Neural Attentions for Natural Language Understanding and Modeling

by

Hongyin Luo

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author .....
Department of Electrical Engineering and Computer Science
May 23, 2019

Certified by .....
James Glass
Senior Research Scientist, CSAIL
Thesis Supervisor

Accepted by .....
Leslie Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Neural Attentions for Natural Language Understanding and Modeling

by

Hongyin Luo

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

In this thesis, we explore the use of neural attention mechanisms for improving natural language representation learning, a fundamental concept for modern natural language processing. With the proposed attention algorithms, our model made significant improvements in both language modeling and natural language understanding tasks.

We regard language modeling as a representation learning task that learns to align local word contexts and their following words. We explore the use of attention mechanisms for both the context and following words to improve the performance of language models, and measure perplexity improvements on classic language modeling tasks. To learn better representation of contexts, we use a self-attention mechanism with a convolutional neural network (CNN) to simulate long short-term memory networks (LSTMs). The model process sequential data in parallel and still achieves competitive performances. We also propose a phrase induction model and headword attention to learn the embedding of following phrases. The model is able to learn reasonable phrase segments and outperforms several state-of-the-art language models on different data sets. The approach outperformed AWD-LSTM model by reducing 2 perplexities on the Penn Treebank and Wikitext-2 data sets, and achieved new state-of-the-art performance on the Wikitext-103 data set with 17.4 perplexity.

For language understanding tasks, we propose the use of a self-attention CNN for video question answering. The performance of this model is significantly higher than the baseline video retrieval engine. Finally, we also investigate an end-to-end co-reference resolution model by applying cross-sentence attentions to utilize knowledge in contextual data and learn better contextualized word and span embeddings. The model achieved 66.69% MAP@1, and 87.42% MAP@5 accuracy of video retrieval and 57.13% MAP@1, 80.75 MAP@5 accuracy of a moment detection task, significantly outperforming the baselines.

Thesis Supervisor: James Glass
Title: Senior Research Scientist, CSAIL

Acknowledgments

This thesis is completed with kind guidance and assistance of my research advisor, Dr. James Glass. His patience and kindness helped me made through all the difficulties.

The study is partly supported by Ford Motor Company. They provided real-life, industrial data for us to conduct exciting research projects.

I would also send my sincere thanks to my parents who are always supportive, and my friends that kept encouraging me. The thesis could not be completed with their help.

Thanks to Marcia for hosting treats for the group and making sure that the water mark on the thesis is towards the correct direction!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, neural networks have been frequently applied in NLP tasks and led to significant improvements. Neural language models [Bengio et al., 2003] represent words with low-dimensional, distributed word vectors. The term "distributed vector" stands for a vector whose dimensions represent semantics jointly and a single dimension is not interpretable. Distributed word embedding has become a popular method for many NLP tasks and achieved state-of-the-art performances. Recently, distributed embedding algorithms have been proposed for representing basic language units, including characters [Zhang et al., 2015] and words [Mikolov et al., 2013, Pennington et al., 2014]. These models are trained based on task-oriented objective functions or by regressing semantic similarities. Outputs of the trained representation models, which are referred to as pre-trained character/word embeddings, are often used as the inputs to other NLP models.

Besides characters and words, many NLP tasks require higher level language representations, such as phrase and sentence embeddings. To solve this problem, several models have been proposed for phrases [Lee et al., 2016] or sentences [Bahdanau et al., 2014, Chung et al., 2014]. Variants of recurrent neural networks (RNNs), for example long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] and gated recurrent units (GRU) [Chung et al., 2014] are applied for sequence representation tasks. The networks take sequences of word vectors as inputs, embed the sequences with distributed vectors. The sequence embeddings can be applied in many ap-

plications including co-reference resolution [Lee et al., 2017], machine translation [Bahdanau et al., 2014], and language modeling [Merity et al., 2017].

Although LSTMs and GRUs have led to significant improvements in sequence representations learning, the attention mechanism [Bahdanau et al., 2014, Xu et al., 2015a] can further push the performances of sequence embedding. RNNs model sentences following a sequential order, while natural sentences often contains non-sequential structures and dependencies. Attention mechanisms enable RNNs to consider skip-word dependencies and highlights the most important knowledge for learning best word and sequence embeddings. More specifically, the attention mechanism can help with two aspects. First, it improves the learning of contextual word representations. The quality of contextual word representations influences the performances of many NLP tasks. The authors of [Peters et al., 2018] proposed a deep highway LSTM architecture to learn contextualized word representations, which improved the performance of several tasks including co-reference and reading comprehension. The model has a stronger ability to memorize input sequences and model contexts, but has not shown an ability to model structural and syntactic knowledge in sentences.

Secondly, the attention mechanism can improve the quality of sequence embeddings, including sentences and contexts. [Shen et al., 2017] proposed syntactic distance and attention for language modeling, which enables the model to learn the parsing tree of the input sentences without any supervision. This method significantly improved the performance of language modeling. [Bahdanau et al., 2014] showed that the attention mechanism can summarize the most informative context for generating the next words in the sentence of the target language.

In this work, we apply and extend the attention mechanism on different NLP tasks where the models learn different level of language representations. We applied cross-sentence attentions to improve the end-to-end co-reference resolution model. We showed that we can build a parallel sequence embedding model that calculates relations among words with self-attentions. We also applied self-attention in a video question answering model and showed that the proposed self-attention convolutional neural network (SA-CNN) improves the quality of question and answer embedding. We

also applied attention mechanism in phrase-level predictions for improving language models.

Chapter 2

Background

2.1 Word Representation Learning

Words are represented as strings in natural languages. However, computers cannot directly conduct computation with strings. As a result, strings are often converted into discrete symbols. However, this approach is not effective enough to represent the complexity of natural languages, since both syntax and semantics are hierarchical, highly correlated, and usually have various and complex combinations. To overcome this difficulty, researchers have proposed to represent words as vectors. In this section, we introduce different models for word vectorization.

2.1.1 One-hot Representation

The most direct and intuitive model for word vectorization is representing words as one-hot vectors. One-hot vector means a vector has only one element assigned with 1, and 0 for all other elements. Given a vocabulary with N unique words, we construct a N dimensional vector space for word representation. The representation of the i-th word in the vocabulary,

$$w_i = [0, \ldots, 0, 1, 0, \ldots, 0] \tag{2.1}$$

The i-th element of w_i is assigned with 1, and all other elements are 0s. Words can be processed by the mathematical models with such representations. However, the

one-hot word representations are not capable to capture the relations among different words. For example, neither computer nor human can tell the semantic and syntactic similarities among different words only with one-hot vectors, since the Euclidean distances between all word pairs are the same.

$$d_{i,j} = \sqrt{1^2 + 1^2} = \sqrt{2} \tag{2.2}$$

2.1.2 Distributed Word Embedding

A good word representation model should be able to capture the semantic similarity and syntactic relation between two words. In this section, we first describe the core of distributed word embeddings - the matrix factorization algorithm, and then introduce two popular frameworks based on this method.

Pointwise Mutual Information Matrix

Before introducing distributed word representation learning, we introduce the pairwise mutual information (PMI) matrix. Consider a corpus with n unique words $w_1, w_2, \ldots, w_n$. We construct a matrix A, where elements are

$$a_{ij} = \log \frac{P(w_i, w_j)}{P(w_i) \cdot P(w_j)} \tag{2.3}$$

where $P(w)$ stands for the frequency that word w appears in the corpus. Say $\#(w)$ is defined as the number of word w and $|D|$ is the total number of words in the corpus, we calculate $P(w)$ with

$$P(w) = \frac{\#(w)}{|D|} \tag{2.4}$$

$P(w_i, w_j)$ stands for the probability that w_i and w_j appears in the same context window. Let $\#(w_i, w_j)$ stands for the number of cases that w_i and w_j appears in a context window,

$$P(w_i, w_j) = \frac{\#(w_i, w_j)}{|D|} \tag{2.5}$$

Matrix Factorization for Distributed Word Embedding

The goal of matrix factorization (MF) is finding two non-identical matrices whose matrix product can be as close as the target matrix. For example given a target matrix A, the goal of MF is to find matrices W and H, satisfying

$$A = W \cdot H \tag{2.6}$$

We use the gradient descent (GD) algorithm to find locally optimal solutions. The objective function that is to be minimized by the optimization problem is

$$l = ||A - WH||_2^2 \tag{2.7}$$

Skip-Gram Word Embeddings

In this chapter, we introduce the Skip-Gram algorithm as an example of online word representation learning. The algorithm scans each word and its context window in the entire corpus. The algorithm assign a word vector w_i and a context vector c_i for the i-th word in the vocabulary.

Consider a target word i and its context window of $2n + 1$ length, $[i - n, i - n + 1, \ldots, i, \ldots, i + n - 1, i + n]$, the Skip-Gram model updates w_i and $c_j, j \neq i$ by maximizing the following objective function,

$$l = \log[\sum_{j \in [i-n, i+n], j \neq i} \sigma(w_i \cdot c_j) + \sum_{j \in NEG_i} (1 - \sigma(w_i \cdot c_j))] \tag{2.8}$$

where $\sigma(\cdot)$ stands for the sigmoid function,

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.9}$$

and NEG_i stands for a set of negative samples that are not in the current context window of the word i. As proved in [Levy and Goldberg, 2014], the Skip-Gram model is implicitly factorizing a PMI matrix.

The learned word vectors have several advantages. Firstly, word vectors indicates word similarities. An example from [Mikolov et al., 2013] is shown in Figure 2-1. As shown in the example, words with similar word vectors generally belong to the same topic.

Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

Figure 2-1: Some word clusters learned by the distributed embedding model. [Mikolov et al., 2013]

The word embeddings also learn relations among different words. Figure 2-2 [Mikolov et al., 2013] shows the word vectors of country names and their capitals. The vectors connecting a country and its capital have roughly the same direction. This suggests that the distributed word embedding encode semantics through the geometry of the vector space.

2.2 Neural Language Models

2.2.1 Language Modeling

In this thesis, we mainly discuss about word-level language models (LMs). The goal of language is to generate text word by word. The language models learn a probabilistic distribution P of word sequences that model natural languages. Given a corpus, a

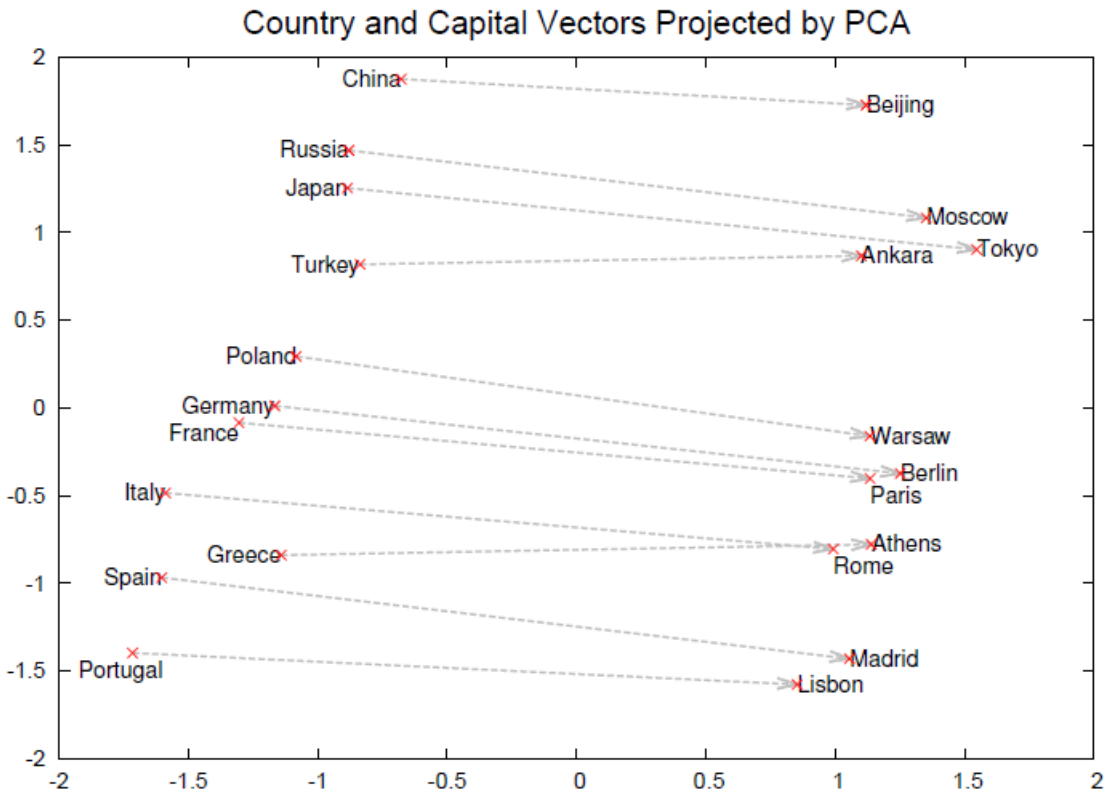


Figure 2-2: The 2-D projection of country words and corresponding capitals. The vectors from countries to capitals are roughly towards the same direction. [Mikolov et al., 2013]

language model learns to fit the probability [Bengio et al., 2003]

$$P(w_1, w_2, \ldots, w_n) = p(w_1) \prod_{i=2}^{n} p(w_i | w_1^{i-1}) \tag{2.10}$$

where w_1^{i-1} stands for a word sequence $[w_1, w_2, \ldots, w_{i-1}]$. We define it as the context of the i-th word

$$c_i = [w_1, w_2, \ldots, w_{i-1}] \tag{2.11}$$

The core of most studies in language modeling is modeling the conditional probabilistic distribution $p(w_i|c_i)$.

2.2.2 A Simple Neural Network for Language Modeling

[Bengio et al., 2003] proposed a simple neural network architecture for language modeling and achieved good performance. Given a target word w_i and its context $c_i = [w_{i-n+1}, \ldots, w_{i-2}, w_{i-1}]$, the neural networks learns $p(w_i|c_i)$ with stochastic gradient descent (SGD). Note that n is a hyper-parameter and stands for the length of a context window.

The first module of the network is a lookup layer whose parameter is a randomly initialized matrix $D \in R^{|V| \times d}$, where $|V|$ stands for the number of unique words in the vocabulary and d stands for the output size of the lookup layer. Given the index of an input word in the vocabulary, for example j, the lookup layer outputs the j-th row of the parameter matrix D.

Given c_i, the output of the lookup layer is a sequence of vectors, $[x_{i-n+1}, \ldots, x_{i-2}, x_{i-1}]$. A hidden layer is applied for calculating the context embedding h_i,

$$h_i = Tanh(W^h \cdot [x_{i-n+1}, \ldots, x_{i-2}, x_{i-1}] + b^h) \tag{2.12}$$

Then we can calculate the distribution of the next word, Y with

$$Y = [p(w_1), p(w_2), \ldots, p(w_{|V|})] = Softmax(W^y \cdot h_i + b^y) \tag{2.13}$$

where

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.14}$$

2.2.3 Recurrent Language Models

[Mikolov et al., 2010] proposed a language model with recurrent neural networks (RNNs). An RNN is designed for modeling sequential input, and natural language as a word sequence is a good fit. Given a sequence of inputs, $[x_1, x_2, \ldots, x_n]$, the dynamics of a RNN is shown as follows,

$$h_t = \sigma(W^h \cdot h_{t-1} + W^x \cdot x_t + b) \tag{2.15}$$

where $[h_1, h_2, \ldots, h_n]$ are the outputs of the RNN. In [Mikolov et al., 2010], the RNN architecture is used for predicting $p(w_i|c_i)$. Different from the model proposed in [Bengio et al., 2003], the neural network for generating a context embedding h_i is recurrent, while the calculation of Y follows in the same manner. RNNs perform better than the simple multi-layer perceptrons because they can capture sequence information.

2.2.4 Long Short-Term Memory

Previous studies have shown that RNNs suffer from gradient vanishing/explosion problems [Pascanu et al., 2013]. To solve the gradient problems caused by the RNN structure, [Hochreiter and Schmidhuber, 1997] proposed the long short-term memory (LSTM) network. Similarly to traditional RNNs, LSTM networks employ a recurrent architecture and output a sequence of vectors $[h_1, h_2, \ldots, h_n]$ given a sequential input $[x_1, x_2, \ldots, x_n]$.

A gating mechanism is applied in LSTMs to solve the gradient problems. The dynamics of LSTMs is shown as follows.

$$i_t = \sigma(W_h^i \cdot h_{t-1} + W_x^i \cdot x_t + b^i) \tag{2.16}$$

$$f_t = \sigma(W_h^f \cdot h_{t-1} + W_x^f \cdot x_t + b^f) \tag{2.17}$$

$$o_t = \sigma(W_h^o \cdot h_{t-1} + W_x^o \cdot x_t + b^o) \tag{2.18}$$

$$z_t = Tanh(W_h^z \cdot h_{t-1} + W_x^z \cdot x_t + b^z) \tag{2.19}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot z_t \tag{2.20}$$

$$h_t = o_t \odot Tanh(c_t) \tag{2.21}$$

In general, i_t is an input gate, f_t is a forget gate, and o_t is an output gate. These gates control the scale of the gradient to prevent them from vanishing or exploding. c_t is the cell of the LSTM, which maintains long-term information. h_t is the output of the current step, which is also called a hidden state of the LSTM. The architecture of

LSTM is shown in Figure 2-3.

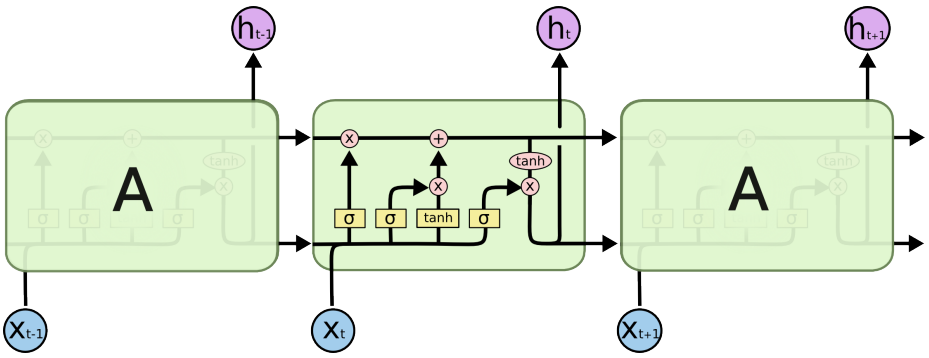


Figure 2-3: An illustration of the LSTM architecture. x_t stand for the input and h_t stands for the output hidden state at the t-th step [Olah, 2015].

The LSTM architecture is frequently used in the recent studies in the area of natural language because it is good at modeling sequences and easy to train. In this thesis, many studies directly applied LSTMs or made improvements to the architecture of LSTMs.

Chapter 3

Language Modeling with Graph Temporal Convolutional Networks

3.1 Introduction

Recently, there have been attempts to use non-recurrent neural models for language modeling but a noticeable performance gap still remains [Bai et al., 2018]. We propose a non-recurrent neural language model, dubbed graph temporal convolutional network (GTCN), that relies on graph neural network blocks and convolution operations. While the standard recurrent neural network language models encode sentences sequentially without modeling higher-level structural information, our model regards sentences as graphs and processes input words within a message propagation framework, aiming to learn better syntactic information by inferring skip-word connections. Specifically, the graph network blocks operate in parallel and learn the underlying graph structures in sentences without any additional annotation pertaining to structure knowledge. Experiments demonstrate that a model without recurrent connections can achieve comparable perplexity results in language modeling tasks and successfully learn syntactic information.

Recurrent neural networks (RNNs) are currently considered the de facto tool for language modeling tasks. RNNs encode sentences by processing each word in a sequential manner. There is a strong inductive bias, though usually implicit, that

the structure of a sentence is a sequence of words. However, natural language has complex underlying syntactic structures. Although recent work [Kuncoro et al., 2018] has demonstrated that RNNs are capable of capturing structural dependencies as long as they have enough capacity, in practice, it is less clear whether RNNs are able to capture such structural information with limited resources, especially over long distances.

The underlying structures of sentences are useful in natural language understanding and generation. To fully understand the meaning of a sentence, a linguistic analysis not only needs to recognize the semantics of words in the sentence, but also needs to understand how the words are organized in the sentence. To compose a correct sentence, one should also follow grammars and syntactic rules. Unfortunately, conventional recurrent architectures are not specifically designed for modeling such structures. As a result, poor performance can be expected for tasks requiring structural information.

Traditionally, natural language parsers are used to model grammar and syntactic knowledge. By generating a parse tree, the parser can describe the structure and word dependencies of a sentence. However, to generate such a structure often involves some form of supervised training of a parser. Recent works have demonstrated several ways of learning structural information of natural sentences without supervision, which can be helpful when performing language modeling tasks [Cheng et al., 2016, Shen et al., 2017].

In contrast to RNNs, convolutional neural networks (CNNs) might be good candidates for learning structures due to their use of hierarchical filters. Recently, increasing attention has been paid to the use of CNNs for language modeling tasks. [Dauphin et al., 2016] propose a gated convolutional language model, and the experiments in [Bai et al., 2018] show that temporal convolutional networks (TCNs) perform almost as well as RNNs for many tasks. However, there is a noticeable performance gap between TCNs and RNNs. Furthermore, those models are not easily interpretable in that they do not explicitly learn the structures of sentences.

A more dedicated architecture for modeling non-sequential structures is based on graph networks (GNs) [Scarselli et al., 2009]. The underlying structure among nodes

is described with adjacency matrices on which the model propagates messages between different nodes [Kipf and Welling, 2016, Li et al., 2016].

In this work we aim to demonstrate how a non-recurrent neural model can more explicitly model structural information in sentences without supervision for improving language modeling tasks. Specifically, this non-recurrent neural model consists of a novel convolutional architecture combined with graph network blocks. To avoid the sequential inductive bias of natural sentences, we regard the underlying structures of the sentences as fully connected graphs to learn as much syntactic information as possible. We also apply a message propagation mechanism to learn the representations of words and sentences. Though individual modules used in this research have been previously reported, to our knowledge, the combination of these components to achieve the reported results on language modeling tasks is novel.

The contributions of this work are as follows:

1. We show how RNNs model implicit connections between words, though they carry a sequential inductive bias.

2. We propose a graph- and convolution- based language model, which can run in parallel. To our knowledge, the model achieves state-of-the-art results on the Penn Treebank dataset compared to existing non-recurrent language models.

3. We introduce an unsupervised approach to learn syntactic parsing structure and other dependencies among words, which further improves the performance of our proposed language model.

3.2 Related Work

RNNs have been widely applied to language modeling tasks for over a decade. For example, [Cheng et al., 2016] use a long short-term memory (LSTM) RNN to learn a self-attention mechanism that indicates contextual dependencies in a language modeling task. As another example, [Shen et al., 2017] propose an unsupervised learning method for syntactic distance, in which the language model can generate a

parsing tree of a sentence. To achieve faster training speed, [Gao et al., 2018] adopts a group strategy for recurrent layers. Also, inspired by human speed reading mechanisms, several RNN variants [Seo et al., 2017, Campos et al., 2017] have been proposed to decide how to update their hidden states based on the importance of input tokens in a sequence. In contrast to RNN-based language models, it has been recently argued that temporal convolutional networks (TCNs) have competitive performance in some NLP tasks including word-level and character-level language modeling [Bai et al., 2018]. As is the case for RNNs, TCNs cannot easily model relational information well by design.

Many kinds of sequential data can be represented as graphs, in which nodes stand for entities and edges stand for relations. In order to model relational structures, our framework is based on graph neural networks (GNNs), which were first proposed in [Scarselli et al., 2009]. Subsequently, the authors in [Li et al., 2016] replaced the Almeida-Pineda algorithm used in [Scarselli et al., 2009] with backpropagation. In [Garcia and Bruna, 2017] GNNs were applied to image classification tasks in a "few-shot" setting. In [Gilmer et al., 2017], GNNs were applied to molecular property prediction tasks. The authors also summarized and generalized a message-passing mechanism. [Kipf and Welling, 2016] proposed graph convolution networks (GCNs) for semi-supervised learning, which conduct message propagation without learning edge representations. [Li et al., 2016] introduce a gating mechanism, which improves the performance of GNNs. Unfortunately, there are relatively few papers discussing how to adapt GNNs to natural language tasks. [Marcheggiani and Titov, 2017] applied GNNs to semantic role labeling. [Schlichtkrull et al., 2017] used GNNs to perform knowledge base completion tasks. [Johnson, 2016] generated a graph based on textual input and update the relationship during training. [Scarselli et al., 2009] proposed a neural model for message propagation in graphs, using an adjacency matrix, A, of a target graph to maintain structural information. Finally, [Seo et al., 2016] combined GNNs with RNNs for graph structured data.

3.3 Background

3.3.1 Long Short-Term Memory and Variants

The LSTM [Hochreiter and Schmidhuber, 1997] attempts to solve gradient problems associated with training recurrent networks and memorize longer sequences. The basic feedforward propagation method of LSTMs is described in Section 2.2.4.

A variant of LSTMs, named Quasi-RNNs (QRNN) [Bradbury et al., 2016], calculates z, f, o, i in parallel by replacing h_{t-1} with x_{t-1}, which significantly accelerates training. However, QRNNs still have to calculate cell values sequentially.

Applying the inductive bias that the underlying structure of natural sentences are linear sequences, LSTMs only explicitly model sequential structures. The relation between two words with a distance larger than 1 is modeled implicitly with a sequence of input and forget gates. We can rewrite the message propagation in LSTM cells in the form of $c = A \cdot z$, where $c = [c_1, c_2, \ldots, c_t]^T$, and $z = [z_1, z_2, \ldots, z_t]^T$ as follow:

$$c = \underbrace{\begin{bmatrix} i_1 & & & \\ f_2 i_1 & i_2 & & \\ \cdots & \cdots & & \\ \prod_{i=2}^{t} f_i \cdot i_1 & \prod_{i=3}^{t} f_i \cdot i_2 & \cdots & i_t \end{bmatrix}}_{A} \begin{bmatrix} z_1 \\ z_2 \\ \cdots \\ z_t \end{bmatrix} \tag{3.1}$$

In the above equation, each element in A, A_{ij} indicates the amount of message passed from z_j to z_i, where

$$a_{ij, i \neq j} = \prod_{j}^{i} f_{j+1} \cdot i_j \tag{3.2}$$

$$a_{ii} = i_i \tag{3.3}$$

Inspired by recent work on graph neural models (GNNs) [Scarselli et al., 2009] and graph convolution networks (GCNs) [Kipf and Welling, 2016], we can regard A as the adjacency matrix of a weighted, directed graph $G = (z, E)$, where $z = [z_1, z_2, \ldots, z_t]^T$

are nodes and each edge $e \in E$ indicates the amount of information needed to be propagated from one node to another. Equation 3.1 shows how LSTMs implicitly learn relation between each pair of words in a sentence.

3.3.2 Graph Neural Networks

We argue that sentences can be structured as graphs, in which nodes stand for entities and edges stand for relations. Following [Scarselli et al., 2009, Kipf and Welling, 2016], the message propagation in a GCN is given by:

$$X^{l+1} = f(D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \cdot X^l W) \tag{3.4}$$

where x^l and x^{l+1} are the input and output of the current layer, f is a non-linear activation, D is the diagonal degree matrix of adjacency matrix A, and W stands for a learnable weight matrix in the current layer.

The gating mechanism can also be applied to message propagation in graphical models. [Li et al., 2016] propose gated graph neural networks (GGNNs), which filter the embeddings of the target node and its neighbors with gates instead of simply averaging them as in GCNs.

3.4 Graph Temporal Convolutional Networks

Inspired by the analysis of LSTMs and recent work on graphical neural models, in this section we introduce graph temporal convolutional networks (GTCNs) for language modeling. We are motivated by the fact that we can calculate the adjacency matrix, A, in parallel, and thus avoid the difficulties of training recurrent neural structures and adopting the sequential hypothesis for language models. The GTCN model consists of two modules: context attention and message propagation.

3.4.1 Position-aware Context Attention

To decide how much the previous words contribute to predicting the next word, the GTCN model generates an attention over a context for each target word. Given the input sequence consists of n words, $X = [X_1, X_2, \ldots, X_n]$, we calculate key and value vectors, k and v, for each word for calculating the attention,

$$q_i = tanh(W_1^q \cdot x_i + W_2^q \cdot x_{i-1} + b^q) \tag{3.5}$$

$$k_i = tanh(W_1^k \cdot x_i + w_{2\,i-1}^k + b^k) \tag{3.6}$$

where $*$ is a convolution operation.

For words x_i and $x_j (j < i)$, we calculate their relation a_{ij} as follows,

$$a_{ij} = \frac{e_{ij}}{\sum_{s<k<i} e_{ik}} \tag{3.7}$$

$$e_{ij} = exp(q_i \cdot (k_j + W_{i-j}^p) + b^e) \tag{3.8}$$

where s is the start location of a visible window, and W_d^p is the relative position representation proposed in [Shaw et al., 2018], which encodes the distances between different words.

3.4.2 Message Propagation

We apply the gated graph neural network (GGNN) architecture described in [Li et al., 2016] for message propagation in each layer of the GTCN. In contrast to the original GGNN, the GTCN message propagation method does not have a recurrent architecture. Given input sequence $X = [x_1, x_2, \ldots, x_t]^T$, we summarize the entire sentence with generated attention to predict x_{t+1}. The context representation x_t^c is calculated as follows:

$$c_t = \sum_{i}^{t-1} a_{ti} x_i \tag{3.9}$$

We then calculate input gate i, forget gate f, and residual gate r with x_t and x_t^c,

$$g = \sigma(W_1^g x_t + W_2^g c_t + b^g), g \in [i, f, r] \tag{3.10}$$

We also calculate an output gate for the GTCN,

$$o = \sigma(W_1^o x_t + W_2^o x_{t-1} + b^o) \tag{3.11}$$

The GTCN predicts the embedding of x_{t+1} as

$$m_t = o \odot \phi(W_1^h \cdot (x_t \odot i) + W_2^h \cdot (c_t \odot f) + b^h) \tag{3.12}$$

$$\hat{x}_{t+1} = h_t = i \odot m_t + (1 - i) \odot m_t \tag{3.13}$$

where $\odot$ stands for element-wise product, and ϕ is a non-linear activation function.

3.4.3 Syntax Learning and Representation

Given a sentence $X = [x_1, x_2, \ldots, x_n]$, the GTCN model generates a attention weights for every pair of words. In other words, for each word x_t, there is a corresponding attention weight sequence $[a_1, a_2, \ldots, a_{t-1}]$. The attention weights are calculated with Equation 8 described above. This feature makes our approach different from the syntactic distance model [Shen et al., 2017], which learns one parsing tree for an entire sentence.

Similarly to recurrent neural network grammars (RNNG) [Dyer et al., 2016], our model constructs a unique parsing tree for each word in an input sentence with other visible tokens. Experiment shows that the attention weights generated by the GTCN captures information about the structure of the ground truth parse tree.

3.4.4 Other Details

Convolution Windows We hope the multi-layer GTCN model is able to learn different levels of semantics in the given sentence. In our design of the model, the lower GTCN layers learn local information, while the upper layers learn information from a longer context. In practice, the size of the context window of layer i is $i \cdot L$, where L is the context window length of the first GTCN layer. Experiments show that this strategy works better than making the entire sentence visible for all layers and using the same context window size for each layer.

Variational Dropout The standard dropout model [Srivastava et al., 2014] randomly samples masks for input tensors at each time step in sequence processing. In our GTCN model, we employ variational dropout [Gal and Ghahramani, 2016], which samples one mask for the same variables in different time steps, to improve training convergence.

3.5 Experiments

The experiments are designed to answer the following questions: **Q1:** How effective is the graph convolutional language model, compared with RNN-based language models? **Q2:** Can the GTCN language model learn syntactic information without any annotations?

3.5.1 Settings

We conduct experiments on language modeling (LM) on the Penn Treebank (PTB) dataset [Mikolov et al., 2010], which includes 10,000 different words. Our GTCN model employed 4 layers, in which each layer applies a convolution window sized 10, 20, 30, and 40 respectively. The embedding size of words is chosen as 400, and the hidden layers were 800 dimensions. We also applied tied weights for the encoder and decoder.

In the optimization process, we apply stochastic gradient descent (SGD) and the

Model	PPL	Model Size	Recurrence	Syntax
LSTM (Large)	78.4	66M	Yes	-
Variational LSTM	73.4	66M	Yes	-
LSTM + Cache	72.1	-	Yes	-
Variational RHN	65.4	23M	Yes	-
IndRNN	65.3	-	Yes	-
AWD-LSTM	**57.3**	24M	Yes	-
TCN	88.7	-	-	-
CharCNN	78.9	19M	-	-
GTCN (ours)	**66.1**	27M	-	Yes
RNNG	102.4	-	Yes	Yes
PRPN	**62.0**	-	Yes	Yes

Table 3.1: Evaluation of test perplexity of different recent models, including LSTM (Large) [Zaremba et al., 2014], Variational LSTM [Gal and Ghahramani, 2016], LSTM + Cache [Grave et al., 2016], Variational RHN [Zilly et al., 2016], IndRNN [Li et al., 2018], AWD-LSTM [Merity et al., 2017], RNNG [Dyer et al., 2016], and PRPN [Shen et al., 2017].

average-SGD (ASGD) strategy proposed in [Merity et al., 2017]. The batch size is 20, and the length of training sequences is 70. The initial learning rate of the SGD step is 30. The dropout rate of the embedding layer is 0.4, while the hidden layers apply 0.25.

3.5.2 Effectiveness (Q1)

Table 3.1 shows the performance of our model and lists a selection of recent baselines. The experiments show that our model outperforms many strong baselines, but is not as good as the current best LSTM language models. However, our model outperforms other CNN-based model significantly, decreasing the perplexity of the TCN model by over 20 points. To our knowledge, this is the first convolution based neural language model that has reached such performance on the PTB dataset. We also compare our model with prior work that make use of syntactic information in language modeling. The GTCN model performs better than all models in this family except the parse-read-predict network (PRPN) [Shen et al., 2017], which employed an LSTM architecture with semantic and syntactic attention.

3.5.3 Learning Structural Information (Q2)

In this section, we examine the structural information learned by the GTCN language model with an example. By taking the example "I shot an elephant in my pajamas[1]" as the input of a trained 4-layer GTCN language model, we can visualize the attentions generated by GTCN while encoding the sentence, for predicting the words, on each layer of the network.

Figure 3-1 shows two parsing trees of the example sentence. In this example, the parsing tree on the left makes more sense because "I" am more likely to be "in my pajamas". Admitting both are possible. With this example, we show how the attention mechanism works in the GTCN.

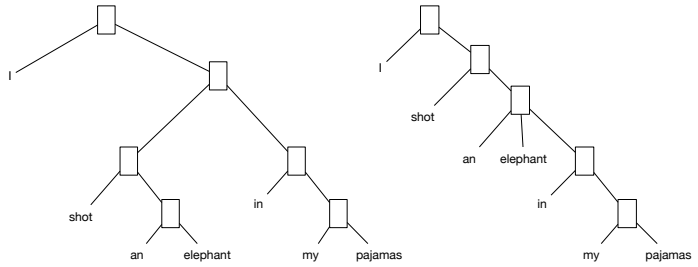


Figure 3-1: Two possible parse trees for the sentence "I shot an elephant in my pajamas."

To analyze the syntactic information learned by the model, we visualize the attentions generated while processing each word. For each word, we plot its attention weights on all GTCN layers.

Figure 3-2 visualizes attentions generated when the GTCN model processes the input sentence. Tree A is the groundtruth parsing tree and Tree B is the generated tree of attention weights output by the GTCN. The target word processed at each step is shown in red. Slashed edges stand for words and nodes that are not visible at the current step. The arrows are the edges of temporal parse trees constructed only with visible words. We color the visible nodes in the parse trees according to attentions shown above.

Since we calculate attention weights for every pair of words, the GTCN can use

[1] `https://www.nltk.org/book/ch08.html`

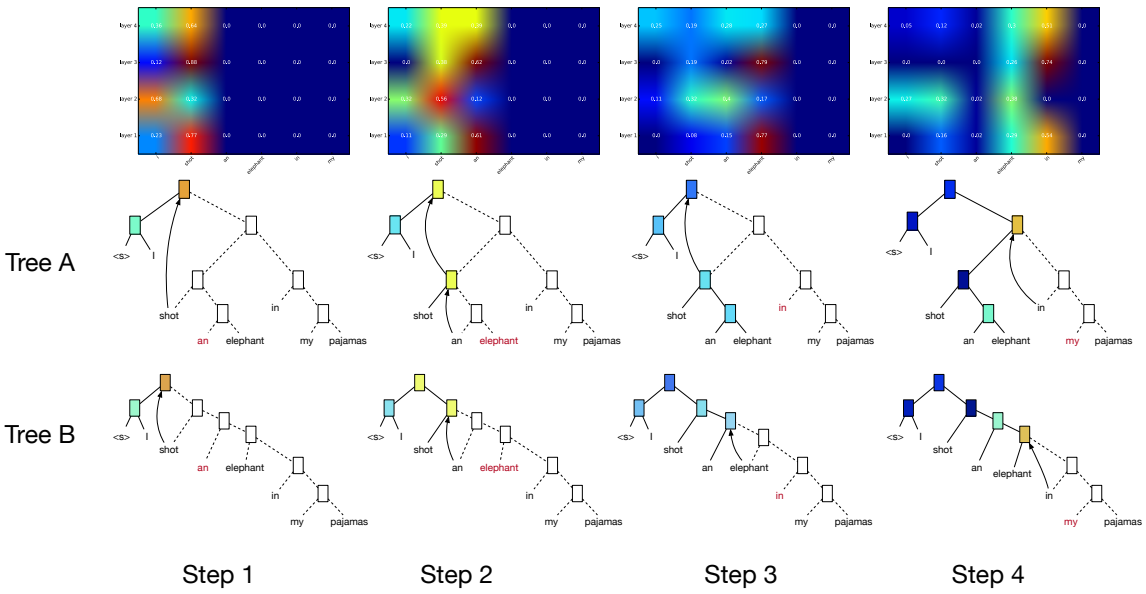


Figure 3-2: Attentions generated when processing target words marked with red. The upper plots show the attention weights, and the lower figures show the possible underlying syntactic structures.

different syntactic information while processing the sentence at different steps for better encoding target words. For example, when processing the word "elephant", the model highlights "shot" and "an". In the Step 4 of parsing tree B, "elephant" and "in" receive higher weights when the model encodes word "my". In step 3, while processing word "in", the model assigns higher weights on "shot", and "elephant", but pays less attention to "shot". This suggests that the model decides that "in" is leading a word sequence that describes "an elephant". This prediction indicates that the model prefers the second parse tree when encoding the sentence.

Although the model generates different attentions on context while processing different words, we find a common phenomenon of the learned attention weights. As illustrated in Figure 3-2, the attention mechanism learns a path from target words to the root (PTR) of the parse trees. In general, the nodes on this path are assigned higher attention weights. While modeling the word "in", the model decides that "an elephant" contains enough information to encode "in", so it does not use much higher level information.

The above example illustrates that our model is more flexible than CNNs and LSTMs in learning structures. CNNs have a fixed kernel size and dilation size [Bai et al., 2018]. The structure of context modeling in CNNs totally depends on hyper-parameter settings, including kernel size, dilation size, and the number of layers. In contrast, LSTMs process sentence sequentially, and the relation between words i and j, $a_{i,j}$, is always smaller than $a_{i+1,j}$. With our model, which does not employ fixed-length filters or recurrent cells, the language model can learn more information about context structures, such as PTR, for better predictions.

3.6 Conclusion and Future Work

In this work, we proposed a graph temporal convolution network (GTCN) for language modeling that processes natural sentences as graphs instead of linear sequences. Without any recurrent components, our proposed model can run in parallel. Although there is still a performance gap between our model and state-of-the-art RNN-based LMs, our GTCN LM significantly outperforms existing convolution-based models. We also proposed the right-root parsing trees (RPTs) for representing syntactic information of sub-sentences in language modeling tasks. By visualizing the attention weights generated by the GTCN and constructing the RPTs, we illustrated that the GTCN can capture syntactic information for language modeling. We believe that due to the ability to run in parallel and capturing syntactic information, it is worth exploring further how the GTCN model performs on other natural language processing applications that need sentence encoding with syntactic knowledge.

Chapter 4

Improving Neural Language Models by Segmenting, Attending, and Predicting the Future

4.1 Introduction

Neural language models are typically trained by predicting the next word given a past context [Bengio et al., 2003]. However, natural sentences are not constructed as simple linear word sequences, as they usually contain complex syntactic information. For example, a subsequence of words can constitute a phrase, and two non-neighboring words can depend on each other. These properties make natural sentences more complex than simple linear sequences.

Most recent work on neural language modeling learns a model by encoding contexts and matching the context embeddings to the embedding of the next word [Bengio et al., 2003, Merity et al., 2017, Melis et al., 2017]. In this line of work, a given context is encoded with a neural network, for example a long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] network, and is represented with a distributed vector. The log-likelihood of predicting a word is computed by calculating the inner product between the word embedding and the context embedding. Although

most models do not explicitly consider syntax, they still achieve state-of-the-art performance on different corpora. Efforts have also been made to utilize structural information to learn better language models. For instance, parsing-reading-predict networks (PRPN) [Shen et al., 2017] explicitly learn a constituent parsing structure of a sentence and predict the next word considering the internal structure of the given context with an attention mechanism. Experiments have shown that the model is able to capture some syntactic information.

Similar to word representation learning models that learns to match word-to-word relation matrices [Mikolov et al., 2013, Pennington et al., 2014], standard language models are trained to factorize context-to-word relation matrices [Yang et al., 2017]. In such work, the context comprises all previous words observed by a model for predicting the next word. However, we believe that context-to-word relation matrices are not sufficient for describing how natural sentences are constructed. We argue that natural sentences are generated at a higher level before being decoded to words. Hence a language model should be able to predict the following sequence of words given a context. In this work, we propose a model that factorizes a context-to-phrase mutual information matrix to learn better language models. The context-to-phrase mutual information matrix describes the relation among contexts and the probabilities of phrases following given contexts. We make the following contributions:

- We propose a phrase prediction model that improves the performance of state-of-the-art word-level language models.
- Our model learns to predict approximate phrases and headwords without any annotation.

4.2 Related Work

Neural networks have been widely applied in natural language modeling and generation [Bengio et al., 2003, Bahdanau et al., 2014] for both encoding and decoding. Among different neural architectures, the most popular models are recurrent

neural networks (RNNs) [Mikolov et al., 2010], long short-term memory networks (LSTMs) [Hochreiter and Schmidhuber, 1997], and convolutional neural networks (CNNs) [Bai et al., 2018, Dauphin et al., 2017].

Many modifications of network structures have been made based on these architectures. LSTMs with self-attention can improve the performance of language modeling [Tran et al., 2016, Cheng et al., 2016]. As an extension of simple self-attention, transformers [Vaswani et al., 2017] apply multi-head self-attention and have achieved competitive performance compared with recurrent neural language models. A current state-of-the-art model, Transformer-XL [Dai et al., 2019], applied both a recurrent architecture and a multi-head attention mechanism. To improve the quality of input word embeddings, character-level information is also considered [Kim et al., 2016]. It has also been shown that context encoders can learn syntactic information [Shen et al., 2017].

However, instead of introducing architectural changes, for example a self-attention mechanism or character-level information, previous studies have shown that careful hyper-parameter tuning and regularization techniques on standard LSTM language models can obtain significant improvements [Melis et al., 2017, Merity et al., 2017]. Similarly, applying more careful dropout strategies can also improve language models [Gal and Ghahramani, 2016, Melis et al., 2018]. LSTM language models can be improved with these approaches because LSTMs suffer from serious over-fitting problems.

Recently, researchers have also attempted to improve language models at the decoding phase. [Inan et al., 2016] showed that reusing the input word embeddings in the decoder can reduce the perplexity of language models. [Yang et al., 2017] showed the low-rank issue in factorizing the context-to-word mutual information matrix and proposed a multi-head softmax decoder to solve the problem. Instead of predicting the next word by using only similarities between contexts and words, the neural cache model [Grave et al., 2016] can significantly improve language modeling by considering the global word distributions conditioned on the same contexts in other parts of the corpus.

To learn the grammar and syntax in natural languages, [Dyer et al., 2016] proposed the recurrent neural network grammar (RNNG) that models language incorporating

a transition parsing model. Syntax annotations are required in this model. To utilize the constituent structures in language modeling without syntax annotation, parse-read-predict networks (PRPNs) [Shen et al., 2017] calculate syntactic distances among words and computes self-attentions. Syntactic distances have been proved effective in constituent parsing tasks [Shen et al., 2018a]. In this work, we learn phrase segmentation with a model based on this method and our model does not require syntax annotation.

4.3 Syntactic Height and Phrase Induction

In this work, we propose a language model that not only predicts the next word of a given context, but also attempts to match the embedding of the next phrase. The first step of this approach is conducting phrase induction based on syntactic heights. In this section, we explain the definition of syntactic height in our approach and describe the basics ideas about whether a word can be included in an induced phrase.

Intuitively, the syntactic height of a word aims to capture its distance to the root node in a dependency tree. In Figure 4-1, the syntactic heights are represented by the red bars. A word has high syntactic height if it has low distance to the root node.

A similar idea, named syntactic distance, is proposed by [Shen et al., 2017] for constructing constituent parsing trees. We apply the method for calculating syntactic distance to calculate syntactic height. Given a sequence of embeddings of input words $[x_1, x_2, \cdots, x_n]$, we calculate their syntactic heights with a temporal convolutional network (TCN) [Bai et al., 2018].

$$d_i = W_d \cdot [x_{i-n}, x_{i-n+1}, \cdots, x_i]^T + b_d \tag{4.1}$$

$$h_i = W_h \cdot ReLU(d_i) + b_h \tag{4.2}$$

where h_i stands for the syntactic height of word x_i. The syntactic height h_i for each word is a scalar, and W_h is a $1 \times D$ matrix, where D is the dimensionality of d_i. These heights are learned and not imposed by external syntactic supervision. In

[Shen et al., 2017], the syntactic heights are used to generate context embeddings. In our work, we use the syntactic heights to predict induced phrases and calculate their embeddings.

We define the phrase induced by a word based on the syntactic heights. Consider two words x_i and x_k. x_k belongs to the phrase induced by x_i if and only if for any $j \in (i, k)$, $h_j < max(h_i, h_k)$. For example, in Figure 4-1, the phrase induced by the red marked word **the** is "the morning flights", since the syntactic height of the word **morning**, $h_{morning} < h_{flights}$. However, the word "to" does not belong to the phrase because $h_{flights}$ is higher than both h_{the} and h_{to}. The induced phrase and the inducing dependency connection are labeled in blue in the figure.

Note that this definition of an induced phrase does not necessarily correspond to a phrase in the syntactic constituency sense. For instance, the words "to Houston" would be included in the phrase "the morning flights to Houston" in a traditional syntactic tree. Given the definition of induced phrases, we propose phrase segmenting conditions (PSCs) to find the last word of an induced phrase. Considering the induced phrase of the i-th word, $s_i = [x_i, x_{i+1}, \cdots, x_j]$, there are two conditions that x_j should satisfy:

1. The syntactic height of x_j must be higher than the height of x_i, that is

$$h_j - h_i > 0 \tag{4.3}$$

2. The syntactic height of x_{j+1} should be lower that x_j.

$$h_j - h_{j+1} > 0 \tag{4.4}$$

Given the PSCs, we can decide the induced phrases for the sentence shown in Figure 4-1. The last word of the phrase induced by "United" is "canceled", and the last word of the phrase induced by "flights" is "Houston".

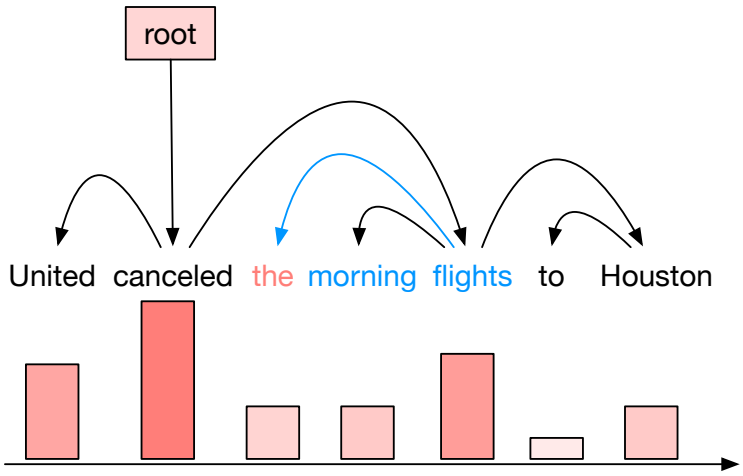


Figure 4-1: Groundtruth dependency tree and syntactic heights of each word.

4.4 Model

In this work, we formulate multi-layer neural language models as a two-part framework. For example, in a two-layer LSTM language model [Merity et al., 2017], we use the first layer as phrase generator and the last layer as a word generator:

$$[c_1, c_2, \cdots, c_T] = RNN^1([x_1, x_2, \cdots, x_T]) \tag{4.5}$$

$$[y_1, y_2, \cdots, y_T] = RNN^2([c_1, c_2, \cdots, c_T]) \tag{4.6}$$

For a L-layer network, we can regard the first L_1 layers as the phrase generator and the next $L_2 = L - L_1$ layers as the word generator. Note that we use y_i to represent the hidden state output by the second layer instead of h_i, since h_i in our work is defined as the syntactic height of x_i. In the traditional setting, the first layer does not explicitly learn the semantics of the following phrase because there is no extra objective function for phrase learning.

We force the first layer to output context embeddings c_i for phrase prediction with three steps. Firstly, we predict the induced phrase for each word. Secondly, we calculate the embedding of each phrase with a head-finding attention. Lastly, we align the context embedding and phrase embedding with negative sampling. The word

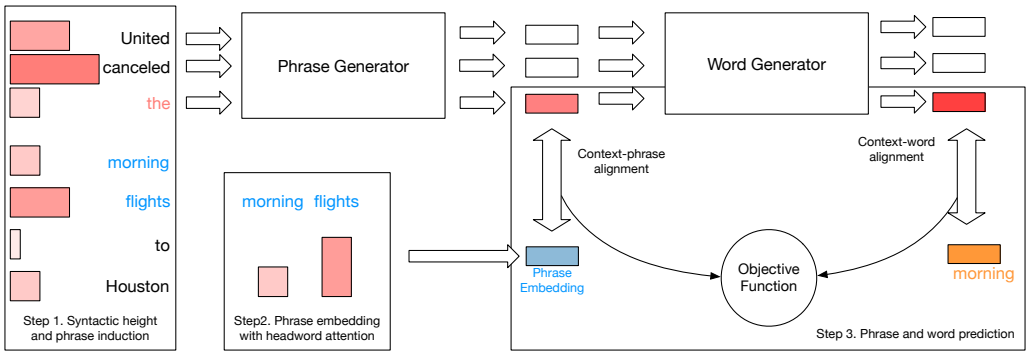


Figure 4-2: The 3-step diagram of our approach. The current target word is "the", the induced phrase is "morning flights", and the next word is "morning".

generation is trained in the same way as standard language models. The diagram of the model is shown in Figure 4-2.

4.4.1 Phrase Segmentation

We calculate the syntactic height and predict the induced phrase for each word:

$$h_i = TCN([x_{i-n}, x_{i-n+1}, \cdots, x_i]) \tag{4.7}$$

where $TCN(\cdot)$ stands for the TCN model described in Equations 4.1 and 4.2, and n is the width of the convolution window.

Based on the proposed phrase segmenting conditions (PSCs) described in the previous section, we predict the probability of a word being the first word outside a induced phrase. Firstly, we decide if each word, $x_{j-1}, j \in (i+1, n]$, satisfies the two phrase segmenting conditions, PSC-1 and PSC-2. The probability that x_j satisfies PSC-1 is

$$p^1_{psc}(x_j) = \frac{1}{2} \cdot (f^{HT}(h_j - h_i) + 1) \tag{4.8}$$

Similarly, the probability that x_j satisfies PSC-2 is

$$p^2_{psc}(x_j) = \frac{1}{2} \cdot (f^{HT}(h_j - h_{j+1}) + 1) \tag{4.9}$$

where f_{HT} stands for the HardTanh function with a temperature a:

$$f^{HT}(x) = \begin{cases} -1 & x \leq -\frac{1}{a} \\ a \cdot x & -\frac{1}{a} < x \leq \frac{1}{a} \\ 1 & x > \frac{1}{a} \end{cases}$$

This approach is inspired by the context attention method proposed in the PRPN model [Shen et al., 2017].

Then we can infer the probability of whether a word belongs to the induced phrase of x_i with

$$p^{ind}(x_j) = \prod_{k=1}^{j} \hat{p}(x_k) \tag{4.10}$$

where $p^{ind}(x_i)$ stands for the probability that x_i belongs to the induced phrase, and

$$\hat{p}(x_k) = \begin{cases} 1 & k \leq i+1 \\ 1 - p^1_{psc}(x_{k-1}) \cdot p^2_{psc}(x_{k-1}) & k > i+1 \end{cases}$$

Note that the factorization in Equation 4.10 assumes that words are independently likely to be included in the induced phrase of x_i.

4.4.2 Phrase Embedding with Attention

Given induced phrases, we can calculate their embeddings based on syntactic heights. To calculate the embedding of phrase $s = [x_1, x_2, \cdots, x_n]$, we calculate an attention distribution over the phrase:

$$\alpha_i = \frac{h_i \cdot p^{ind}(x_i) + c}{\sum_j h_j \cdot p^{ind}(x_j) + c} \tag{4.11}$$

where h_i stands for the syntactic height for word x_i and c is a constant real number for smoothing the attention distribution. Then we generate the phrase embedding

with a linear transformation:

$$s = W \cdot \sum_i \alpha_i \cdot e_i \tag{4.12}$$

where e_i is the word embedding of x_i. In training, we apply a dropout layer on s.

4.4.3 Phrase and Word Prediction

A traditional language model learns the probability of a sequence of words:

$$p(x_1, x_2, \cdots, x_n) = p(x_1) \cdot \prod_i p(x_{i+1}|x_1^i) \tag{4.13}$$

where x_1^i stands for $x_1, x_2, \cdots, x_i$, which is the context used for predicting the next word, x_{i+1}.

In our model, we learn the language model by jointly learning phrase prediction. The conditional probability of each word can be written as:

$$p(x_{i+1}, s_1^i|x_1^i) = p(s_1^i|x_1^i) \cdot p(x_{i+1}|s_1^i) \tag{4.14}$$

where s_i stands for the phrase induced by word x_i and s_1^i stands for the induced phrase sequence s_1 to s_i. x_{i+1} is the next word of x_i, and also the second word in phrase s_i.

We use classical cross-entropy losses [Bengio et al., 2003, Merity et al., 2017] to train the word generation model $p(x_{i+1}|s_i)$ based on the induced phrase s_i. We define the phrase generation, or context-phrase alignment model as follows:

$$p(s_1^i|x_1^i) \approx \prod_i p(s_i|x_1^i) \tag{4.15}$$

$$p(s_i|x_1^i) = \sigma(c_i^T \cdot s_i) \tag{4.16}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$, and c_i stands for the context embedding of $x_1, x_2, \cdots, x_i$, defined in the next subsection.

We use an extra objective function and negative sampling to align context repre-

sentations and the embeddings of induced phrases. Given the context embedding c_i, the induced phrase embedding s_i, and random sampled negative phrase embeddings s_i^{neg}, we define the following objective function for context i:

$$l_i^{CPA} = 1 - \sigma(c_i^T \cdot s_i) + \frac{1}{n}\sum_{j=1}^{n} \sigma(c_i^T \cdot s_j^{neg}) \tag{4.17}$$

where n stands for the number of negative samples. With this loss function, the model learns to maximize the similarity between the context and true induced phrase embeddings, and minimize the similarity between the context and negative samples randomly selected from the induced phrases of other words.

It worth noting that our approach is model-agnostic and can be applied to various architectures. The TCN network for calculating the syntactic heights and phrase inducing is an independent module. In context-phrase alignment training with negative sampling, the objective function provides phrase-aware gradients and does not change the word-by-word generation process of the language model.

4.5 Experiments

We evaluate our model with word-level language modeling tasks on Penn Treebank (PTB) [Mikolov et al., 2010], Wikitext-2 (WT2) [Bradbury et al., 2016], and Wikitext-103 (WT103) [Merity et al., 2016] corpora.

The PTB dataset has a vocabulary size of 10,000 unique words. The entire corpus includes roughly 40,000 sentences in the training set, and more than 3,000 sentences in both valid and test set.

The WT2 data is about two times larger the the PTB dataset. The dataset is consist of Wikipedia articles. The corpus includes 30,000 unique words in its vocabulary and is not cleaned as heavily as the PTB corpus.

The WT103 corpus contains a larger vocabulary and more articles than WT2. It consists of 28k articles and more than 100M words in the training set. WT103 can evaluate the ability of capturing long-term dependencies [Dai et al., 2019].

In each corpus, we apply our approach to publicly-available, state-of-the-art models. This demonstrates that our approach can improve different existing architectures. Our trained models will be published for downloading.

Model	**#Params**	**Dev PPL**	**Test PPL**
[Inan et al., 2016] - Tied Variational LSTM	24M	75.7	73.2
[Zilly et al., 2017] - Recurrent Highway Networks	23M	67.9	65.7
[Shen et al., 2017] - PRPN	-	-	62.0
[Pham et al., 2018] - Efficient NAS	24M	60.8	58.6
[Melis et al., 2017] - 4-layer skip LSTM (tied)	24M	60.9	58.3
[Shen et al., 2018b] - ON-LSTM	25M	58.3	56.2
[Liu et al., 2018] - Differentiable NAS	23M	58.3	56.1
[Yang et al., 2017] - AWD-LSTM-MoS	22M	58.1	56.0
[Merity et al., 2017] - AWD-LSTM	24M	60.7	58.8
[Merity et al., 2017] - AWD-LSTM + fine-tuning	24M	60.0	57.3
Ours - AWD-LSTM + Phrase Induction - Attention	24M	60.2	58.0
Ours - AWD-LSTM + Phrase Induction	24M	59.6	57.5
Ours - AWD-LSTM + Phrase Induction + finetuning	24M	**57.8**	**55.7**
[Dai et al., 2019] - Transformer-XL	24M	56.7	54.5
[Yang et al., 2017] - AWD-LSTM-MoS + finetuning	22M	**56.5**	**54.4**

Table 4.1: Experimental results on Penn Treebank dataset. Compared with the AWD-LSTM baseline models, our method reduced the perplexity on test set by 1.6.

4.5.1 Penn Treebank

We train a 3-layer AWD-LSTM language model [Merity et al., 2017] on PTB data set. We use 1,150 as the number of hidden neurons and 400 as the size of word embeddings. We also apply the word embedding tying strategy [Inan et al., 2016]. We apply variational dropout for hidden states [Gal and Ghahramani, 2016] and the dropout rate is 0.25. We also apply weight dropout [Merity et al., 2017] and set weight dropout rate as 0.5. We apply stochastic gradient descent (SGD) and averaged SGD

Model	#Params	Dev PPL	Test PPL
[Inan et al., 2016] - Variational LSTM (tied)	28M	92.3	87.7
[Inan et al., 2016] - VLSTM + augmented loss	28M	91.5	87.0
[Grave et al., 2016] - LSTM	-	-	99.3
[Grave et al., 2016] - LSTM + Neural cache	-	-	68.9
[Melis et al., 2017] - 1-Layer LSTM	24M	69.3	69.9
[Melis et al., 2017] - 2-Layer Skip Connection LSTM	24M	69.1	65.9
[Merity et al., 2017] - AWD-LSTM + finetuning	33M	68.6	65.8
Ours - AWD-LSTM + Phrase Induction	33M	68.4	65.2
Ours - AWD-LSTM + Phrase Induction + finetuning	33M	**66.9**	**64.1**

Table 4.2: Experimental results on Wikitext-2 dataset.

(ASGD) [Polyak and Juditsky, 1992] for training. The learning rate is 30 and we clip the gradients with a norm of 0.25.

We compare the word-level perplexity of our model with other state-of-the-art models and our baseline is AWD-LSTM [Merity et al., 2017].. The experimental results are shown in Table 4.1. Although not as good as the Transformer-XL model [Dai et al., 2019] and the mixture of softmax model [Yang et al., 2017], our model significantly improved the AWD-LSTM, reducing 2.2 points of perplexity on the validation set and 1.6 points of perplexity on the test set.

We also did ablation study without headword attention and the result is listed in Table 4.1. Without the attention mechanism, the model performs worse than the full model but is still better than our baseline. Hence we just test the full model in the following experiments.

4.5.2 Wikitext-2

We also trained a 3-layer AWD-LSTM language model on the WT2 dataset. The network has the same input size, output size, and hidden size as the model we applied on PTB dataset, following the experiments done by [Merity et al., 2017]. Some hyper-

Model	#Params	Dev PPL	Test PPL
[Grave et al., 2016] - LSTM	-	-	48.7
[Bai et al., 2018] - TCN	-	-	45.2
[Dauphin et al., 2017] - GCNN-8	-	-	44.9
[Grave et al., 2016] - LSTM + Neural cache	-	-	40.8
[Dauphin et al., 2017] - GCNN-14	-	-	37.2
[Merity et al., 2018] - 4-layer QRNN	151M	32.0	33.0
[Rae et al., 2018] - LSTM + Hebbian + Cache	-	29.7	29.9
[Dai et al., 2019] - Transformer-XL Standard	151M	23.1	24.0
[Baevski and Auli, 2018] - Adaptive input	247M	19.8	20.5
[Dai et al., 2019] - Transformer-XL Large	257M	17.7	18.3
Ours - Transformer-XL Large + Phrase Induction	257M	-	**17.4**

Table 4.3: Experimental results on Wikitext-103 dataset.

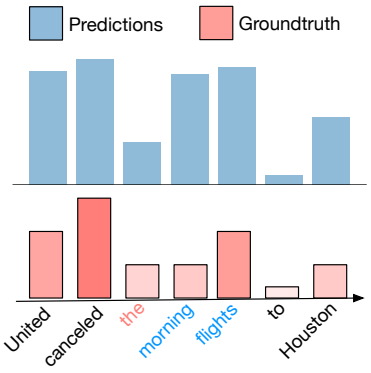
parameters are different from the PTB language model. We use a batch size of 60. The embedding dropout rate is 0.65 and the dropout rate of hidden outputs is set to 0.2. Other hyper-parameters are the same as we set in training on the PTB dataset.

The experimental results are shown in Table 4.2. Our model improves the AWD-LSTM model by reducing 1.7 points of perplexity on both the validation and test sets, while we did not make any change to the architecture of the AWD-LSTM language model.

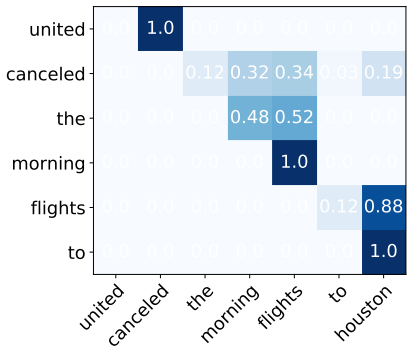
4.5.3 Wikitext-103

The current state-of-the-art language model trained on Wikitext-103 dataset is the Transformer-XL [Dai et al., 2019]. We apply our method on the state-of-the-art Transformer-XL Large model, which has 18 layers and 257M parameters. The input size and hidden size are 1024. 16 attention heads are used. We regard the first 14 layers as the phrase generator and the last 4 layers as the word generator. In other words, the context-phrase alignment is trained with the outputs of the 14th layer.

The model is trained on 4 Titan X Pascal GPUs, each of which has 12G memory. Because of the limitation of computational resources, we use our approach to fine-tune



(a) Syntactic heights of each word.



(b) Induced phrases and headword attentions.

Figure 4-3: Examples of induced phrases and corresponding headword attention for generating the phrase embedding. The word of each row stands for the target word as the current input of the language model, and the values in each row in the matrices stands for the words consisting the induced phrase and their weights.

the officially released pre-trained Transformer-XL Large model for 1 epoch. The experimental results are shown in Table 4.3. Our approach got 17.4 perplexity with the officially released evaluation scripts, significantly outperforming all baselines and achieving new state-of-the-art performance.

4.6 Discussion

In this section, we show what is learned by training language models with the context-phrase alignment objective function by visualizing the syntactic heights output by the TCN model and the phrases induced by each target word in a sentence. We also visualize the headword attentions over the induced phrase.

The first example is the sentence showed in Figure 4-1. The sentence came from [Jurafsky and Martin, 2014] and did not appear in our training set. Figure 4-1 shows the syntactic heights and the induced phrase of "the" according to the ground-truth dependency information. Our model is not given such high-quality inputs in either training or evaluation.

Figure 4-3 visualizes the structure learned by our phrase induction model. The inferred syntactic heights are shown in Figure 4-3a. Heights assigned to words "the"

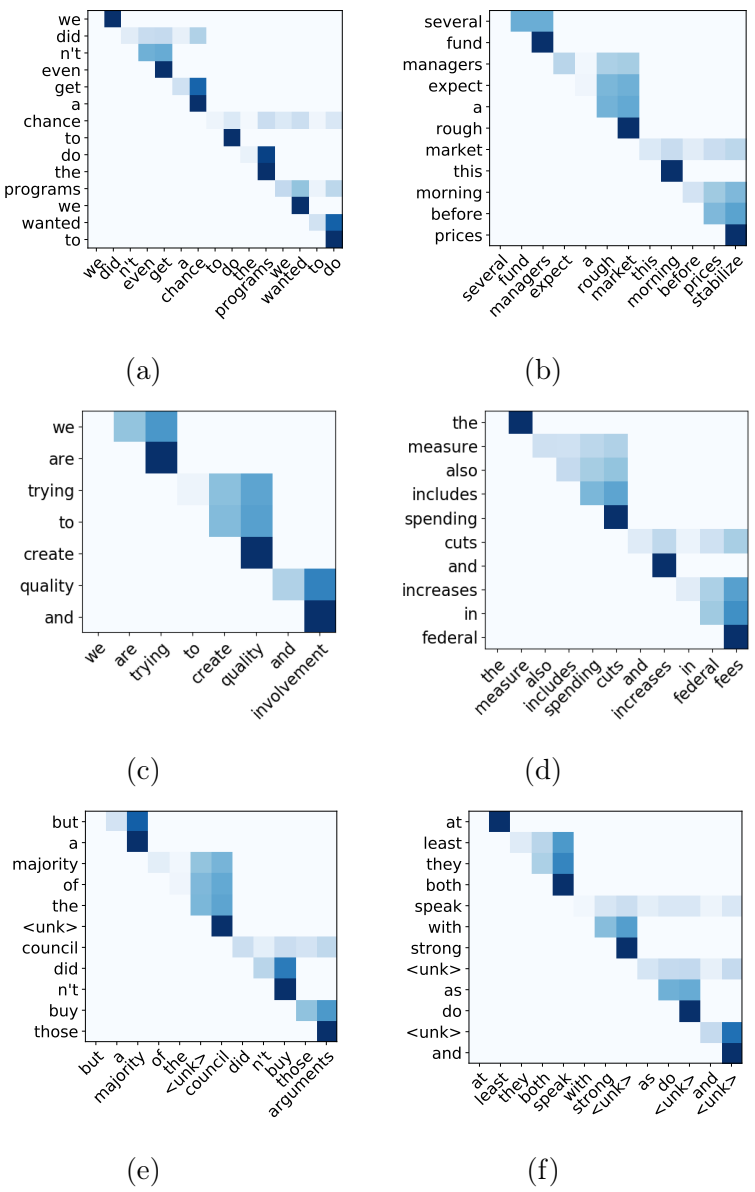


Figure 4-4: Examples of phrase inducing and headword attentions. Each row of the matrix stands for the induced phrase of a word listed at the beginning of each row. The darkness of the color indicates the attention weights.

and "to" are significantly lower than others, while the verb "canceled" is assigned the highest score in the sentence. Induced phrases are shown in Figure 4-3b. The words at the beginning of each row stand for the target word of each step. Values in the matrix stand for attention weights for calculating phrase embedding. The weights are calculate with the phrase segmenting conditions (PSC) and the syntactic heights described in Equations 4.8 to 4.11. For the target word "united", $h_{united} < h_{canceled}$

and $h_{canceled} > h_{the}$, hence the induced phrase of "united" is a single word "canceled", and the headword attention of "canceled" is 1, which is indicated in the first row of Figure 4-3b. The phrase induced by "canceled" is the entire following sequence, "the morning flights to houston", since no following word has a higher syntactic height than the target word. It is also shown that the headword of the induced phrase of "canceled" is "flights", which agrees with the dependency structure indicated in Figure 4-1.

More examples are shown in Figure 4-4. Figures 4-4a to 4-4d show random examples without any unknown word, while the examples shown in Figures 4-4e and 4-4f are randomly selected from sentences with unknown words, which are marked with the UNK symbol. The examples show that the phrase induction model does not always predict the exact structure represented by the dependency tree. For example, in Figure 4-4b, the TCN model assigned the highest syntactic height to the word "market" and induced the phrase "expect a rough market" for the context "the fund managers". However, in a ground-truth dependency tree, the verb "expect" is the word directly connected to the root node and therefore has the highest syntactic height.

Although not exactly matching linguistic dependency structures, the phrase-level structure predictions are reasonable. The segmentation is interpretable and the predicted headwords are appropriate. In Figure 4-4c, the headwords are "trying", "quality", and "involvement". The model is also robust with unknown words. In Figure 4-4e, "the <unk> council" is segmented as the induced phrase of "but a majority of". In this case, the model recognized that the unknown word is dependent of "council". The sentence in Figure 4-4f includes even more unknown words. However, the model still correctly predicted the root word, the verb "speak". For the target word "with", the induced phrase is "strong <unk>". Two unknown words are located in the last few words of the sentence. The model failed to induce the phrase "<unk> and <unk>" for the word "do", but still successfully split "<unk>" and "and". Meanwhile, the attentions over the phrases induced by "speak", "do", and the first "<unk>" are not quite informative, suggesting that unknown words made some difficulties for headword prediction in this example. However, the unknown words are assigned significantly higher syntactic heights than the word "and".

4.7 Conclusion

In this work, we improved state-of-the-art language models by aligning context and induced phrases. We defined syntactic heights and phrase segmentation rules. The model generates phrase embeddings with headword attentions. We improved the AWD-LSTM and Transformer-XL language models on different data sets and achieved state-of-the-art performance on the Wikitext-103 corpus. Experiments showed that our model successfully learned approximate phrase-level knowledge, including segmentation and headwords, without any annotation. In future work, we aim to capture better structural information and possible connections to unsupervised grammar induction.

Chapter 5

Cross-Sentence Attention for Co-reference Resolution

5.1 Introduction

In this chapter, we present a word embedding model that learns cross-sentence dependency for improving end-to-end co-reference resolution (E2E-CR). While the traditional E2E-CR model generates word representations by running long short-term memory (LSTM) recurrent neural networks on each sentence of an input article or conversation separately, we propose linear sentence linking and attentional sentence linking models to learn cross-sentence dependency. Both sentence linking strategies enable the LSTMs to make use of valuable information from context sentences while calculating the representation of the current input word. With this approach, the LSTMs learn word embeddings considering knowledge not only from the current sentence but also from the entire input document. Experiments show that learning cross-sentence dependency enriches information contained by the word representations, and improves the performance of the co-reference resolution model compared with our baseline.

Co-reference resolution requires models to cluster mentions that refer to the same physical entities. The models based on neural networks typically require different levels of semantic representations of input sentences. The models usually need to

calculate the representations of word spans, or mentions, given pre-trained character and word-level embeddings [Turian et al., 2010, Pennington et al., 2014] before predicting antecedents. The mention-level embeddings are used to make co-reference decisions, typically by scoring mention pairs and making links [Lee et al., 2017, Clark and Manning, 2016a, Wiseman et al., 2016]. Long short-term memories (LSTMs) are often used to encode the syntactic and semantic information of input sentences.

Articles and conversations include more than one sentences. Considering the accuracy and efficiency of co-reference resolution models, the encoder LSTM usually processes input sentences separately as a batch [Lee et al., 2017]. The disadvantage of this method is that the models do not consider the dependency among words from different sentences, which plays a significant role in word representation learning and co-reference predicting. For example, pronouns are often linked to entities mentioned in other sentences, while their initial word vectors lack dependency information. As a result, a word representation model cannot learn an informative embedding of a pronoun without considering cross-sentence dependency in this case.

It is also problematic if we encode the input document considering cross-sentence dependency and treat the entire document as one sentence. An input article or conversation can be too long for a single LSTM cell to memorize. If the LSTM updates itself for too many steps, gradients will vanish or explode [Pascanu et al., 2013], and the co-reference resolution model will be very difficult to optimize. Regarding the entire input corpus as one sequence instead of a batch also significantly increases the time complexity of the model.

To solve the problem that traditional LSTM encoders, which treat the input sentences as a batch, lack an ability to capture cross-sentence dependency, and to avoid the time complexity and difficulties of training the model concatenating all input sentences, we propose a cross-sentence encoder for end-to-end co-reference (E2E-CR). Borrowing the idea of an external memory module from [Sukhbaatar et al., 2015], an external memory block containing syntactic and semantic information from context sentences is added to the standard LSTM model. With this context memory block, the proposed model is able to encode input sentences as a batch, and also calculate the

representations of input words by taking both target sentences and context sentences into consideration. Experiments showed that this approach improved the performance of co-reference resolution models.

5.2 Related Work

5.2.1 Co-reference Resolution

A popular method of co-reference resolution is mention ranking [Durrett and Klein, 2013]. Reading each mention, the model calculates co-reference scores for all antecedent mentions, and picks the mention with the highest positive score to be its co-reference. Many recent works are based on this approach. [Durrett and Klein, 2013] designed a set of feature templates to improve the mention-ranking model. [Peng et al., 2015] proposed a mention-ranking model by jointly learning mention heads and co-references. [Clark and Manning, 2016a] proposed a reinforcement learning framework for the mention ranking approach. Based on similar ideas but without using parsing features, the authors of [Lee et al., 2017] proposed the current state-of-the-art model which uses neural networks to embed mentions and calculate mention and antecedent scores. [Lee et al., 2018] applied ELMo embeddings [Peters et al., 2018] to improve within-sentence dependency modeling and word representation learning. [Wiseman et al., 2016] and [Clark and Manning, 2016b] proposed models using global entity-level features.

5.2.2 Language Representation Learning

Distributed word embeddings have been used as the basic unit of language representation for over a decade [Bengio et al., 2003]. Pre-trained word embeddings, for example GloVe [Pennington et al., 2014] and Skip-Gram [Mikolov et al., 2013] are widely used as the input of natural language processing models.

Long short-term memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] are widely used for sentence modeling. A single-layer LSTM network was applied in the previous state-of-the-art co-reference model [Lee et al., 2017] to generate word

and mention representations. To capture longer distance dependencies , [**?**] proposed a recurrent model that outputs hidden states by skipping input tokens.

Recently, memory networks [Sukhbaatar et al., 2015] have been applied in language modeling [Cheng et al., 2016, Tran et al., 2016]. Applying an attention mechanism on memory cells, memory networks allow the model to focus on significant words or segments for classification and generation tasks. Previous works have shown that applying memory blocks in LSTMs also improves long-distance dependency extraction [Yogatama et al., 2018].

5.3 Learning Cross-Sentence dependency

To improve the word representation learning model for better co-reference resolution performance, we propose two word representation models that learn cross-sentence dependency.

5.3.1 Linear Sentence Linking

Instead of treating the entire input document as separate sentences and encode the sentences as a batch with an LSTM, the most direct way to consider cross-sentence dependency is to initialize LSTM states with the encodings of adjacent sentences. We name this method linear sentence linking (LSL).

In LSL, we encode input sentences with a 2-layer bidirectional LSTM. Give input sentences $[s_1, s_2 \dots s_n]$, the outputs of the first layer are $[[\overrightarrow{s}_1; \overleftarrow{s}_1], [\overrightarrow{s}_2; \overleftarrow{s}_2], \dots [\overrightarrow{s}_n; \overleftarrow{s}_n]]$. In the second LSTM layer, the initial state of the forward LSTM of s_i is initialized as

$$\overrightarrow{S}_i = [\overrightarrow{c}^2_0; [\overrightarrow{s}_{i-1}; \overleftarrow{s}_{i-1}]]$$

while the backward state is initialized as

$$\overleftarrow{S}_i = [\overleftarrow{c}^2_0; [\overrightarrow{s}_{i-1}; \overleftarrow{s}_{i-1}]]$$

where c^i_0 stands for the initial cell of the i-th layer, and x stands for the final

output of the LSTMs in first layer. We then concatenate the outputs of the forward and backward LSTMs in the second layer as the word representations for co-reference prediction.

5.3.2 Attentional Sentence Linking

It is difficult for LSTMs to embed enough information about a long sentence into a low-dimensional distributed vector. To collect richer knowledge from neighbor sentences, we propose a long short-term recurrent memory module and an attention mechanism to improve sentence linking.

To describe the architecture of the proposed model, we focus on adjacent input sentences s_{i-1} and s_i. We present the input embeddings of the j-th word in the i-th sentence with $x_{i,j}$.

Long Short-Term Memory RNNs

To solve the traditional recurrent neural networks, [Hochreiter and Schmidhuber, 1997] proposed the LSTM architecture. The detail of recurrent state updating in LSTMs $h_t = f_{lstm}(x_t, h_{t-1}, c_{t-1})$ is shown in section 2.2.4.

where x_t is the input embedding and h_t is the output representation of the t-th word.

LSTMs with Cross-Sentence Attention

We design an LSTM module with cross-sentence attention for capturing cross-sentence dependency. We name this method attentional sentence linking (ASL). Considering input word $x_{i,t}$ in the i-th sentence and all words from the previous sentence $X_{i-1} = [x_{i-1,1}, x_{i-1,2}, \ldots, x_{i-1,m}]$, we regard the matrix X_{i-1} as an external memory module and calculate an attention on its cells, where each cell contains a word embedding.

$$\alpha_j = \frac{e^{c_j}}{\sum_k e^{c_k}} \tag{5.1}$$

$$c_k = f_c([x_{i,t}; h_{t-1}; x_{i-1,k}]^T) \tag{5.2}$$

With the attention distribution $\boldsymbol{\alpha}$, we can get a vector summarizing related information from s_{i-1},

$$v_{i-1} = \sum_j \alpha_j \cdot x_{i-1,j} \tag{5.3}$$

The model decides if it needs to pay more attention on the current input or cross-sentence information with a context gate.

$$g_t = \sigma(f_g([x_{i,t}; h_{t-1}; v_{i-1}]^T)) \tag{5.4}$$

$$\hat{x}_{i,t} = g_t \cdot x_{i,t} + (1 - g_t) \cdot v_{i-1} \tag{5.5}$$

$\sigma(\cdot)$ stands for the Sigmoid function. The word representation of the target word is calculated as

$$h_{i,t} = f_{lstm}(\hat{x}_{i,t}, h_{i,t-1}, c_{i,t-1}) \tag{5.6}$$

where f_{lstm} stands for standard LSTM update described in section 5.3.2.

5.3.3 Co-reference Prediction

In this work, we apply the mention-ranking end-to-end co-reference resolution (E2E-CR) model proposed by [Lee et al., 2017] for co-reference prediction. The word representations applied in E2E-CR model is formed by concatenating pre-trained word embeddings and the outputs of LSTMs. In our work, we represent words by concatenating pre-trained word embeddings and the outputs of LSL- and ASL-LSTMs.

Models	MUC			B^3			Ceafe			Avg.
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	F1
[Wiseman et al., 2016]	77.5	69.8	73.4	66.8	57.0	61.5	62.1	53.9	57.7	64.2
[Clark and Manning, 2016b]	78.9	69.8	74.0	70.1	57.0	62.9	62.5	55.8	59.0	65.3
[Clark and Manning, 2016a]	79.2	70.4	74.6	69.9	58.0	63.4	63.5	55.5	59.2	65.7
[Lee et al., 2017]	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
E2E-CR + LSL	81.0	71.5	76.0	72.6	59.4	65.3	65.0	57.5	61.0	67.4
E2E-CR + ASL	79.2	73.7	**76.4**	69.4	62.1	**65.6**	64.0	58.9	**61.4**	**67.8**

Table 5.1: Experimental results of previous models and cross-sentence dependency learning models on the CoNLL-2012 shared task.

5.4 Experiments

We train and evaluate our model on the English corpus of the CoNLL-2012 shared task [Pradhan et al., 2012]. We implement our model based on the published implementation of the baseline E2E-CR model [Lee et al., 2017] [1]. Our implementation is also available online for reproducing the results reported in this thesis [2]. In this section, we first describe our hyperparameter setup, and then show the experimental results of previous work and our proposed models.

5.4.1 Model and Hyperparameter Setup

In practice, the LSTM modules applied in our model have 200 output units. In ASL, we calculate cross-sentence dependency using a multi-layer perceptron with one hidden layer consisting of 150 hidden units. The initial learning rate is set as 0.001 and decays 0.001% every 100 steps. The model is optimized with the Adam algorithm [Kingma and Ba, 2014]. We randomly select up to 40 continuous sentences for training if the input is too long. In co-reference prediction, we select 250 candidate antecedents as our baseline model.

5.4.2 Experiment Results and Discussion

We evaluate our model on the test set of the CoNLL-2012 shared task. The performance of previous work and our model are shown in Table 5.1. We mainly focus on the

[1] `https://github.com/kentonl/e2e-coref`
[2] `https://github.com/luohongyin/coatt-coref`

- I remember receiving **an SMS like this one** last year before it snowed since snowfall would affect road conditions in Beijing to a large extent. - Uh-huh . However, **it** did not give people such a special feeling as it did this time.
- **Reporters** are tired of the usual stand ups. - **They** want to be riding on a train or walking in the rain or something to get attention .
- Planned terrorist bombing that ripped a 20 x 40 - foot hole in the **Navy destroyer USS Cole** in the Yemeni port of Aden. - **The ship** was there for refueling.
- Yemeni authorities claimed they have detained **over 70 people** for questioning. - **These** include some Afghan - Arab volunteers.

Table 5.2: Examples predictions of the ASL model and the baseline model. The bold and underlined words in the second sentences are the target pronouns or references. The underlined phrases and words in the first sentences are the groundtruth co-references. Red colored words are the predictions of the baseline E2E-Coref model, while the green phrases and words are the predictions of our model. The examples showed that our model successfully captured the cross-sentence information.

average F1 score of MUC, B^3, and CEAF metrics. Comparing with the baseline model that achieved 67.2% F1 score, the ASL model improved the performance by 0.6% and achieved 67.8% average F1. Experiments show that the models that consider cross-sentence dependency significantly outperform the baseline model, which encodes each sentence from the input document separately.

Experiments also indicated that the ASL model has better performance than the LSL model, since it summarizes extracts context information with an attention mechanism instead of simply viewing sentence-level embeddings. This gives the model a better ability to model cross-sentence dependency.

Examples for comparing the performance of the ASL model and the baseline are shown in Table 2. Each example contains two continuous sentences with co-references distributed in different sentences. Underlined spans in bold are target mentions and

annotated co-references. Spans in green are ASL predictions, and spans in red are baseline predictions. A prediction on "-" means that no mention is predicted as a co-reference.

Table 2 shows that the baseline model, which does not consider cross-sentence dependency, has difficulty in learning the semantics of pronouns whose co-references are not in the same sentence. The pre-trained embeddings of pronouns are not informative enough. In the first example, "it" is not semantically similar with "SMS" in GloVe without any context, and in this case, "it" and "SMS" are in different sentences. As a result, if reading this two sentences separately, it is hard for the encoder to represent "it" with the semantics of "SMS". This difficulty makes the co-reference resolution model either prediction a wrong antecedent mention, or cannot find any co-reference.

However, with ASL, the model learns the semantics of pronouns with an attention to words in other sentences. With the proposed context gate, ASL takes knowledge from context sentences if local inputs are not informative enough. Based on word represents enhanced with cross-sentence dependency, the co-reference scoring model can make better predictions.

5.5 Conclusion and Future Work

We proposed linear and attentional sentence linking models for learning word representations that captures cross-sentence dependency. Experiments showed that the embeddings learned by proposed models successfully improved the performance of the state-of-the-art co-reference resolution model, indicating that cross-sentence dependency plays an important role in semantic learning in articles and conversations consists of multiple sentences. It worth exploring if our model can improve the performance of other natural language processing applications whose inputs contain multiple sentences, for example, reading comprehension, dialog generation, and sentiment analysis.

Chapter 6

Self-Attention Convolutional Neural Networks for Video Question Answering

6.1 Introduction

With the tremendous increase in new devices and machines, people are not always aware of the various features and functions of their devices or how to use new functions they never tried before. However, there are hundreds of video instructions over the Internet to help people understand how to use their devices and corresponding functions. Most people usually enter their questions as a query to a video search engine, e.g. Youtube, to search for an instruction. However, the search engines have some limitations. Firstly, they are not designed for answering question. Secondly, they retrieves entire videos sometimes with a long duration. Therefore, users need to manually search inside a video to find their desired answers.

To address those limitations, several models are proposed by previous research to find the users' interest points in videos. Unfortunately, these approaches are obstructed by another limitation: lack of labeled data, including manual annotation of video segments and moments. While deep neural networks with attention mechanisms can

Figure 6-1: An example of moment detection in a video for an input question "what does the auto function for air conditioner do?"

infer and extract such moments automatically in an unsupervised way, potentially better results can be achieved when having the target moments provided in advance, which enables supervised or semi-supervised training of the attention. This would allow not only more reliable video retrieval, but also better moment detection.

Following this idea, we have developed a corpus identifies the related videos and its segments to provide an accurate answer for an input question. In our corpus, each video introduces a set of devices and describes their aspects and functions. The videos also include instructions about how can users operate, or interact with the devices. We annotated the videos to manually split them into smaller segments, where each segment focuses on a single aspect or a single function for answering users' questions

more directly. For example, the question shown in Figure 6-1, "What does the auto function for air conditioner do?" can be clearly answered by a 30s long segment, instead of the entire video.

We develop two models–self-attention convolutional neural networks and memory neural networks [Mohtarami et al., 2018]–with our corpus for the video retrieval and moment detection tasks. The models encode input questions, videos and their segments into their embedding representations, and use attentions over the encoded representations to retrieve the best video and to detect the desired moment for answering the input question. In general, the experiments show that (*i*) the moment detection task is more challenging than the video retrieval task, and (*ii*) the models can significantly perform better if they use the labels for video segments/moments during training, and (*iii*) our models outperform the YouTube baseline.

6.2 Related Work

Our work and collected corpus have the following features:

- **Modality:** In our corpus, there are textual questions which their answers are in two modalities–videos and transcripts.
- **Integration:** In our corpus, there are videos that discuss a more general topic, e.g., introducing the air conditioner (AC) of a vehicle. These videos have different segment parts about more detailed sub-topics, e.g., how to turn on the AC or how to adjust it. The former is related to video retrieval and the latter is related to moment detection task. This enable us to integrate both video retrieval and moment detection tasks to find the exact answer of a given question.
- **Chain:** In our corpus, a video is highly focused on a certain topic with the segments that are usually about similar sub-topics and highly related to each other. This makes the moment detection task more challenging.

With respect to the above features, we compare our work with following previous work categories.

6.2.1 Video Retrieval.

[Jiang et al., 2007] presented a model with rich hand-crafted features to retrieve the related videos for a given query. [Xu et al., 2015b] proposed a model that jointly learns video and language embeddings for better retrieval task. [Araujo and Girod, 2018] showed that videos can be retrieved with image queries. While, instead of merely using visual inputs, [Yang and Meinel, 2014] used transcripts to improve the performance of video retrieval. The works in this category aims to retrieve the videos related to a specific query without considering which segment of the video is the exact answer to the given query (i.e., moment detection). Thus, our work is different from this category in terms of the *Integration* and *Chain* features of our work.

6.2.2 Visual/Video Question Answering

[Lin et al., 2014] and [Antol et al., 2015] worked on the Visual Questions Answering (VQA) task and respectively presented `MSCOCO` and `VQA` datasets focused on answering questions about scene understanding. [Das et al., 2017] proposed a multi-turn visual question corpus. While the VQA is developed for images, our work focuses on videos. In video QA, [Rohrbach et al., 2015] presented the `MPII-MD` dataset that contains movies and their descriptions. [Tapaswi et al., 2016] presented the `MovieQA` dataset which contains collected movies, subtitles, stories, questions, and candidate textual answers for multiple choice questions. The answers could be generated or selected from the textual candidates. In contrast of their works, we aim to retrieve the videos that include answers for a given question and then detect the moments of the retrieved videos that provide the best answers. Thus, our work is different in terms of the *Integration* and *Chain* features.

6.2.3 Community Question Answering

Given a Community Question Answering (cQA) thread containing a question and a list of answers, the works in this category aim to automatically rank the answers according to their relevance to the question [Mohtarami et al., 2016, Màrquez et al., 2015,

Belinkov et al., 2015, Nakov et al., 2016]. The answers may have some relations and in discussion with each other [Barrón-Cedeno et al., 2015, Joty et al., 2015], but in general they are written by different users and are mostly independent. Thus, this is different from the *Chain* feature of our work. Furthermore, our work is different in terms of the *Integration* and *Chain* features.

6.3 The Corpus

Our corpus contains videos and their transcripts, where each video is divided into several segments (i.e. video clips) and each segment is annotated with a set of questions. Overall, the process of corpus creation has several stages: (*i*) video extraction, (*ii*) video segmentation, and (*iii*) question annotation, which we describe below.

6.3.1 Video Extraction

We consider a YouTube channel–Ford Motor Company [1]–as the source of our videos. This channel contains the *How-To* videos that introduce a set of functions on vehicles, e.g., "*How to Check Your Tires with the Penny Test?*" We collected all its 107 *How-To* videos, transcribed them as a part of this corpus. The statistics of the videos and transcripts, e.g., the lengths of videos and transcripts, the vocabulary size, are presented in rows 1–9 of Table 6.1.

6.3.2 Video Segmentation

Following our aim of detecting the moment of a video with respect to a given question, we split each video into segments based on its transcript. Each segment includes one or more complete sentences and can be used to answer the How-to questions about a specific topic. For example, if a video is about the air conditioner (AC) system of a vehicle, a segment might introduce how to turn on the AC or the function of the "AUTO" button on the panel. The annotators also provide questions based on a single

[1] `https://www.youtube.com/user/ford`

	Videos	Segments
Videos		
1. Num. of Videos	107	464
2. Avg. Num. of seg.	4.34	-
3. Total Length (sec)	9,605.35	-
4. Avg. of Length (sec)	89.77	20.70
5. Min. of Length (sec)	11.45	4.13
6. Max. of Length (sec)	292.87	104.3
Transcripts		
7. Avg. Num. of Words	264.50	60.99
8. Total Num. of Words	28,301	28,301
9. Vocab. Size	2,489	2,489
Questions		
10. Num. of Questions	-	9,482
11. Num. of Ques./seg.	-	20.44
12. Avg. Num. of Words	-	9.32
13. Vocab. Size	-	3,329

Table 6.1: The statistics of the collected videos, transcripts, and questions.

video segment and its transcript instead of watching the entire video. The statistics of the segments are shown in Table 6.1.

6.3.3 Question Annotation

We have used Amazon Mechanical Turk[2] (AMT) to collect questions for the video segments. Each video segment was assigned to 10 to 12 annotators, who were asked to enter two different questions that are answered by the content of the given video segment. Note that only the videos are shown to the annotators without their transcripts to avoid using the exact words for the transcripts in the questions. In total, we collected around 10K questions. The corpus and the implementation of our models are publicly available [3]. Rows 10–13 in Table 6.1 show the statistics of the collected questions, and an example of the questions is shown in Figure 6-1.

[2] `https://www.mturk.com/`

[3] `https://github.com/luohongyin/VehicleVQA`

6.4 Models

In this work, we evaluate three models for video retrieval and moment detection tasks using our corpus; YouTube video retrieval search engine[4], self-attention convolutional neural networks (SACNN) and memory neural networks (MemNN), which are explained below.

6.4.1 YouTube

We use the YouTube API [5] as a baseline model. Given a question as a query to the API, it attempts to find the related videos to the question from the specific channel used for our corpus as explained in Section6.3. Then, it retrieves a ranked set of videos, and this set is used to evaluate the performance of YouTube.

6.4.2 Memory Networks

A memory network (MemNN) model is proposed in [Mohtarami et al., 2018] for stance detection task with the capability of extracting rationales from documents with respect to given claims. In this work, we investigate the model for the video retrieval and moment detection tasks. We give a question and a video including all its segments to the model and it outputs a score for the video and a set of scores corresponding to the video segments–as rationales–that indicate the relatedness of the video and its segments to the input question. In this work, we employ the same MemNN architecture proposed in [Mohtarami et al., 2018]. The architecture of the model is shown in Figure 6-2 [Sukhbaatar et al., 2015].

We train a single MemNN for both video retrieval and moment detection. The only supervision signal we used for the training is the video-level labels. Thus, the moment detection task is a semi-supervised task for MemNN. In this work, we use the final output of the MemNN to find the best video and use the attention weights over segments to find the best segment.

[4] `https://www.youtube.com/`

[5] `https://www.youtube.com/yt/dev/api-resources/`

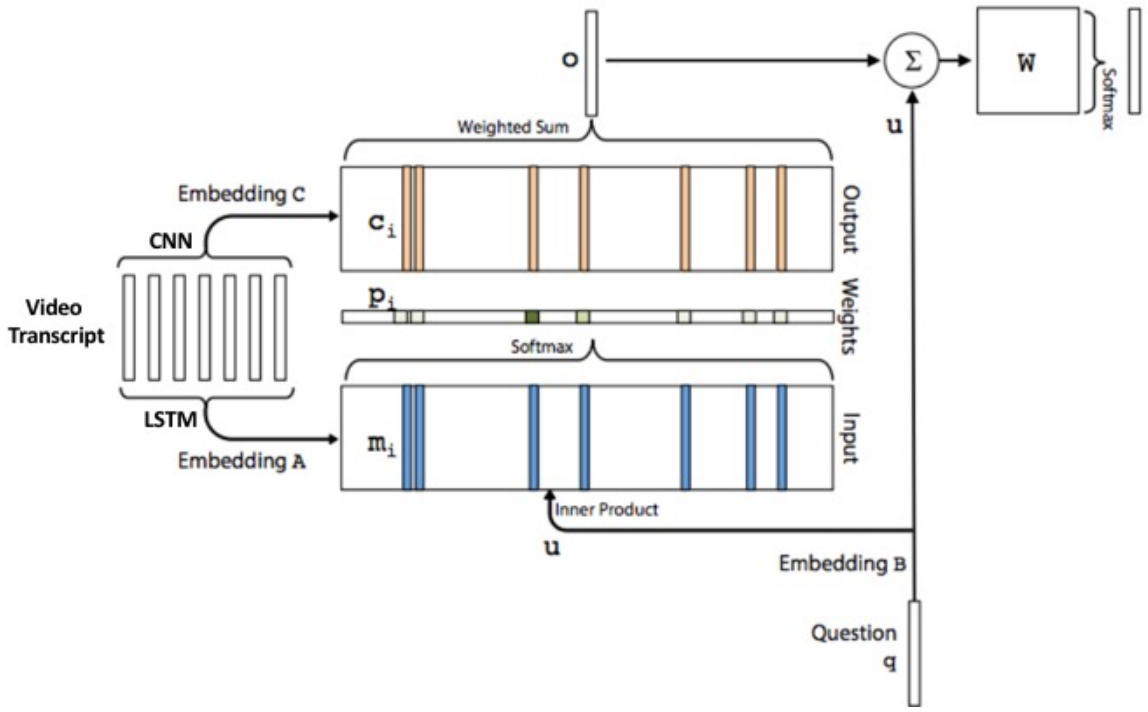


Figure 6-2: The structure of the applied MemNN model [Sukhbaatar et al., 2015].

6.4.3 SACNNs

We also present the SACNN model for our tasks. We apply a convolutional neural network (CNN) as the first step to encode the words and their contexts to their embedding representations. Upon the CNN layers, we apply a self-attention mechanism [Cheng et al., 2016, Tran et al., 2016, Vaswani et al., 2017] as a pooling layer to generate fixed-length embeddings for input questions, videos, and video segments. These embeddings are used to calculate the cosine similarity between the input questions with the videos or their segments. Then, these scores are used to select the top N videos as possible answers. The details of the model is shown in Figure 6-3.

Figure 6-3: The architecture of the SACNN model. The blue blocks stand for word-level and sentence-level distributed embeddings. The red blocks stand for the attention weights assigned to each word. The sentence embedding is calculated by averaging all word embeddings with the attention distribution.

In practice, we use a two-layer CNN with ReLU activation function for the hidden layer. The size of the convolution window is 5, for both layers. The output of the

CNN is consist of two parts - an attention score and a embedding vector. The feed forward process of the SACNN is shown as follow,

$$H = ReLU(W_h * X + b_h) \tag{6.1}$$

$$\hat{E}, Y = W_y * H + b_y \tag{6.2}$$

$$E = ReLU(\hat{E}) \tag{6.3}$$

where $E = [e_1, e_2, \ldots, e_n]$ stands for the context embeddings output by the CNN module. $Y = [y_1, y_2, \ldots, y_n]$ is a vector of scores. We then calculate an attention distribution with the scores Y. The attention of the i-th word, α_i, is

$$\alpha_i = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{6.4}$$

With the attention distribution over the input sentence (question or transcript), the final sentence embedding is calculated as

$$s^t = \sum_i \alpha_i \cdot e_i \tag{6.5}$$

where $t \in \{Q, T\}$, standing for either question and transcript. In practice, the question and transcript encoders do not share parameters.

6.4.4 Training

In this work, we train our models with negative sampling. For each question-transcript pair, we randomly select 15 negative transcripts for the input question. With the question embedding q and a series of transcript embedding $[t^p, t_1^n, t_2^n, \ldots, t_m^n]$, where t^p stands for the positive sample and t_i^n are negative samples, we calculate the probability that a transcript is the answer to a question by

$$p(t_i|q) = \frac{e^{t_i \cdot q}}{\sum_j e^{t_j \cdot q}} \tag{6.6}$$

Then we update the parameters in the model with stochastic gradient descent (SGD) based on cross-entropy losses,

$$l = -\sum_{i}(y_i \cdot log(p(t_i|q)) + (1 - y_i) \cdot log(1 - p(t_i|q))) \tag{6.7}$$

where y_i is 1 for the positive samples and 0 for negative samples.

6.5 Experiments and Evaluation

We evaluate the models explained in Section 6.4 with our corpus on the video retrieval and moment detection tasks. We apply 10-fold cross-validation to evaluate the performance of our models, where all questions for the same video are assigned to the same fold. We report mean averaged precision (MAP) at $N = \{1, 5, 10\}$, which is the standard evaluation metric of ranking and retrieval tasks. We do not report the MAP@10 score of the local moment detection task, since many videos contain less than 10 segments. In our published corpus, we also split the folds as same as we used in our experiments.

6.5.1 Video Retrieval

The experimental results for the video retrieval task is shown in Table 6.2 (rows 1–3). The results indicate that the YouTube API performs not well (row 1), since traditional video search engines are not specially designed for retrieving information based on questions. This proves that designing a special video question answering system for video instruction retrieval is necessary.

Both neural models significantly outperform the YouTube baseline. The MemNN and SACNN can achieve high performance, in particular for MAP@5 and MAP@10. The reason is that they encode the videos transcriptions using embedding representations through attention mechanism. With this mechanism, the models can highlight vehicle-related words and terms, which are more important and informative for the video retrieval model to make the decision. The MemNN relatively performs better

than SACNN, Because it applies higher-level attentions over video segments, helping the model to learn better representation of the entire videos.

6.5.2 Local Moment Detection

In this task, we assume the model is given a *related* video to an input question, and the model aims to retrieve the moment in the given video that makes the best answer to the question. The experimental results are shown in Table 6.2 (rows 4–5). We do not report the performance of Youtube search engine because it cannot perform moment detection.

Experimental results show that the SACNN leads to significantly better performance than MemNN, in particular for MAP@1. The reason is that SACNN uses the labels for video segments during training, While MemNN uses only the video labels–not segment labels–during training. Although not using explicit segment-level labels, the MemNN model still achieved high MAP@5 accuracy. This suggests that the segmental attention of the MemNN successfully captured some of the moments that directly answer the questions.

However, in real-life situations, the video question answering system is not provided with the groundtruth related video as the settings in the local moment detection task. Thus the global moment detection task is more important.

6.5.3 Global Moment Detection

To align our experiments better with the real-life application, we propose the global moment detection task. In this task, we relax our assumption for the local moment detection (described above), where the model is given all *related* and *unrelated* videos for an input question, and is asked to retrieve the best moment from the entire given set of videos.

The experimental results are shown in Table 6.2 (rows 6–7). The retrieving performances of both models are lower than the local moment detection task, indicating that the problem becomes more difficult when considering all videos, which significantly

	MAP@1	MAP@5	MAP@10
Video Retrieval			
1. YouTube	36.54	56.24	-
2. MemNN	65.02	**90.36**	**93.91**
3. SACNN	**66.69**	87.42	91.09
Local Moment Detection			
4. MemNN	37.38	80.17	-
5. SACNN	**77.94**	**97.65**	-
Global Moment Detection			
6. MemNN	24.53	54.66	75.14
7. SACNN	**57.13**	**80.75**	**85.20**

Table 6.2: Experimental results of SACNN and MemNN models, and YouTube baseline for video retrieval and moment detection tasks. The experimental results show that our method significantly outperformed the Youtube baseline in the video retrieval task. The proposed models also perform well on both moment detection tasks.

enlarged the search space of the question-answering model.

With segment-level supervision, the SACNN model achieved higher performance than MemNN. The model is able to successfully hit the moment with the best answer in its top-5 choices in around 4 out of 5 test cases. In addition, although the MemNN model for global moment detective is semi-supervised, it still reached good MAP@10 performance. This indicates that the MemNN learns to retrieve the best video by paying attention to the most related video segments.

6.6 Conclusion and Future Work

We have described a novel corpus that unifies video retrieval and moment detection tasks. This is the first corpus to offer such a combination. We further developed a self-attention convolutional neural network and a memory network model, and evaluated them and the YouTube video search engine on our corpus. The results showed that the neural models can achieve better performance compared to the YouTube baseline. In future work, we plan to extend the annotations to cover other domains, other modalities such as spoken language, and other important aspects of video and moment retrieval

such as personalized retrieval using the personal interests of users, which have been shown useful in previous research [Wang et al., 2017, He et al., 2017, Cao et al., 2017].

Chapter 7

Conclusions

This thesis proposed different attention mechanisms for different natural language tasks. Experimental results have shown the effectiveness of our proposed models.

7.1 Summary

In Chapter 3, we simulated an LSTM architecture with CNNs and self attentions. with this method, the model can process natural languages in parallel, while still achieving comparable performance with the standard LSTMs.

In Chapter 4, we proposed a phrase-aware word-level language model. We employed syntactic heights for phrase inducing and generated phrase embeddings with attentions calculated with the heights. This method improved several strong baselines, such as AWD-LSTM and Transformer-XL. We also found that the learned induced phrases are interpretable.

In Chapter 5, we improved the state-of-the-art end-to-end co-reference resolution (E2E-Coref) model by integrating cross-sentence information to the LSTM processing the target sentence. The model is realized with an cross-sentence attention model.

In Chapter 6, we constructed a video question answering corpus. We solved both video retrieval and moment detection tasks with memory networks and self-attention convolutional neural networks. Our methods significantly outperformed the Youtube video retrieval baseline.

7.2 Contributions

This thesis proposed different attention mechanisms for different natural language tasks. Experimental results have shown the effectiveness of our proposed models. Experiments on different tasks proved that the proposed attention mechanisms can improve different levels of language representations. Cross-sentence attention improved contextual word embeddings for co-reference recognition. Context attention in GTCNs improved context representation for language modeling. The headword attention for following phrases helps the language model to induce the following phrase and learn syntax without any supervision. Self-attention improved sentence embeddings for video question answering.

7.3 Future Work

The proposed methods in this thesis can improve many other models since they learn the natural language better than current models, as suggested by the experimental results. In future work, we will apply the proposed attention mechanisms in different problems and applications in the field of natural langauge processing, including machine translation, reading comprehension, question answering, and information extraction. We will also explore more effective method for improving language modeling and representation.

Bibliography

[Antol et al., 2015] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., and Parikh, D. (2015). VQA: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

[Araujo and Girod, 2018] Araujo, A. and Girod, B. (2018). Large-scale video retrieval using image queries. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6):1406–1420.

[Baevski and Auli, 2018] Baevski, A. and Auli, M. (2018). Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bai et al., 2018] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

[Barrón-Cedeno et al., 2015] Barrón-Cedeno, A., Filice, S., Da San Martino, G., Joty, S., Marquez, L., Nakov, P., and Moschitti, A. (2015). Threadlevel information for comment classification in community question answering. In *Proceedings of the ACL-IJCNLP*, volume 15, pages 687–693.

[Belinkov et al., 2015] Belinkov, Y., Mohtarami, M., Cyphers, S., and Glass, J. (2015). VectorSLU: A continuous word vector approach to answer selection in community question answering systems. *SemEval-2015*, page 282.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[Bradbury et al., 2016] Bradbury, J., Merity, S., Xiong, C., and Socher, R. (2016). Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*.

[Campos et al., 2017] Campos, V., Jou, B., Giró-i Nieto, X., Torres, J., and Chang, S.-F. (2017). Skip RNN: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*.

[Cao et al., 2017] Cao, D., Nie, L., He, X., Wei, X., Zhu, S., and Chua, T.-S. (2017). Embedding factorization models for jointly recommending items and user generated lists. In *Proceedings of the SIGIR*, pages 585–594, New York, NY, USA.

[Cheng et al., 2016] Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[Clark and Manning, 2016a] Clark, K. and Manning, C. D. (2016a). Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*.

[Clark and Manning, 2016b] Clark, K. and Manning, C. D. (2016b). Improving coreference resolution by learning entity-level distributed representations. *arXiv preprint arXiv:1606.01323*.

[Dai et al., 2019] Dai, Z., Yang, Z., Yang, Y., Cohen, W. W., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

[Das et al., 2017] Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J. M., Parikh, D., and Batra, D. (2017). Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Dauphin et al., 2016] Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2016). Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

[Dauphin et al., 2017] Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org.

[Durrett and Klein, 2013] Durrett, G. and Klein, D. (2013). Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982.

[Dyer et al., 2016] Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016). Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

[Gao et al., 2018] Gao, F., Wu, L., Zhao, L., Qin, T., Cheng, X., and Liu, T.-Y. (2018). Efficient sequence learning with group recurrent networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 799–808.

[Garcia and Bruna, 2017] Garcia, V. and Bruna, J. (2017). Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*.

[Gilmer et al., 2017] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org.

[Grave et al., 2016] Grave, E., Joulin, A., and Usunier, N. (2016). Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

[He et al., 2017] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182.

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

[Inan et al., 2016] Inan, H., Khosravi, K., and Socher, R. (2016). Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

[Jiang et al., 2007] Jiang, Y.-G., Ngo, C.-W., and Yang, J. (2007). Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 494–501. ACM.

[Johnson, 2016] Johnson, D. D. (2016). Learning graphical state transitions. *https://openreview.net/pdf?id=HJ0NvFzxl*.

[Joty et al., 2015] Joty, S., Barrón-Cedeno, A., Da San Martino, G., Filice, S., Marquez, L., Moschitti, A., and Nakov, P. (2015). Global thread-level inference for comment classification in community question answering. In *Proceedings of the EMNLP*, volume 15.

[Jurafsky and Martin, 2014] Jurafsky, D. and Martin, J. H. (2014). *Speech and language processing*, volume 3. Pearson London.

[Kim et al., 2016] Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *Proceedings of ICLR*.

[Kuncoro et al., 2018] Kuncoro, A., Dyer, C., Hale, J., Yogatama, D., Clark, S., and Blunsom, P. (2018). Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

[Lee et al., 2017] Lee, K., He, L., Lewis, M., and Zettlemoyer, L. (2017). End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

[Lee et al., 2018] Lee, K., He, L., and Zettlemoyer, L. (2018). Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*.

[Lee et al., 2016] Lee, K., Salant, S., Kwiatkowski, T., Parikh, A., Das, D., and Berant, J. (2016). Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.

[Levy and Goldberg, 2014] Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

[Li et al., 2018] Li, S., Li, W., Cook, C., Zhu, C., and Gao, Y. (2018). Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5457–5466.

[Li et al., 2016] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2016). Gated graph sequence neural networks. *Proceedings of ICLR*.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

[Liu et al., 2018] Liu, H., Simonyan, K., and Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

[Marcheggiani and Titov, 2017] Marcheggiani, D. and Titov, I. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings EMNLP*.

[Màrquez et al., 2015] Màrquez, L., Glass, J., Magdy, W., Moschitti, A., Nakov, P., and Randeree, B. (2015). SemEval-2015 Task 3: Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation*.

[Melis et al., 2018] Melis, G., Blundell, C., Kočiskỳ, T., Hermann, K. M., Dyer, C., and Blunsom, P. (2018). Pushing the bounds of dropout. *arXiv preprint arXiv:1805.09208*.

[Melis et al., 2017] Melis, G., Dyer, C., and Blunsom, P. (2017). On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*.

[Merity et al., 2017] Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

[Merity et al., 2018] Merity, S., Keskar, N. S., and Socher, R. (2018). An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

[Merity et al., 2016] Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

[Mohtarami et al., 2018] Mohtarami, M., Baly, R., Glass, J., Nakov, P., Màrquez, L., and Moschitti, A. (2018). Automatic stance detection using end-to-end memory networks. In *Proceedings of the NAACL*, NAACL-HLT '18, New Orleans, LA, USA.

[Mohtarami et al., 2016] Mohtarami, M., Belinkov, Y., Hsu, W.-N., Zhang, Y., Lei, T., Bar, K., Cyphers, S., and Glass, J. (2016). SLS at semeval-2016 task 3: Neural-based approaches for ranking in community question answering. In *Proceedings of NAACL-HLT Workshop on Semantic Evaluation*, pages 753–760, San Diego, California. Association for Computational Linguistics.

[Nakov et al., 2016] Nakov, P., Màrquez, L., Magdy, W., Moschitti, A., Glass, J., and Randeree, B. (2016). SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California. Association for Computational Linguistics.

[Olah, 2015] Olah, C. (2015). Understanding lstm networks.

[Pascanu et al., 2013] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

[Peng et al., 2015] Peng, H., Chang, K.-W., and Roth, D. (2015). A joint framework for coreference resolution and mention head detection. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 12–21.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[Pham et al., 2018] Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. (2018). Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

[Polyak and Juditsky, 1992] Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

[Pradhan et al., 2012] Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., and Zhang, Y. (2012). Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

[Rae et al., 2018] Rae, J. W., Dyer, C., Dayan, P., and Lillicrap, T. P. (2018). Fast parametric learning with activation memorization. *arXiv preprint arXiv:1803.10049*.

[Rohrbach et al., 2015] Rohrbach, A., Rohrbach, M., Tandon, N., and Schiele, B. (2015). A dataset for movie description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3202–3212.

[Scarselli et al., 2009] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, pages 61–80.

[Schlichtkrull et al., 2017] Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., and Welling, M. (2017). Modeling relational data with graph convolutional networks. *arXiv preprint arXiv:1703.06103*.

[Seo et al., 2017] Seo, M., Min, S., Farhadi, A., and Hajishirzi, H. (2017). Neural speed reading via skim-rnn. *arXiv preprint arXiv:1711.02085*.

[Seo et al., 2016] Seo, Y., Defferrard, M., Vandergheynst, P., and Bresson, X. (2016). Structured sequence modeling with graph convolutional recurrent networks. *arXiv preprint arXiv:1612.07659*.

[Shaw et al., 2018] Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

[Shen et al., 2017] Shen, Y., Lin, Z., Huang, C.-W., and Courville, A. (2017). Neural language modeling by jointly learning syntax and lexicon. *arXiv preprint arXiv:1711.02013*.

[Shen et al., 2018a] Shen, Y., Lin, Z., Jacob, A. P., Sordoni, A., Courville, A., and Bengio, Y. (2018a). Straight to the tree: Constituency parsing with neural syntactic distance. *arXiv preprint arXiv:1806.04168*.

[Shen et al., 2018b] Shen, Y., Tan, S., Sordoni, A., and Courville, A. (2018b). Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv preprint arXiv:1810.09536*.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Sukhbaatar et al., 2015] Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2440–2448.

[Tapaswi et al., 2016] Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., and Fidler, S. (2016). MovieQA: Understanding Stories in Movies through Question-Answering. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[Tran et al., 2016] Tran, K., Bisazza, A., and Monz, C. (2016). Recurrent memory networks for language modeling. *arXiv preprint arXiv:1601.01272*.

[Turian et al., 2010] Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[Wang et al., 2017] Wang, X., Nie, L., Song, X., Zhang, D., and Chua, T.-S. (2017). Unifying virtual and physical worlds: Learning toward local and global consistency. *ACM Trans. Inf. Syst.*, 36(1):4:1–4:26.

[Wiseman et al., 2016] Wiseman, S., Rush, A. M., and Shieber, S. M. (2016). Learning global features for coreference resolution. *arXiv preprint arXiv:1604.03035*.

[Xu et al., 2015a] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015a). Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

[Xu et al., 2015b] Xu, R., Xiong, C., Chen, W., and Corso, J. J. (2015b). Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In *AAAI*, volume 5, page 6.

[Yang and Meinel, 2014] Yang, H. and Meinel, C. (2014). Content based lecture video retrieval using speech and video text information. *IEEE Transactions on Learning Technologies*, (2):142–154.

[Yang et al., 2017] Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. (2017). Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*.

[Yogatama et al., 2018] Yogatama, D., Miao, Y., Melis, G., Ling, W., Kuncoro, A., Dyer, C., and Blunsom, P. (2018). Memory architectures in recurrent neural network language models. In *International Conference on Learning Representations. https://openreview. net/forum*.

[Zaremba et al., 2014] Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

[Zhang et al., 2015] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

[Zilly et al., 2016] Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. (2016). Recurrent highway networks. *arXiv preprint arXiv:1607.03474*.

[Zilly et al., 2017] Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. (2017). Recurrent highway networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4189–4198. JMLR. org.