a word-level using Mississippi State (MsState) time alignments (more details on data preprocessing in Section 4.4.3.) We use scaled real-valued pause information. Scaling pause lengths this way, including the threshold for pauses longer than 1 sec (which are rare), makes the pause distribution less skewed.

Word Duration. Similar to pause information, we extract word duration information using MsState time alignments. We do not need to do the standard word-based duration normalization, since the idea behind the innovation model is to normalize prosodic features using a richer context representation.

Fundamental frequency (F0) and Energy (E). Similar to Tran *et al.* [99], we use three F0 features and three energy features. The three F0 features include normalized cross correlation function (NCCF), log-pitch weighted by probability of voicing (POV), and the estimated delta of log pitch. The three energy features include the log of total energy, the log of total energy from lower 20 mel-frequency bands and the log of total energy from higher 20 mel-frequency bands. The contour features are extracted from 25-ms frames with 10-ms hops using Kaldi [73]. Our model is trained to predict the mean of these features across the frames in a word.

MFCCs. In addition to features used in Tran *et al.* [99], we also use 13 mel-frequency cepstral coefficients, averaged at the word level, similar to F0 and energy features as described above.

Prosody Innovation Cues

Given a word-level text encoding h_i, for each token in a sentence we predict each of the k prosodic cues $\widetilde{p_i}^k$ listed above. We assume that the predicted prosody cues conditioned on text have a Gaussian distribution:

$$\begin{aligned} \widetilde{p_i}^k | h_i &\sim \mathcal{N}(\mu_{i,k},\, \sigma^2_{i,k}) \\ \mu_{i,k} &= f(W_1^k h_i + b_1^k) \\ \sigma^2_{i,k} &= softplus(W_2^k h_i + b_2^k) \end{aligned} \tag{4.4}$$

W_1^k, b_1^k, W_2^k, b_2^k are learnable parameters; the activation function

$$softplus(x) = \log(1 + \exp(x))$$

ensures that the variance is always positive; f is an activation function, which is $softplus$ for pauses and durations, and $tanh$ for the rest of the prosodic cues. The objective function is a sum of the negative log-likelihood of prosodic cues $\widetilde{p_i}^k$ given text encoding. Then, given the predicted $\mu_{i,k}$, $\sigma^2_{i,k}$ and true values of prosodic cues $\widetilde{p_i}^k$, we calculate z-scores for each of

the cues, which should have high absolute value for tokens with unusual prosodic behaviour:

$$z_i^k = \frac{\widetilde{p}_i^k - \mu_{i,k}}{\sigma_{i,k}} \tag{4.5}$$

The prosody prediction module is illustrated in Figure 4.2(a).

These z-scores, or *innovations*, are used as additional features in our disfluency detection model. We train the prosody prediction model only on sentences that do not contain any disfluencies. Any unusual behaviours in disfluency regions, therefore, should have large innovation values predicted by our model.

Disfluency Detection System

Following our pattern-match neural network described in detail in Section 4.3.1, we use a bidirectional LSTM-CRF model as our disfluency detection framework. The overall performance is measured in F-score of correctly predicted disfluencies in the reparandum. Previous work used textual features only. Here, we evaluate the importance of innovation cues with two types of multimodal fusion - early and late fusion. In early fusion, we concatenate innovations and/or prosody features with the rest of the textual features used in the framework at the input to LSTM layer. In late fusion, we create two separate models - one with only textual features and another with innovations and/or prosody features. Then we do a linear interpolation of the states of two models just before feeding the result to the CRF layer:

$$u_i^{shared} = \alpha u_i^{prosody} + (1-\alpha) u_i^{text} \tag{4.6}$$

We tune the interpolation weight α and report the best in our experiments section. We train our model jointly, optimizing both prosodic cues prediction and disfluency detection. The schematic view of the late fusion system is presented in Figure 4.2(b).

4.4.3 Experiments

In our experiments we evaluate the usefulness of innovation features, and compare it to baselines with text-only or raw prosodic cues. For each model configuration, we run 10

	Model	dev		test		α
		mean	best	mean	best	
single	text	86.54	86.80	86.47	86.96	–
	raw	35.00	37.33	35.78	37.70	–
	innovations	80.86	81.51	80.28	82.15	–
early	text + raw	86.46	86.65	86.24	86.53	–
	text + innovations	86.53	86.77	86.54	87.00	–
	text + raw + innovations	86.35	86.69	86.55	86.44	–
late	text + raw	86.71	87.05	86.35	86.71	0.2
	text + innovations	**86.98**	**87.48**	**86.68**	**87.02**	0.5
	text + raw + innovations	86.95	87.30	86.60	86.87	0.5

Table 4.8: F1 scores on disfluency detection when using a single set of features (text-only, raw prosody features or innovation features), with early fusion and late fusion. "Raw" indicates the usage of original prosodic features (Section 4.4.2), while "innovations" indicate the usage of innovation features (Section 4.4.2).

experiments with different random seeds. This alleviates the potential of making wrong conclusions due to "lucky/unlucky" random seeds. We report both the mean and best scores among the 10 runs.

Data Preprocessing

Switchboard [28] is a collection of telephone conversations between strangers, containing 1126 files hand-annotated with disfluencies. Because human transcribers are imperfect, the original transcripts contained errors. MsState researchers ran a clean-up project which hand-corrected the transcripts and word alignments [21]. In this work, we use the MsState version of the word alignments, which allows us to extract more reliable prosodic features. Since the corrected version of Switchboard does not contain updated disfluency annotations, we

but it 's just you know leak leak leak everywhere
people should know that that 's an option
and i think you do accomplish more after that
i mean [it was + it]
interesting thing [about gas is when + i mean about battery powered cars is]

Table 4.9: Examples of sentences where prosody innovations help. Words in red are correctly labeled when using prosody but not with text only. The first three show fluent phrases; the last two have disfluencies that are missed without prosody.

i like to run [about + oh about] [two + two and a half] miles
the old-timers even the people who are technologists do n't know how to operate
i do n't know whether that 's because they you know sort of give up hope
it must be really challenging to um try to juggle a job

Table 4.10: Examples of the sentences where prosody innovations hurt. Words in red are incorrectly labeled when using prosody but not with text only. The first shows a disfluency missed when using prosody; the other three are fluent regions with false detections.

corrected the annotations using a semi-automated approach: we used a text-based disfluency detection algorithm to re-annotate tokens that were corrected by MsState, while keeping the rest of the original disfluency annotations. The result is referred to as a silver annotation. Most of the corrected tokens are repetitions and restarts. To assess the quality of the automatic mapping of disfluencies, we hand-annotated a subset (6.6k tokens, 453 sentences) of the test data and evaluated the performance of the silver annotation against the gold annotation, which has an F1 score of 90.1 (Prec 90.1, Rec 90.1). Comparing the performance estimates from gold and silver annotations on this subset, we find that the silver annotations give somewhat lower F1 scores (2-3% absolute), both due to lower precision and recall scores.

Results

Our experiments evaluate the use of innovations with two popular multimodal fusion approaches: early fusion and late fusion. Our baselines include models with text-only, prosody cues only (raw), and innovation features only as inputs. Since innovations require both text and raw prosodic cues, this baseline is multimodal. In addition, for the late fusion experiments, we show the optimal value of α, the interpolation weight from Equation 4.6. All experiment results are presented in Table 4.8.

We found that innovations are helpful in both early and late fusion frameworks, while late fusion performs better on average. The interpolation weight α for the late fusion experiments is high when innovations are used, which further indicates that innovation features are useful in overall prediction. Interestingly, innovation features alone perform surprisingly well. We also take a closer look at the importance of joint training of the disfluency detection system with prosody prediction. To do this, we pretrain the prosody prediction part of the model first. Then, we train the full model with innovation inputs while freezing the part of the network responsible for predicting prosodic cues. The mean F-score of this disjointly trained model is 49.27% on the dev set, compared to 80.86% for the jointly trained model. This result suggests that training the system end-to-end in a multitask setup is very important.

4.4.4 Analysis

Error analysis

In order to better understand the impact of the prosody innovations, we perform an error analysis where we compare the predictions of two models: a late fusion model that uses both text and innovation features, and a baseline model that uses text only. All of the analysis is done on the dev set with the model that has the median performance out of 10 that were trained.

First, we extract all the sentences where the number of disfluency detection errors using the innovation model is lower than when using the text-only model (168 sentences). Examples

of such sentences are presented in Table 4.9. By looking at the sentences where the model with innovations performs better, we see fluent repetitions and other ambiguous cases where audio is useful for correctly identifying disfluencies.

On the other hand, in Table 4.10, we have examples of sentences that have a higher number of errors when prosody is used (143 sentences). In the first example, the labeling of "two" as fluent by the model with prosody is arguably correct, with the repetition indicating a range rather than a correction. The next involves a parenthetical phrase, the start of which may be confused with an interruption point. In the last two cases, there is a prosodic disruption and an interegnum, but no correction.

In order to understand whether incorporating prosody through our model supports the hypotheses in Section 4.4.1, we compare the performance of two models for different categories of disfluencies. We found that using prosody innovations improves detection of: non-repetition disfluencies (from 68.2% to 73.7%), particularly for disfluencies with content words (65.2% to 71.0%); long repairs (64.0% to 72.7% and 40.0% to 64.6% for disfluencies with length of repair greater than 3 and 5 correspondingly); and restarts (from 36.0% to 37.4%). Prosodic innovations also help decrease the rate of false positives for fluent repetitions: the false positives rate decreased from 46.5% to 38.4%. However, the prosody model increases the false positives in other contexts, such as in the examples in Table 4.10.

Innovation Predictors

In order to understand what the model actually learns with respect to innovations, we look at innovation distributions for words preceding interruption points compared to fluent words. The histograms are presented in Figure 4.3. As expected, we see that words preceding interruption points have atypically longer duration and lower energy. The intonation features did not show substantial distribution differences, probably due to the overly simplistic word-level averaging strategy.

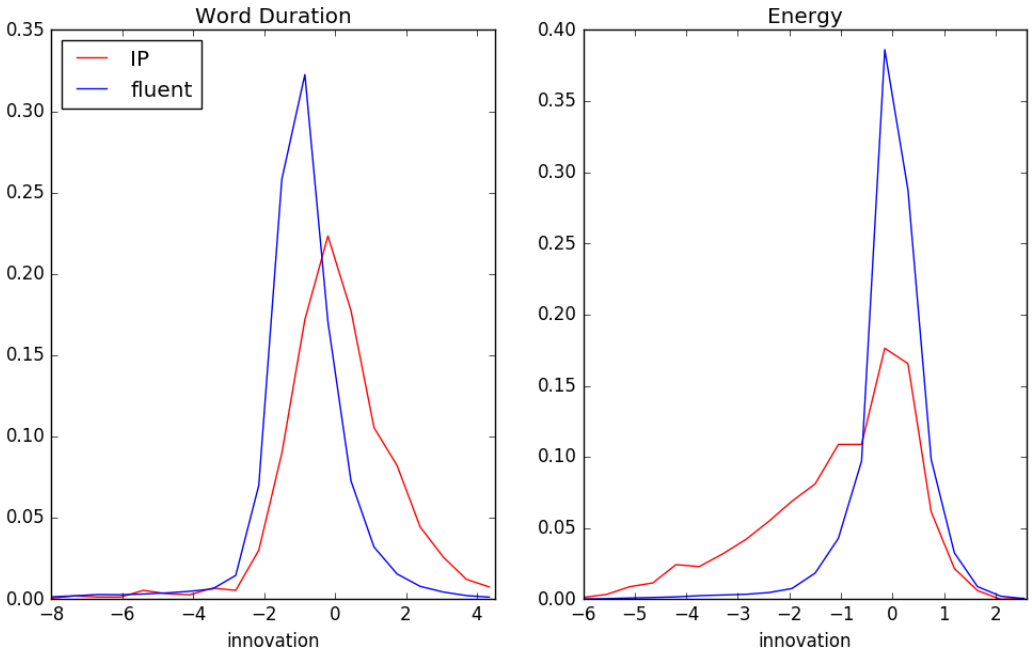


Figure 4.3: Histogram of innovations for word duration and energy features for words preceding an interruption point vs. fluent words.

4.4.5 Conclusions

In this work, we introduce a novel approach to extracting acoustic-prosodic cues with the goal of improving disfluency detection, but also with the intention of impacting spoken language processing more generally. Our initial analysis of a text-only disfluency detection system shows that despite high performance of such models, there exists a big gap in the performance of text-based approaches for some types of disfluencies, such as restarts and non-trivial or long rephrases. Thus, prosody cues, which can be indicative of interruption points, have a potential to contribute towards detection of more difficult types of disfluencies. Since the acoustic-prosodic cues carry information related to multiple phenomena, it can be difficult to isolate the cues that are relevant to specific events, such as interruption points. In this work, we introduce a novel approach where we extract relevant acoustic-prosodic information using text-based distributional prediction of acoustic cues to derive vector z-score features, or

innovations. The innovations point to irregularities in prosody flow that are not predicted by the text, helping to better isolate signals relevant to disfluency detection that are not simply redundant with textual cues. We explore both early and late fusion approaches to combine innovations with text-based features. Our experiments show that innovation features are better predictors of disfluencies compared to the original acoustic cues.

Our analysis of the errors and of the innovation features point to a limitation of the current work, which is in the modeling of F0 features. The current model obtains word-based F0 (and energy) features by simply averaging the values over the duration of the word, which loses any distinctions between rising and falling F0. By leveraging polynomial contour models, we expect to improve both intonation and energy features, which we hope will reduce some of the false detections associated with emphasis and unexpected fluent phrase boundaries.

An important next step is to test the system using ASR rather than hand transcripts. It is possible that errors in the transcripts could hurt the residual prediction, but if prosody is used to refine the recognition hypothesis, this could actually lead to improved recognition. Finally, we expect that the innovation model of prosody can benefit other NLP tasks, such as sarcasm and intent detection, as well as detecting paralinguistic information.

Chapter 5
TABLE STRUCTURE

5.1 Introduction

Tables are a common type of information representation used across the internet. With billions of search queries a day,[1] question answering on tables is an important task that translates into a large number of search queries every second about information present in tables. In general, research on Question Answering (QA) can be categorized in terms of the resources that are used in answering the question: text documents (often referred as *unstructured* text in the literature), tables, or a structured knowledge base (KB). In our work we are interested in the combination of text-based and structured resources for question answering, particularly articles that contain both tables and text. This is a natural next step for question answering on tables, in that most tables are embedded in documents that discuss them, creating the challenge of determining whether the answer is in the text or the table (if anywhere). In addition, very often information presented in tables is compact and abbreviated. The associated text provides rich context that can be used to enhance the representation of the table for more robust question answering.

Consider the example in [69] of the table of summer Olympic games shown in Figure 5.1. This is one of four tables in the associated Wikipedia article. The heading of the first column is "Olympiad," but in the text they are most often referred to as "games". The heading "games" is used in a different table for a column related to the number of Olympiads that a country has participated in. Linking the phrase "The United States hosted the games four times (in 1904, 1932, 1984 and 1996)" to the related cells in the table provides a mechanism to interpret that relation for other countries.

---

[1] https://www.internetlivestats.com

The 1896 Summer Olympics, officially known as the Games of the Olympiad, was an international multi-sport event which was celebrated in Athens, Greece, from 6 to 15 April 1896. It was the first Olympic Games held in the Modern era. About 100,000 people attended for the opening of the games. The athletes came from 14 nations, with most coming from Greece.

The Games have been held three times in the United Kingdom (in 1908, 1948 and 2012); twice each in Greece (1986, 2004), France (1900, 1924), Germany (1936, 1972) and Australia (1956, 2000); and once each in …

Olympiad	Year	Host	Opened by	Dates	Nations
I	1896	Athens, Greece	King George I	6-15 April	14
XXVIII	2004	Athens, Greece	President Konstantinos Stephanopoulos	13-29 August	201

Figure 5.1: Fragments of text and a table from the Wikipedia article on Summer Olympics Games with correspondences underlined.

The main focus of this chapter is to investigate how to improve question answering on documents that contain both text and tables. While recently there has been a lot of interest in reading comprehension for both text and tables, little research has been done in combining the two sources of information. The only prior study we are aware of is by Chen *et al.* [16] who introduced a new dataset for multi-hop QA over tabular and textual data. In their work, the authors heavily rely on the assumption that the questions would be unanswerable if text or table information is missing. On the other hand, here we investigate a more realistic scenario of naturally occurring questions, where the answer can be found in either text, tables, both, or neither. We evaluate our approach on the Natural Questions corpus [42] which consists of real anonymized queries issued to the Google search engine and corresponding Wikipedia articles, simulating a real use case of such system.

Prior work on the Natural Questions dataset has treated text and tables uniformly, linearizing tables and representing them and text segments using the same contextual token representations (for example, starting from pre-trained transformers [100] like BERT [22]). However, representations developed for text are sub-optimal for tables, since they do not account for the special relationships between table cells, defined by the row and column structure. In this work, we extend the BERT architecture to account for inter-cell relationships in tables. This approach is motivated by Graph Neural Networks with a transformer [82] and is closely related to the one in Müller *et al.* [62]. In our work, we pretrain parameters for the new relationships using a large table corpus extracted from Wikipedia [9].

In addition, we present a novel approach that refines table representations by attending to related representations of text in the surrounding article. This allows information to propagate from the text to table elements, improving the ability of the model to interpret tables and find answers in them.

Overall, the main contributions of our work are two-fold. First, to the best of our knowledge, this is the first work that investigates how to effectively combine text-based and table-based approaches in a setting where it is unknown which, if any, of these modalities contains an answer to a question. We introduce a novel mechanism to enrich table representations based on text surrounding the table, which improves the performance of a model for question answering from tables. Secondly, this is the first model that uses the Natural Questions corpus for question answering on tables, improving the baseline in Alberti *et al.* [1] that does not distinguish between tables and text.

5.2 Related Work

Most work on QA with tables prior to BERT involves first converting the table to a Knowledge Graph (KG) where cell entries are entities with row/column relations, then using entity linking to identify spans in the question that match an entity in the knowledge graph, and finally parsing the question to generate an SQL query using some variant of a sequence-to-sequence model [41]. Due to the advances in contextualized word embeddings, more recent work proposed a modification of the BERT transformer architecture to be used for representing tables. Hwang *et al.* [37] proposed the usage of additional `[SEP]` tokens between headers of the table to make BERT model more suitable for the tables. Our approach for table encoding is most similar to that of Müller *et al.* [62], where the authors generalized the BERT architecture similarly to [82] with new types of relations to encode table-specific relationships. The main differences between our table representation and Müller *et al.* [62] is that in our representation we use 5 types of relations, cell-column, cell-row, in-cell, cross-column and cross-row (more details in Section 5.3.1), while in their work the authors use cell-column and cell-row relations only for the table representation, but in addition use question-cell rela-

tions for marking matches between tokens in the question and corresponding cell values. In addition, their approach relies on having relations on a cell level rather than on a token level. The above mentioned paper includes additional relations based on n-gram matches with the question, and special processing of numerical values. Finally, the two most recent works on table representation learning [33, 113] also use pretraining on the Wikitables dataset [9] that we use in our work. Therefore, our table representations based on transformers and method to pretrain them, are comparable to those in recent and concurrent works.

Leveraging tables is a hard problem. However, most studies on table-based QA omit an important additional information source: the text in the article discussing the table. Prior attempts at integrating a KB and text use early fusion of document text and KG information [91], where they integrate text and a KG sub-graph in a single graph, from which an entity is selected to answer the question. Structured KGs are often easier to interpret than tables, which have a wide variety of possible schemas. In this work, we extend the transformer architecture to incorporate textual context and enrich the representations of tables.

The Natural Questions is a large corpus that contains real user queries along with their corresponding Wikipedia articles, which may or may not contain an answer anywhere in the article. Alberi *et al.* [1] provided a BERT-based baseline that treats both table and text segments like text: a sequence of tokens with word and position embeddings. Recently, Liu *et al.* [50] improved this baseline by using dynamic dual-attention over paragraphs and cascade answer predictor. In another direction, Ravula *et al.* [78] used an extended transformer architecture that models extra-long documents with limited propagation of information among different segments. All three approaches did not distinguish between text and table input, treating tables as text while not taking into account table structure. To the best of our knowledge, this is the first work that focuses on table-based QA for the real user queries in the Natural Questions corpus, and shows that table-based and overall QA performance can be improved by building on state-of-the-art pre-trained representations of table structure, additionally enriched via attention to related article text.

5.3 Methods

This section describes two methods that improve table representations for QA from tables: an extension of the BERT architecture for table encoding to better capture the relationships between table elements, and a mechanism that incorporates related unstructured text of an article as context to further improve the table representation. We then describe an approach for question answering from both text and tables that combines predictions from text QA and table QA models using a late fusion approach.

5.3.1 Table Encoding

Our table encoding is a generalization of the transformer architecture [100], with the self-attention sub-layer extended to incorporate relations between structural components in tables, similar to the one introduced by Müller *et al.* [62]. This approach is motivated by Graph Neural Networks (GNNs) with a transformer [82, 81] where the authors generalize the transformer by introducing multiple types of relations between inputs.

GNNs with a Transformer

In the original transformer, the multi-head self-attention for head h is calculated using query (Q), key (K) and value (V) projections as follows:

$$Att(Q,K,V) = \\ softmax(\frac{(W_h^Q Q)^T (W_h^K K)}{\sqrt{d_k}}) W_h^V V$$

where W_h^Q, W_h^K and W_h^V are learned parameters, and d_k is the query/key dimension. By calculating the dot product between query and key projections, a transformer captures the interaction between each pair of inputs x_i and x_j (e.g. wordpieces in BERT) at positions i and j for $0 \leq i, j \leq N$. This interaction can be generalized to account for relation type t

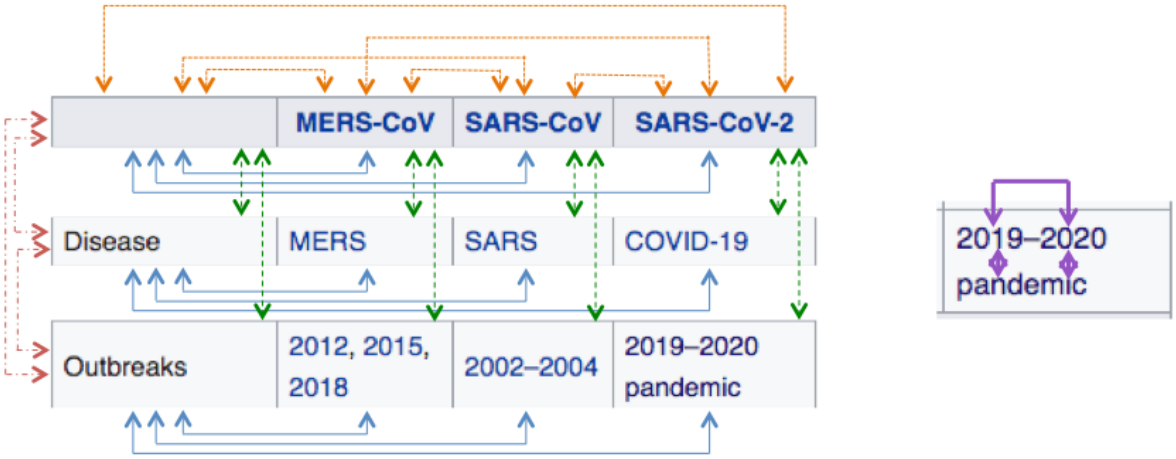


Figure 5.2: Example of a table encoding with relations: (*green, dashed*) token cell - token column header relations; (*blue, solid*) token cell - token row header relations; (*violet, bold*) in-cell token relations; (*red, dash-dotted*) cross-column header relations; (*orange, dotted*) cross-row header relations.

between x_i and x_j by biasing the key projection using r^t_{ij}:

$$s^h_{ij} = \frac{(W^Q_h x_i)^T (W^K_h x_j + r^{t,h}_{ij})}{\sqrt{d_k}}$$

and then scaling using softmax across all inputs $0 \leq j \leq N$. Thus, the standard transformer can be considered as a special case with $r_{ij} = 0$. Similarly, the value projection can also be updated with the corresponding relation type represented using the bias term ρ^t_{ij}, with the overall attention head h calculated as follows:

$$\alpha^h_{ij} = softmax(\frac{s^h_{ij}}{\sum_j s^h_{ij}})$$
$$w^h_i = \sum_j \alpha^h_{ij} (W^V_h x_j + \rho^{t,h}_{ij})$$

The parameters $r^{t,h}$ and $\rho^{t,h}$ are head- and layer-specific.

Table Encoding using GNN

Table structure can be encoded using the model described above to account for special relations in the table. Here, we use the following relation types:

- token cell - token column header relation
- token cell - token row header relation
- in-cell token relations
- cross-column header relations
- cross-row header relations

For each of these table-specific types of relations we learn different type-specific biases r^t and ρ^t for each layer and head, while for the rest of the relations we use the original BERT configuration with zero bias. Figure 5.2 shows an example of a table with the table relations used in this work.

Our table encoding is similar to the one independently proposed by Müller *et al.*[62] with the main difference of having relations on a token level rather than cell level. Also, in their approach, the authors use only cell-column and cell-row relation, while in our work we also use cross-column header, cross-row header and in-cell relations. The above mentioned paper includes additional relations based on n-gram matches with the question, and special processing of numerical values.

Pretraining

The main motivation for applying the original BERT model to table encoding is to use contextualized embeddings that are pretrained on a large amount of data. The new relation biases incorporated as part of the proposed table encoding are randomly initialized, starting from a standard transformer model pre-trained on text. Since these parameters are added for each layer, they can significantly change the activations of the pretrained model. In order

to derive a better initialization point for the additional bias terms, we pretrain a GNN model with the masked LM objective used in BERT the Wikitables dataset [9], which contains 1.6M tables from English Wikipedia. In order to limit the amount of overfitting on that table set, we freeze all original BERT parameters while updating only the bias terms introduced in the GNN. We tune the model using perplexity on a subset of the Wikitables dataset.

5.3.2 Context-aware Table Representation

We hypothesize that text in the article of a web page containing a table can help build an improved representation of the table for QA. Recent work has explored building encoders over large input sequences, including [78] scaling input sequence length to more than 8,000 tokens for the NQ dataset. However, to make the model efficient, encodings of individual text or table segments communicate through single-vector global memories. Here, we take the approach of using asymmetric attention from table token representations to a small number of relevant text token representations, that are pre-computed independently. Our approach is more similar to the handling of prior segment context in Transformer-XL [20], but relevant context is selected based on word overlap and not contiguity.

The two components of our approach, described next, include the definition of relevant text context for table elements and the mechanism for using contextualized embeddings of the relevant text to enrich the table token representations.

Table-Textual Context Linking

Let a table cell that contains a sequence of input tokens be defined as $(u_0^t \dots u_K^t)$, with the corresponding s sub-word units (wordpiecies in BERT or byte-pair encodings in RoBERTa) for the k-th word to be defined as $(x_0^{t,k} \dots x_{S_k}^{t,k})$, and let the textual context of the article surrounding the table be defined as $(u_0^c \dots u_N^c)$, with the corresponding sub-word units for the n-th word be as $(x_0^{c,n} \dots x_{S_n}^{c,n})$. For each word in the table u_i^t, we find the corresponding context in the text using the exact match of the lower-case sequence of tokens, starting with

the trigram matches, following by bigram and unigram matches.[2] For example, a trigram match for a word u_i^t is $\alpha_3(u_i^t) = u_j^c$ if a lower-case expression $(u_i^t, u_{i+1}^t, u_{i+2}^t)$ equals to the lower-case expression $(u_j^c, u_{j+1}^c, u_{j+2}^c)$. For each of the table tokens u_k^t we collect up to 6 corresponding matches from the text, $u_{s1}^c \dots u_{s6}^c$, and extract their sub-word embeddings represented by the last layer of pretrained RoBERTa, $e(x_0^{c,s1}) \dots e(x_s^{c,s6})$. Then, we stack all the sub-word unit embeddings associated with u_k^t to get a text-aware representation for the word k in table $e(u_k^t) = [e_0(x_0^{c,s1}); \dots; e_r(x_s^{c,s6})] \in \mathbb{R}^{r \times d}$, with d being a hidden size of the RoBERTa embedding and r being the total number of sub-word units corresponding to $u_{s1}^c \dots u_{s6}^c$. For simplicity of implementation, we use $r = 12$, where we prune any extra word-piecies or use padding for the cases where $r \neq 12$.

Enriching Table Representations

In order to incorporate text context $e(u_i^t)$ into a table representation of $x_s^{t,i}$, we use a slightly modified version of self-attention in the transformer, with the goal of generating an additional text context head, concatenated with the rest of the attention heads across multiple transformer layers that would be aware of the broader text context.

In the original transformer, self-attention is calculated between an input at position i using a query projection $W^Q x_i$ and the rest of the inputs in positions $j \in 0 \dots N$ using key and value projections, $W^K x_j$ and $W^V x_j$, respectively. Now, when calculating a text-aware head for each of the sub-word units $x_i^{t,k}$, we use attention between a query of $x_i^{t,k}$ and all of the sub-word units of the corresponding text context $e(u_k^t)$ using the corresponding key and value projections $W_e^K e(u_k^t)$ and $W_e^V e(u_k^t)$:

$$Att(Q_e, K_e, V_e) = softmax(\frac{(W_e^Q Q)^T (W_e^K e(K))}{\sqrt{d_k}}) W_e^V e(V)$$

[2]Function words are frequent in multi-word expressions. To avoid exact match of expressions solely consisting of function words, matching expressions must contain at least one non-frequent word, defined based on the 200 most frequent words in the training set and the NLTK stop-word list.

Then, we concatenate the new text-aware head with the rest of the heads in the layer h, resulting in a total of $m + 1$ heads ($m = 16$ for RoBERTa$_{\text{large}}$), each of a size $k = 64$. In addition, we extend the current projection layer of a size $km \times km$ that used after the concatenation of heads in order to fit the additional head by randomly initializing additional $k \times km$ parameters. For computational efficiency, we incorporate the text-aware representation only at layers 12, 16, 20 and 24.

5.3.3 Combining Text and Table Answer Predictions

In end-user question answering, both text and table contexts can be used to support meeting the user information need. Question answering systems should therefore be able to consider both sources of information to present the most suitable answer. So far we have presented enriched table representations that can be used for question answering from tables. We now consider approaches for the full document-level QA task, where an answer may be found in either or none of the two modalities.

Since a text-based QA model would not benefit from the architecture and pre-training extensions for our table representations, we use a standard text-based representation for QA from text. We combine predictions from two separate models for the full document-level QA task. Specifically, we train a generic model for full article question answering following [1]. This model assigns scores to candidate answers in both text and tables using a standard pre-trained text representation (RoBERTa). We also train a separate model which uses enriched table representations and pre-training, and focuses on predicting answers in tables. The two model predictions are combined using a late fusion approach detailed below.

Calculating Prediction and Confidence Scores

We follow Alberti *et al.* [1] to define a loss function of training and an answer span prediction method. More specifically, at inference time the scores that correspond to the start and end

of a possible answer span are defined as follows:

$$\begin{aligned} g(c, s, e) = f_{start}(s, c; \theta) + f_{end}(e, c; \theta) \\ - f_{start}(s = [\text{CLS}], c; \theta) \\ - f_{end}(e = [\text{CLS}], c; \theta) \end{aligned}$$

where c is a context of 512 sub-word unit ids (including question and document tokens), $s, e \in \{0, 1, ..., 511\}$ are inclusive indices pointing to the start and end of the target answer span, θ is our model parameters, and f_{start}, f_{end} are two different outputs derived from the last layer of our model using linear projections. Following this work, the `[CLS]` token is used at training time to predict no answer instances, making $g(c, s, e)$ the log-odds of the likelihood of an answer span and the `[CLS]` span. All the contexts from each document are scored and document spans (s, e) are ranked to return the highest scoring span that does not exceed 30 tokens. We denote the highest scoring span for the generic model as g_c, and the highest scoring span for table model as g_t. In addition to the prediction score, we also calculate a confidence score κ for each span:

$$\begin{aligned} \kappa(c, s, e) = \log(\frac{\exp(f_{start}(s, c; \theta))}{\sum_{s'} \exp(f_{start}(s', c; \theta))}) \\ + \log(\frac{\exp(f_{end}(e, c; \theta))}{\sum_{e'} \exp(f_{end}(e', c; \theta))}) \end{aligned}$$

Combining Predictions

To combine the scores of the generic model g_c and the table model g_t, we use grid search on three parameters: a scaling factor α, a bias β, and a confidence threshold γ associated with the confidence of the table prediction κ_t:

$$g = \begin{cases} \max\limits_{t,c}(g_c, \alpha \cdot g_t + \beta) & \text{if } g_c^l \in \text{t}, \kappa_t > \gamma \\ g_c & \text{otherwise} \end{cases}$$

where $g_c^l \in \text{t}$ indicates whether the long answer span that is predicted by the generic model points to a table. The search for parameters α, β, and γ is done on a validation set.

5.4 Experiments

We evaluate our model on the Natural Questions (NQ) corpus [42] that consists of questions and paired Wikipedia pages, with the task of finding the exact location of the answer that is present in the article, if any. The main benefits of using the NQ dataset are its size (300k training samples) and the general nature of the dataset - answers (if present) can appear in either or both text and tables. NQ contains one human annotation for every question in the training set, and 5 annotations for every question in the development and test sets. Based on the statistics of the development set, around 14% of questions contain a short answer in a table. Only 48% of the questions have a long answer annotation (a paragraph or a table that contains an answer), while only 33% contain a short answer annotation (an exact location of a short answer phrase). In this work we first evaluate our models on a subset of questions that contain at least one short answer in tables, referred as NQTables, and then further evaluate the model on the full dataset, referred as FullNQ.

For the questions in NQTables, we further evaluate in two settings: (*i*) $\text{NQTables}_{\text{Tab}}$, where systems are limited to predict answer spans only from tables, and (*ii*) NQTables, where systems can predict answer spans from both text and tables. Note that although all questions in NQTables have at least one answer in a table, they might also have answers in text, and systems operating in the full NQTables setting have more chances to arrive at a correct answer than systems in $\text{NQTables}_{\text{Tab}}$ setting. While our primary focus in this work on improving the short answer prediction, we also report the long answer prediction results for our best model used for the full NQ dataset, according to official metrics.

The development set (Dev) for NQTables used in this work contains 1118 questions where the short answer can be found in a table. Since the official test set of the corpus is not public, all our experiments use the official development set as our test set, while splitting the training set into training (90%) and 2 validation sets (each contains 4% of the data). The first validation set (Val-1) is used for tuning the parameters of the table-based models, while the second validation set (Val-2) is used for tuning the parameters to combine text-based

and table-based models. For clarity, in all the experiment variations we use the notation defined above where NQTables$_{\text{Tab}}$-Dev/Val-1/Val-2 is limited to predict answer spans only from tables, and NQTables-Dev/Val-1/Val-2 is capable of predict answer spans from both text and tables.

The official evaluation script computes F1 scores and considers any questions that have at least 2 annotated answers as being answerable, while questions with 1 or no answer are unanswerable. An F1 score is calculated on the ability to predict a span that matches at least one of the annotations for the answerable questions, and to correctly predict unanswerable ones. Since the validation sets contain only a single annotation, the F1 score measure based on 5-way annotation cannot be used directly on VAL-1 or VAL-2. Therefore, in the table-based experiments on the VAL-1, we report both accuracy (percent of correctly predicted answers) and modified F1 score (F1*), where modified F1 is based on comparing predictions to a single annotation. Finally, we report a string-based F1 score that accepts any exact string match of a predicted span to an answer when combining text and table models.

5.4.1 Table-based Results

First, we evaluate the proposed table representation approaches on NQTables subset.In this set of experiments we use the article's tables as our input, omitting direct usage of the article's text except in the form of context-aware updates of the model used in Section 5.3.2. The results on NQTables$_{\text{Tab}}$-Val$_1$ and NQTables$_{\text{Tab}}$-Dev are presented in Table 5.1. First, we evaluate our model using a RoBERTa baseline, following Alberti *et al.* [1] with RoBERTa pretrained model instead of BERT (line 1). This baseline is trained using the FullNQ corpus and contains both text and table inputs. We improve this baseline by finetuning on the NQTables part of the training dataset (line 2). Previous work on table representation using BERT has shown improvement from using `[SEP]` tokens to highlight cell boundaries [37]. This is the additional baseline we report in line 3. Since this baseline performs well, we combine the usage of `[SEP]` tokens with our models for the rest of the table-encoding experiments (lines 3-7). We evaluate our table representation in lines 4-6,

Model	Pretrained on WikiTables	Finetuned on	Val-1 F1*	Val-1 Acc	Dev F1
RoBERTa baseline	no	FullNQ	49.1	48.2	53.3
RoBERTa tables	no	tables only	52.5	51.2	56.8
`[SEP]` encoding	no	tables only	53.2	51.1	57.5
GNN	no	tables only	52.6	51.2	55.0
GNN	yes	tables only	54.4	52.9	56.7
GNN	yes	FullNQ $\rightarrow$ tables only	55.7	54.3	58.9
Context-aware + GNN	yes	FullNQ $\rightarrow$ tables only	56.7	55.4	60.0

Table 5.1: Evaluation of the table encodings and context-aware table representation on $\text{NQTables}_{\text{Tab}}$-$\text{Val}_1$ (Val-1) and $\text{NQTables}_{\text{Tab}}$-Dev (Dev) sets of NQTables that contains tables input exclusively.

where results in lines 5 and 6 are obtained by initially pretraining the additional weights introduced by the GNN using WikiTables, as described in Section 5.3.1. We also found that by finetuning on the full NQ dataset first and further finetuning on the NQTables subset, the results are substantially improved (lines 6 and 7). Finally, our context-aware model combined with the table encoding from line 6 achieves the best result in this set of experiments. The improvements of the GNN and textual context-aware models are statistically significant with $p < 0.01$ according to a Wilcoxon signed-rank test.

5.4.2 Combining Tables and Text

In this section, we describe experiments of combining text and table predictions. Since the official development set contains up to 5 annotations, some of those annotations can be associated with tables, while others are associated with text. Unlike in the previous section, where text-related answers are ignored, here we allow a match to either text- or table-based answers. For this purpose, we need to combine scores from the text-based and table-based models. For text predictions, we use the RoBERTa baseline that was trained on the FullNQ

	NQTables (span)			FullNQ (span)			FullNQ (str)		
Model	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**
Alberti *et al.* [1]	58.9	63.6	54.8	52.4	57.6	48.0	54.0	59.3	49.5
RoBERTa	62.5	65.0	60.2	54.4	60.3	49.6	56.2	62.3	51.2
GNN	63.1	64.6	61.7	**54.6**	60.2	50.0	**56.6**	62.3	51.8
Context-aware + GNN	**64.0**	65.5	62.6	**54.6**	59.4	50.6	**56.6**	61.5	52.4

Table 5.2: Results of combining text-based and table-based predictions from two models: F1, Precision, and Recall scores reported on NQTables-Dev and FullNQ-Dev. (span) and (str) indicate span-based and string-based scores.

dataset; for the table predictions we use our best GNN model and the context-aware model (lines 6 and 7 in Table 5.1, respectively). Oracle analysis on the development set suggests that a linear score transformation in the case of the FullNQ is not effective. To combine the scores of the text model g_c and the table model g_t, we use grid search, as described in Section 5.3.3. The search for parameters α, β, and γ is done on Val-2 containing questions with table answers, questions with text answers, and questions without answers, using string-based F1 score instead of span-based one to compensate for the lack of multiple annotations in the validation set. According to this metric, both answers in tables and text are considered correct if they have sufficient string overlap with the gold answer span in a table annotated in Val-2.

We evaluate those models on both the NQTables-Dev and FullNQ-Dev, allowing both text and table answers. The results are presented in Table 5.2. Results from our RoBERTa baseline that was trained on FullNQ are shown in line 1. Then, as mentioned above, we combine this generic model with table-only models (lines 2 and 3). Our experiments suggest that our proposed best model that uses table encoding improves the F1 score on NQTables by 1.5 points. When the table+text model combination is used on the full NQ dataset, there is a small improvement from both table models, but the textual context attention model

Sampling (pos : neg_within : neg_outside)	**NQTables Val-1 (F1)**	**FullNQ Dev (F1)**	γ
(1) Equal sampling within positive article (1:1:0)	55.7	54.6	-0.005
(2) Random sampling across all articles (0.63:0.28:0.72)	52.5	55.1	-3
(3) Equal sampling within and across articles (2:1:1)	54.0	54.9	-0.5

Table 5.3: The effect of negative sampling technique on table-only models and text+table model combinations.

is comparable to the GNN model, increasing recall at the cost of reducing precision. This might be explained by the relatively small fraction of questions that have answers only in tables (8% of all questions) and difficulty in calibrating answers from tables and text against each other. The next section explores the impact of negative sampling on the performance of table-based models and their combination with text-based models.

Impact of Negative Sampling

In all our experiments using models finetuned on tables only data (lines 2-7 in Table 5.1 and lines 2-3 in Table 5.2), during training we used an equal proportion of positive and negative samples,[3] where all negative samples were taken from the articles in NQTables-Train, which contain an answer in a table. This approach is successful when models were evaluated on their ability to predict answers in tables, for articles that are known to contain such answers. This was done based on our initial experimentation that showed the benefit of such approach for table-based experiments (Section 5.4.1) comparing to random sampling of negatives across all the articles, where the ratio of negatives-to-positives is 0.63, as used in Alberti *et al.* [1]. On the other hand, when the table-based models are asked to make predictions for articles not known to contain answers in tables (or any answers), they may be over-confident in comparison to a generic text-based model, trained with negative examples

[3]A sample is a table or a table fragment, together with an indication of short answer span or NULL.

across all articles. This over-confidence is evident from the extremely high selected confidence parameters of $\gamma = -0.005$ for GNN model (line 2 in Table 5.2) and $\gamma = -0.0025$ for context-aware model (line 3 in Table 5.2), suggesting that only high-value and high-confidence scores are considered from the table-based model.

To investigate the effect of the negative sampling method we perform an ablation study that compares three techniques: 1) sampling negative samples from within articles that contain an answer in tables, 2) random sampling of negatives across all NQ articles, in the proportion used by Alberti *et al.* [1], and 3) sampling negatives with equal proportion from articles that contain answer in tables, and articles that do not. The results for the GNN model are shown in Table 5.3. While the first sampling strategy works best when a table-based model is used to predict answers from tables in NQTables, sampling negatives from a more diverse set of articles improves the overall FullNQ results for the combination of text and table-based models, by allowing better calibration between text and table models. The threshold values γ are seen to be much lower in these cases. However, the second and third strategies reduce performance on QA from tables. When optimizing the random sampling strategy for highest performance on FullNQ, no benefit was found from contextual text attention for table representations.

Finally, we compare the performance of our model to other work on the Natural Questions on both the short and long answer prediction tasks.In our model, the long answer is predicted based on the segment that corresponds to the short answer prediction. The results are presented in Table 5.4. As we can see, our method obtains a substantial improvement over the baseline of alberti2019bert in short answer F1, and a smaller improvement in long answer F1. Advances in long answer F1 from state-of-the-art recent works are likely complementary to our method and can be integrated for additional gain.

5.5 Conclusion

Tables in web documents are pervasive and can be directly used to answer many search queries. In this work, we presented an approach to enrich table representations using infor-

Model	Short F1	Long F1
Alberti *et al.* [1]	52.7	64.7
Liu *et al.* [50]	57.7	73.9
Ravula *et al.*[78]	58.5	78.2
GNN, random sample (2)	55.1	65.9

Table 5.4: Comparison to other NQ models.

mation from article text, and showed that it improves a state-of-the-art pretrained structure-aware table representation for question answering from tables. We also studied how to effectively combine text-based and table-based approaches. Finally, we performed the first study focusing on table QA for the Natural Questions dataset, and showed that improved representations of tables increase performance. In future work, our methods can be applied to other QA datasets, such as WikiTableQuestions [69] and HybridQA [16].

Chapter 6

CONCLUSIONS

In this chapter, we conclude the thesis by summarizing the main contributions of the dissertation, including the methods developed and experimental findings. Further, we discuss the future directions of research and broader impact of the methods developed in this thesis in terms of potential applications.

6.1 Summary

This dissertation explores ways in which observable structure and language can be combined in a variety of scenarios. In particular, we study how structure can be incorporated in three different tasks, with each of the tasks having very different structural components requiring unique structure representation techniques. Below are the main contributions of this dissertation.

Conversation Structure in Popularity Prediction. Modeling discussions on social media platforms can benefit from accounting for the structure of the conversation (reply history). In this work, we present a novel approach for modeling threaded discussions on social media using a graph-structured bidirectional LSTM which represents both hierarchical and temporal conversation components. The propagation of hidden state information in the graph provides a mechanism for representing contextual language, including the history that a comment is responding to as well as the ensuing discussion it spawns. Experiments on Reddit discussions show that the graph-structured LSTM leads to improved results in predicting comment popularity compared to a node-independent model. Analyses show that the model benefits prediction over the extent of the discussion, and that language cues are particularly important for distinguishing controversial comments from those that are very

positively received. Responses from even a small number of comments seem to be useful, so it is likely that the bidirectional model would still be useful with a short-time lookahead for early prediction of popularity. This work represents the first successful use of text in predicting popularity without highly constrained matched context data.

Phrase Structure in Disfluency Detection. Disfluencies are irregularities that occur in spontaneous speech, thus leading to a unique patterns in disfluent phrase structure. In this dissertation we represent a phrase structure for disfluency detection from two perspectives. First, we represent a parallel structure between the reparandum and repair by introducing a novel neural network architecture which allows automatic discovery of patterns in disfluencies and directly uses similarity scores as input features to a CNN. We show that our approach can be as effective as using carefully designed, hand-engineered pattern match features in a disfluency detection task, eliminating the need for feature engineering. More importantly, we show that the pattern match network is robust in cross-domain testing, unlike the engineered features. Secondly, we incorporate prosody that used as an approximation of another (acoustic) view of the phrase structure in order to improve disfluency detection. In addition to disfluency-related signal, acoustic cues contain other sources of information. Due to high variability in acoustic correlates, using simple multimodal fusion approaches do not perform well. In order to address this problem we propose a novel approach that reduces irrelevant signal in the prosody observation making the prosodic information complementary to the one found in the textual component. This work represents the first successful use of prosody in disfluency detection since the development of neural disfluency detection approaches.

Question Answering on Tables. Question Answering (QA) on tables is an important and fast-growing branch of reading comprehension. While most of the work on QA on Tables considers the tables as the only input to the system, very often tables appear within a textual context. In this work, we are interested in the combination of text-based and structured resources for question answering, particularly articles that contain both tables and text. The main contribution of this work is two-fold. First, we propose a novel approach to represent a table using modified BERT architecture that make a use of pertained BERT

model. Secondly, we propose a new approach that allows to propagate information from the article surrounding the table to the relevant parts of the table, thus improving the limited context of table entries and further updating table representation.

6.2 Future Directions

Structural components are very common in language-related tasks. In this section I will present how the novel methods introduced in this dissertation can be incorporated in other work in future, giving a better perspective on the potential for broader impact of this dissertation.

The Graph-LSTM architecture introduced in Chapter 3 can be used in a variety of tasks on social media, where it can be incorporated to represent the conversational structure. This makes it especially easy to incorporate for other social media platforms, such as Facebook, Twitter, or Quora, where the conversation structure is very similar to a reply structure on Reddit (original post with multiple replies/retweets at each level). While our approach is evaluated on popularity prediction task, incorporating conversational structure can be beneficial for a lot of other tasks on social media, including sentiment analysis, summarization, or generation.

Our pattern-match network approach (Chapter 4) makes it possible to find repeating patterns in the language. While this architecture was designed for the disfluency detection task for capturing reparandum-repair structure, this approach can be used in order to capture other types of similarity across structure and can be used in the tasks such as paraphrase detection, multi-hop reading comprehension, or multi-document summarization. While the idea of the similarity learning in our approach is very similar to the self-attention in a transformer (just a shallow version of it), additional convolutional layers on top facilitates attending to multi-word patterns. Incorporating a similar type of multi-word self-attention can be an interesting next step in obtaining word representations for transformer. In the future, following [26], we are interested in exploring domain adaptation techniques in disfluency detection.

Innovation features that are also introduced in Chapter 4 represent a novel method that was developed for prosody representation. While the approach is evaluated on disfluency detection task, it is general enough to be used for prosody representation on a variety of tasks, with the main requirement of availability of aligned audio and transcript data. Thus, this approach can be used for other tasks on spoken language, including dialog act modeling, emotion detection, or natural language understanding. In addition, this method can be easily adapted for other multi-modal scenarios with highly overlapping information coming from two modalities. The approach is especially well suited for multi-view scenarios such as lip reading [19], sentiment analysis, or emotion detection tasks [116].

Table encoding introduced in Chapter 5 can be applied to other questions answering datasets that contain tables, but also can be used in other tasks that contain tables. In addition, our novel approach of table representation updates based on the text context can be used in other tasks that contain two types of information, such as question answering on knowledge bases.

BIBLIOGRAPHY

[1] Chris Alberti, Kenton Lee, and Michael Collins. A BERT baseline for the natural questions. *arXiv preprint arXiv:1901.08634*, 2019.

[2] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *Proc. ICML*, 2013.

[3] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Proc. NIPS*, 2016.

[4] Nguyen Bach and Fei Huang. Noisy BiLSTM-based models for disfluency detection. In *Proc. Interspeech*, 2019.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, 2015.

[6] Roja Bandari, Sitaram Asur, and Bernardo Huberman. The pulse of news in social media: Forecasting popularity. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*, 2012.

[7] Don Baron, Elizabeth Shriberg, and Andreas Stolcke. Automatic punctuation and disfluency detection in multi-party meetings using prosodic and lexical cues. In *Seventh International Conference on Spoken Language Processing*, 2002.

[8] Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. Graph convolutional encoders for syntax-aware neural machine translation. In *Proc. EMNLP*, 2017.

[9] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Methods for exploring and mining tables on Wikipedia. In *Proc. of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, 2013.

[10] Bin Bi and Junghoo Cho. Modeling a retweet network via an adaptive bayesian approach. In *Proc. WWW*, 2016.

[11] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *Proc. ICLP*, 2014.

[12] E. Charniak and M. Johnson. Edit detection and parsing for transcribed speech. In *Proc. NAACL*, 2001.

[13] Bo Chen, Le Sun, and Xianpei Han. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *Proc. ACL*, 2018.

[14] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: fast learning with graph convolutional networks via importance sampling. In *Proc. ICLR*, 2018.

[15] Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *Proc. ACL*, 2019.

[16] Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*, 2020.

[17] Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. Can cascades be predicted? In *Proc. WWW*, 2014.

[18] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. EMNLP*, 2015.

[19] Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman. Lip reading sentences in the wild. In *Proc. CVPR*. IEEE, 2017.

[20] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[21] Neeraj Deshmukh, Aravind Ganapathiraju, Andi Gleeson, Jonathan Hamaker, and Joseph Picone. Resegmentation of Switchboard. In *Fifth international conference on spoken language processing*, 1998.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL*, 2019.

[23] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. NAACL*, 2015.

[24] Hao Fang, Hao Cheng, and Mari Ostendorf. Learning latent local conversation modes for predicting community endorsement in online discussions. In *Proc. Inter. Workshop on NLP for Social Media*, 2016.

[25] James Ferguson, Greg Durrett, and Dan Klein. Disfluency detection with a semi-markov model and prosodic features. In *Proc. NAACL-HLT*, 2015.

[26] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proc. ICML*, 2015.

[27] Kallirroi Georgila. Using integer linear programming for detecting speech disfluencies. In *Proc. NAACL HLT*, 2009.

[28] J. J. Godfrey, E. C. Holliman, and J. McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. In *Proc. ICASSP*, volume I, 1992.

[29] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proc. NIPS*, 2017.

[30] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12), 2004.

[31] Ji He, Mari Ostendorf, Xiaodong He, Jianshu Chen, Lihong Gao, Jianfeng andLi, and Li Deng. Deep reinforcement learning with a combinatorial action space for predicting popular Reddit threads. In *Proc. EMNLP*, 2016.

[32] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[33] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In *Proc. ACL*, 2020.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), 1997.

[35] Matthew Honnibal and Mark Johnson. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2(1), 2014.

[36] Julian Hough and David Schlangen. Recurrent neural networks for incremental disfluency detection. In *Proc. Interspeech*, 2015.

[37] Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. A comprehensive exploration on WikiSQL with table-aware word contextualization. In *Proc. of Knowledge Representation Reasoning Meets Machine Learning Workshop*, 2019.

[38] Aaron Jaech, Victoria Zayats, Hao Fang, Mari Ostendorf, and Hannaneh Hajishirzi. Talking to the crowd: What do people react to in online discussions? In *Proc. EMNLP*, 2015.

[39] Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. A nested attention neural hybrid model for grammatical error correction. In *Proc. ACL*, 2017.

[40] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. ICLR*, 2017.

[41] Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. Neural semantic parsing with type constraints for semi-structured tables. In *Proc. EMNLP*, 2017.

[42] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[43] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, 2001.

[44] Himabindu Lakkaraju, Julian J McAuley, and Jure Leskovec. What's in a name? understanding the interplay between titles, content, and communities in social media. In *Proc. ICWSM*, 2013.

[45] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proc. ICLR*, 2020.

[46] Phong Le and Willem Zuidema. Compositional distributional semantics with long short term memory. In *Proc. Joint Conf. on Lexical and Computational Semantics*, 2015.

[47] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proc. ACL*, 2014.

[48] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proc. AAAI*, 2018.

[49] Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. Hierarchical recurrent neural network for document modeling. In *Proc. EMNLP*, 2015.

[50] Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. Rikinet: Reading wikipedia pages for natural question answering. *arXiv preprint arXiv:2004.14560*, 2020.

[51] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, and M. Harper. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Trans. Audio, Speech and Language Processing*, 14, 2006.

[52] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized bert pretraining approach. In *Proc. ICLR*, 2020.

[53] Paria Jamshid Lou, Peter Anderson, and Mark Johnson. Disfluency detection using auto-correlational neural networks. In *Proc. EMNLP*, 2018.

[54] Paria Jamshid Lou and Mark Johnson. Disfluency detection using a noisy channel model and a deep neural language model. In *Proc. ACL*, volume 2, 2017.

[55] Paria Jamshid Lou and Mark Johnson. Improving disfluency detection by self-training a self-attentive model. *arXiv preprint arXiv:2004.05323*, 2020.

[56] Lin Ma, Zhengdong Lu, Lifeng Shang, and Hang Li. Multimodal convolutional neural networks for matching image and sentence. In *Proc. of the IEEE international conference on computer vision*, 2015.

[57] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. ACL*, 2016.

[58] Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. Document-level neural machine translation with hierarchical attention networks. In *Proc. EMNLP*, 2018.

[59] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proc. ICLR*, 2013.

[60] Yasuhide Miura, Ryuji Kano, Motoki Taniguchi, Tomoki Taniguchi, Shotaro Misawa, and Tomoko Ohkuma. Integrating tree structures and graph structures with neural networks to classify discussion discourse acts. In *Proc. COLING*.

[61] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proc. ACL*, 2016.

[62] Thomas Müller, Francesco Piccinno, Massimo Nicosia, Peter Shaw, and Yasemin Altun. Answering conversational questions on structured data without logical forms. In *Proc. EMNLP*, 2019.

[63] Farah Nadeem and Mari Ostendorf. Estimating linguistic complexity for science texts. In *Proc. of the thirteenth workshop on innovative use of NLP for building educational applications*, 2018.

[64] C. Nakatani and J. Hirschberg. A corpus-based study of repair cues in spontaneous speech. *Journal of the Acoustical Society of America*, 1994.

[65] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proc. ICML*, 2011.

[66] Thien Huu Nguyen and Ralph Grishman. Graph convolutional networks with argument-aware pooling for event detection. In *Proc. AAAI*, 2018.

[67] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016.

[68] Mari Ostendorf and Sangyun Hahn. A sequential repetition model for improved disfluency detection. In *Proc. Interspeech*, 2013.

[69] P. Pasupat and P. Liang. Compositional semantic parsing on semi-structured tables. In *Proc. ACL*, 2015.

[70] Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5, 2017.

[71] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proc. EMNLP*, 2014.

[72] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. NAACL*, 2018.

[73] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The Kaldi Speech Recognition Toolkit. In *Proc. ASRU*, 2011.

[74] Xian Qian and Yang Liu. Disuency detection using multi-step stacked learning. In *Proc. NAACL-HLT*, 2013.

[75] Yujie Qian, Enrico Santus, Zhijing Jin, Jiang Guo, and Regina Barzilay. Graphie: A graph-based framework for information extraction. In *Proc. NAACL*, 2019.

[76] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *cdn.openai.com/research-covers/language-unsupervised/language$_u$nder standing$_p$aper.pdf*, 2018.

[77] Mohammad Sadegh Rasooli and Joel R Tetreault. Joint parsing and disfluency detection in linear time. In *Proc. EMNLP*, 2013.

[78] Anirudh Ravula, Chris Alberti, Joshua Ainslie, Li Yang, Philip Minh Pham, Qifan Wang, Santiago Ontanon, Sumit Kumar Sanghai, Vaclav Cvicek, and Zach Fisher. Etc: Encoding long and structured inputs in transformers. 2020.

[79] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *Proc. European Semantic Web Conference*. Springer, 2018.

[80] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proc. AAAI*, 2016.

[81] Peter Shaw, Philip Massey, Angelica Chen, Francesco Piccinno, and Yasemin Altun. Generating logical forms from graph representations of text and entities. In *Proc. ACL*, 2019.

[82] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proc. NAACL*, 2018.

[83] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for RNN/CNN-free language understanding. In *Proc. AAAI*, 2018.

[84] E. Shriberg. *Preliminaries to a theory of speech disfluencies*. PhD thesis, Department of Psychology, University of California, Berkeley, CA, 1994.

[85] E. Shriberg. Phonetic consequences of speech disfluency. In *Proc. International conference of Phonetics Sciences*, 1999.

[86] E. Shriberg and A. Stolcke. A prosody-only decision-tree model for disfluency detection. In *Proc. Eurospeech*, 1997.

[87] Matthew Snover, Bonnie Dorr, and Richard Schwartz. A lexically-driven algorithm for disfluency detection. In *Proc. HLT-NAACL*, 2004.

[88] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Proc. NIPS*, 2015.

[89] Kihyuk Sohn, Wenling Shang, and Honglak Lee. Improved multimodal deep learning with variation of information. In *Proc. NIPS*, 2014.

[90] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Proc. NIPS*, 2012.

[91] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proc. EMNLP*, 2018.

[92] Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Joint multimodal learning with deep generative models. In *Proc. ICLR*, 2017.

[93] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. ACL-IJCNLP*, 2015.

[94] Chenhao Tan, Lillian Lee, and Bo Pang. The effect of wording on message propagation: Topic-and author-controlled natural experiments on twitter. In *Proc. ACL*, 2014.

[95] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[96] Dinh V Tran, Nicolò Navarin, and Alessandro Sperduti. On filter size in graph convolutional networks. In *Proc. of 2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018.

[97] Quan Hung Tran, Ingrid Zukerman, and Gholamreza Haffari. A hierarchical neural model for learning sequences of dialogue acts. In *Proc. EACL*, 2017.

[98] Trang Tran and Mari Ostendorf. Characterizing the language of online communities and its relation to community recognition. In *Proc. EMNLP*, 2016.

[99] Trang Tran, Shubham Toshniwal, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Mari Ostendorf. Parsing speech: a neural approach to integrating lexical and acoustic-prosodic information. In *Proc. NAACL*, 2018.

[100] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. NIPS*, 2017.

[101] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *Proc. ICLR*, 2018.

[102] Feng Wang, Wei Chen, Zhen Yang, Qianqian Dong, Shuang Xu, and Bo Xu. Semi-supervised disfluency detection. In *Proc. COLING*, 2018.

[103] Shaolei Wang, Wanxiang Che, and Ting Liu. A neural attention model for disfluency detection. In *Proc. COLING*, 2016.

[104] Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. Transition-based disfluency detection using LSTMs. In *Proc. EMNLP*, 2017.

[105] Weiran Wang, Xinchen Yan, Honglak Lee, and Karen Livescu. Deep variational canonical correlation analysis. *arXiv preprint arXiv:1610.03454*, 2016.

[106] Tim Weninger, Xihao Avi Zhu, and Jiawei Han. An exploration of discussion threads in social news sites: A case study of the Reddit community. In *Proc. IEEE/ACM Inter. Conf. on Advances in Social Networks Analysis and Mining*, 2013.

[107] Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. Efficient disfluency detection with transition-based parsing. In *Proc. ACL-IJCNLP, year=2015.*

[108] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[109] Yunxuan Xiao, Yanru Qu, Lin Qiu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. Dynamically fused graph network for multi-hop reasoning. In *Proc. ACL*, 2019.

[110] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[111] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proc. NIPS*, 2019.

[112] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proc. NAACL-HLT*, 2016.

[113] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*, 2020.

[114] Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Tensor fusion network for multimodal sentiment analysis. In *Proc. EMNLP*, 2017.

[115] Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. Multi-attention recurrent network for human communication comprehension. In *Proc. AAAI*, 2018.

[116] AmirAli Bagher Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Multimodal language analysis in the wild: CMU-MOSEI dataset and interpretable dynamic fusion graph. In *Proc. ACL*, 2018.

[117] Vicky Zayats and Mari Ostendorf. Robust cross-domain disfluency detection with pattern match networks. *arXiv preprint arXiv:1811.07236*, 2018.

[118] Vicky Zayats and Mari Ostendorf. Giving attention to the unexpected: using prosody innovations in disfluency detection. In *Proc. NAACL*, 2019.

[119] Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. Disfluency detection using a bidirectional LSTM. In *Proc. Interspeech*, 2016.

[120] Victoria Zayats and Mari Ostendorf. Conversation modeling on reddit using a graph-structured LSTM. *Transactions of the Association for Computational Linguistics*, 6, 2018.

[121] Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. Multidomain disfluency and repair detection. In *Proc. Interspeech*, 2014.

[122] Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. Unediting: Detecting disfluencies without careful transcripts. In *Proc. NAACL HLT*, 2015.

[123] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[124] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *Proc. AUAI*, 2018.

[125] Xingxing Zhang, Liang Lu, and Mirella Lapata. Top-down tree long short-term memory networks. In *Proc. NAACL-HLT*, 2016.

[126] Xingxing Zhang, Furu Wei, and Ming Zhou. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proc. ACL*, 2019.

[127] Yue Zhang, Qi Liu, and Linfeng Song. Sentence-state LSTM for text representation. In *Proc. ACL*, 2018.

[128] Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. Seismic: A self-exciting point process model for predicting tweet popularity. In *Proc. SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[129] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[130] Simon Zwarts, Mark Johnson, and Robert Dale. Detecting speech repairs incrementally using a noisy channel approach. In *Proc. COLING*, 2010.