Span-based Neural Structured Prediction

Kenton Lee

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

Doctor of Philosophy

University of Washington

2017

Reading Committee:

Luke Zettlemoyer, Chair

Yejin Choi

Noah Smith

Mari Ostendorf

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Span-based Neural Structured Prediction

Kenton Lee

Chair of the Supervisory Committee:
Associate Professor Luke Zettlemoyer
Computer Science & Engineering

A long-standing goal in artificial intelligence is for machines to understand natural language. With ever-growing amounts of data in the world, it is crucial to automate many aspects of language understanding so that users can make sense of this data in the face of information overload. The main challenge stems from the fact that the surface form of language, either as speech or text, is unstructured. Without programmatic access to the semantics of natural language, it is challenging to build general, robust systems that are usable in practice.

Towards achieving this goal, we propose a series of neural structured-prediction algorithms for natural language processing. In particular, we address a challenge common to all such algorithms: the space of possible output structures can be extremely large, and inference in this space can be intractable. Despite the seeming incompatibility of neural representations with dynamic programs from traditional structured prediction algorithms, we can leverage these rich representations to learn more accurate models while using simpler or lazier inference.

We focus on algorithms that model the most basic substructure of language: spans of text. We present state-of-the-art models for tasks that require modeling the internal structure of spans, such as syntactic parsing, and modeling structure between spans, such as question answering and coreference resolution. The proposed techniques are applicable to many problems, and we expect that they will further push the limits of neural structured prediction for natural language processing.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

My advisor, Luke Zettlemoyer, has provided an incredible learning experience throughout my journey as a PhD student. In addition to the technical research knowledge bestowed upon me, his general wisdom of how to conduct scientific research, overcome research challenges, and effectively communicate ideas has been invaluable.

I have had the fortune of working with many brilliant mentors, including Anca Dragan, Constantine Lignos, Yoav Artzi, Mike Lewis, and Omer Levy, all of whom have significantly impacted my research and broadened my horizons. Additionally, I have enjoyed and learned a great deal from countless discussions with many members of UW-NLP.

Lastly, I am grateful to Mitch Marcus for introducing me to the wonderful world of research.

DEDICATION

To my family

Chapter 1

INTRODUCTION

A long-standing goal in artificial intelligence is for machines to understand natural language. With ever-growing amounts of data in the world, it is crucial to automate many aspects of language understanding so that users can make sense of this data in the face of information overload. Much of today's data is in the form of natural language, including news articles, social media, email, broadcast news, encyclopedic articles, chat logs etc. In order for users to interact meaningfully with this information, a system must extract and reason about the underlying meaning of the text.

A key challenge in building such systems is that the surface form of language, either as speech or text, is unstructured. Without programmatic access to the semantic structure of language, we can only perform shallow reasoning, resulting in systems that reduce to little more than pattern matching. Therefore, much of the effort natural language processing (NLP) has been devoted towards recovering structured representations of language. Some structures are linguistically motivated, such as syntax trees, semantic frames, coreference clusters, and discourse structures. Other structures are more task oriented, such as logical forms, knowledge bases, and summaries.

Towards achieving this goal, this thesis presents a series of neural structured-prediction algorithms for natural language processing. In particular, we address a challenge common to all such algorithms: the space of possible output structures can be extremely large, and inference in this space can be intractable. Recent advances in deep learning have enable models to learn rich vector representations of language in context. These representations can implicitly capture much of the information traditionally modeled by expensive, explicit models of output structure. In the presented work, we leverage the expressivity of neural representations of language to make inference either lazier or simpler than previous approaches, while demonstrating state-of-the-art accuracy.

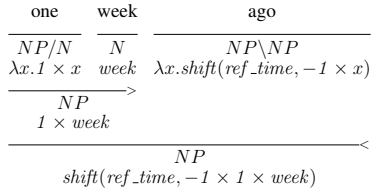
We focus on inference algorithms over the most basic substructure of language: spans of

text. We present state-of-the-art models for tasks that require modeling the internal structure of spans, such as syntactic parsing (Lee et al., 2016a), and modeling structure between spans, such as question-answering (Lee et al., 2016b) and coreference resolution (Lee et al., 2017). Together, these models illustrate the expressive power of general span representations and importance of decomposing models to enable efficient inference. The proposed techniques are applicable to many structured prediction problems and we expect that they will further push the limits of neural structured prediction for natural language processing.

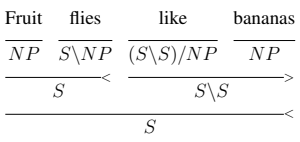
1.1 Background

In order to motivate the tasks that this thesis focuses on, we first provide some background on CCG parsing, question answering, and coreference resolution.

1.1.1 Combinatory Categorial Grammars



(a) CCG syntactic and semantic parse.



(b) CCG syntactic parse.

Figure 1.1: Examples of domain-specific and broad-coverage CCG parses.

Combinatory Categorial Grammar (CCG) is a linguistic categorial formalism for modeling a wide range of language phenomena (Steedman, 1996, 2000). CCG parsing can be used to recover the syntactic and/or semantic structure of language. Recent work on CCG parsing is largely divided into two popular settings: (1) domain-specific CCG parsing (Figure 1.1a) that jointly models syntax and semantics with respect to a target ontology (Zettlemoyer and Collins, 2005, 2009; Artzi and Zettlemoyer, 2013; Lee et al., 2014) , and (2) broad-coverage CCG parsing (Figure 1.1b) that

models syntax, where the semantics are in the form of coarse-grained dependencies that can be deterministically recovered from the syntax (Lewis and Steedman, 2014; Lewis et al., 2016; Lee et al., 2016a).

Chapter 2 focuses on the latter setting, with the goal of learning models that automatically recover the correct syntax tree given a sentence. We show for the first time, a parsing model can perform exact inference while modeling the global structure of the syntax tree, resulting in state-of-the-art accuracy (Lee et al., 2016a).

1.1.2 Question Answering

Question: What does the outside of the Tardis resemble?

Passage: Doctor Who is a British science-fiction television programme produced by the BBC since 1963. The programme depicts the adventures of the Doctor, a Time Lorda space and time-travelling humanoid alien. He explores the universe in his TARDIS, a sentient time-travelling space ship. Its exterior appears as **(a blue British police box)**, which was a common sight in Britain in 1963 when the series first aired. Accompanied by companions, the Doctor combats a variety of foes, while working to save civilisations and help people in need.

Answer: **a blue British police box**

Figure 1.2: Example of the extractive question answer task.

Question answering (QA) is a general problem that encompasses many of the challenges faced in NLP tasks. It is useful both as an end-application, enabling users to issue natural language queries to a machine, and as a benchmark for reading comprehension, enabling researchers to measure how well current systems understand language. Many variants of QA have been proposed, ranging from questions asking for a choice between a small set of given answers (Richardson et al., 2013a; Kembhavi et al., 2017), to questions asking for facts from a knowledge base (Zelle and

Mooney, 1996; Cai and Yates, 2013; Berant et al., 2013), to questions asking for answer snippets from documents (Voorhees, 2001; Rajpurkar et al., 2016; Joshi et al., 2017). In Chapter 3, we focus on the latter type of QA, where the problem can be posed as a miniature structured prediction problem of extracting the best answer span in the given document with respect to a question. An example of this extractive question answering task is depicted in Figure 1.2. We show that relatively straightforward architecture that builds neural representation of passage spans in the context of the question can achieve state-of-the-art performance (Lee et al., 2016b).

1.1.3 Coreference Resolution

We are looking for **(a region of central Italy bordering the Adriatic Sea)**. **(The area)** is mostly mountainous and includes Mt. Corno, the highest peak of the Apennines. **(It)** also includes a lot of sheep, good clean-living, healthy sheep, and an Italian entrepreneur has an idea about how to make a little money of them.

Figure 1.3: Example of a coreference cluster. Coreferent mentions are parenthesized and in bold.

In natural language, the ability recover the identity of a mentioned entity or concept is non-trivial. We often rely on indirect signals, such as pronouns or paraphrases to refer to previous spans of text, or mentions. In order to fully understand a narrative, particularly over long documents, we must first address the fundamental challenge of coreference resolution: finding clusters of mentions that refer to the same underlying entity. An example of such a coreference cluster is depicted in Figure 1.3. In Chapter 4, we demonstrate for the first time that a neural coreference resolution system can be learned end-to-end with state-of-the-art performance. In Chapter 5, we further extend this result by modeling higher-order structures while reducing the computational cost of inference.

1.2 Previous Work

There is a large body of work on structured prediction in NLP, and we refer the reader to Smith (2011) for a thorough survey of well-studied techniques. In this section, we focus primarily on structured prediction with deep learning, which has been a mixture of novel algorithms and non-linear generalizations of existing methods.

One of the successes of deep learning in NLP has been the development of sequence-to-sequence architectures (Bahdanau et al., 2014; Sutskever et al., 2014), which were first applied to machine translation. These architectures learn to map input sequences to output sequences. In theory, this architecture subsumes all of structured prediction, since any structure can be serialized and represented as a sequence. Indeed, sequence-to-sequence models have been applied to a large variety of structured prediction tasks, such as machine translation (Bahdanau et al., 2014; Sutskever et al., 2014), syntactic parsing (Vinyals et al., 2015b), semantic parsing (Dong and Lapata, 2016; Iyer et al., 2017), text generation (Konstas et al., 2017; Kiddon et al., 2016), and abstractive summarization (Rush et al., 2015). Sequence-to-sequence performs best when there is no a priori alignment between the input and output. This property holds in almost all of the above tasks (except syntactic parsing, where sequence-to-sequence is not a competitive approach).

In structured prediction problems where there is a clear alignment between inputs and outputs, typically found in linguistic structured prediction tasks, non-linear generalizations of traditional approaches achieve state-of-the-art performance. These approaches fall largely into two categories: transition-based approaches and graph-based approaches. Transition-based approaches define transition systems that, after taking a sequence of actions, results in a predicted output structure. These approaches are typically intractable and use greedy inference, but they can condition on large parts of the output structure, resulting in highly expressive models. Such transition-based models have been shown to be successful for syntactic parsing (Chen and Manning, 2014; Dyer et al., 2015; Andor et al., 2016; Dyer et al., 2016) and semantic parsing (Krishnamurthy et al., 2016; Swayamdipta et al., 2016; Misra and Artzi, 2016). In contrast, graph based approaches leverage the rich context-dependent neural representations to make very strong conditional independence

assumptions about the output structure, enabling fast and exact inference. This approach is popular for constituency parsing (Stern et al., 2017), dependency parsing (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016), sequence tagging (Lewis et al., 2016; Peters et al., 2017; He et al., 2017), and semantic dependency parsing (Peng et al., 2017). The methods proposed in this thesis largely fall under the second category. They are heavily inspired by traditional structured prediction techniques but redefined to leverage the expressiveness of deep learning techniques.

Relevant to the span representation learning techniques covered in this thesis is the large body of literature on learning vector representations of phrases or sentences. Popular approaches include bag-of-words models (Mitchell and Lapata, 2008), convolutional neural networks (Kalchbrenner et al., 2014), and LSTMs (Bowman et al., 2015). Perhaps most relevant to this thesis are examples that use these representations as part of a larger structured prediction problem (Cross and Huang, 2016; Kong et al., 2015; Lu et al., 2016; Stern et al., 2017).

1.3 Thesis Outline

In Chapter 2, we introduce a novel method for modeling the internal structure of spans recursively. We evaluate our method on CCG parsing, where for the first time we are able to achieve guarantees for global inference. This work showcases the ability of neural networks to perform highly complex, yet efficient inference by computing structured representations and model scores in a lazy manner.

In Chapter 3, we investigate structure prediction from the opposite side of the spectrum, where inference is almost trivial. We introduce span-based extractive question answering model, which sets up a useful abstraction of span representations for the following chapters.

In Chapters 4 and 5, we introduce an approach for coreference resolution that combines ideas for efficient structured inference from Chapter 2 with the highly expressive span representations from Chapter 3. We introduce the first state-of-the-art neural coreference resolvers that can be learned end to end, while being able to efficiently model higher-order structures.

Chapter 2

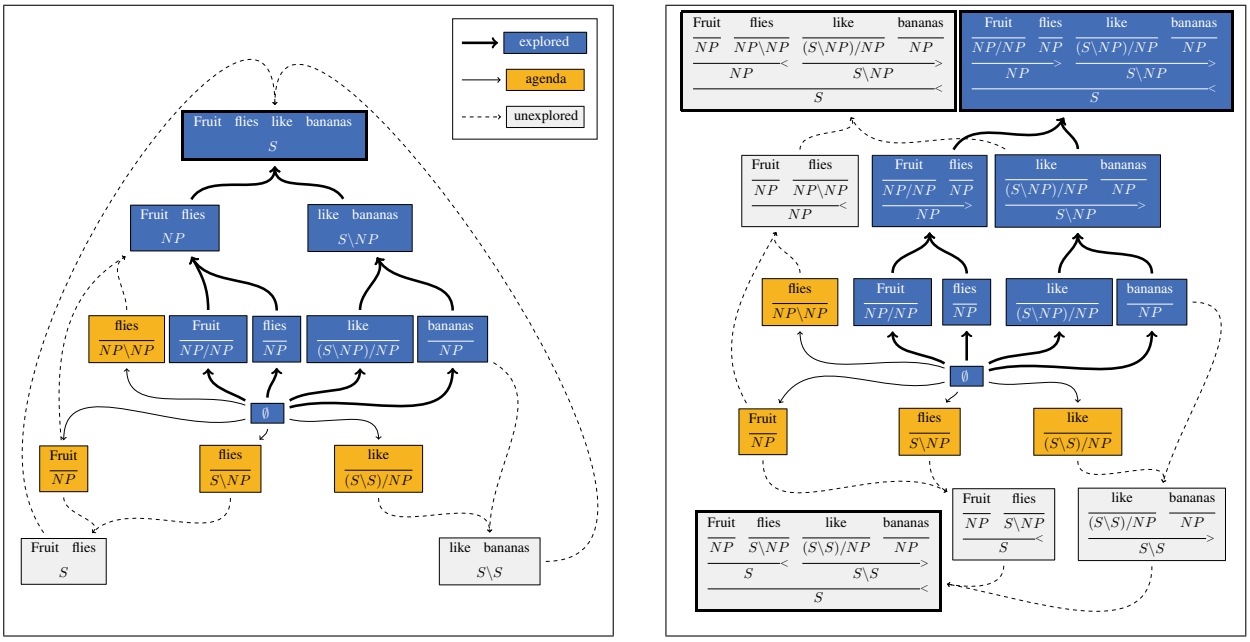
GLOBAL NEURAL CCG PARSING WITH OPTIMALITY GUARANTEES

Recursive neural models perform well for many structured prediction problems, in part due to their ability to learn representations that depend globally on all parts of the output structures. However, global models of this sort are incompatible with existing exact inference algorithms, since they do not decompose over substructures in a way that allows effective dynamic programming. Existing work has therefore used greedy inference techniques such as beam search (Vinyals et al., 2015c; Dyer et al., 2015) or reranking (Socher et al., 2013). We introduce the first global recursive neural parsing approach with optimality guarantees for decoding and use it to build a state-of-the-art CCG parser.

To enable learning of global representations, we modify the parser to search directly in the space of all possible parse trees with no dynamic programming. Optimality guarantees come from A* search, which provides a certificate of optimality if run to completion with a heuristic that is a bound on the future cost. Generalizing A* to global models is challenging; these models also break the locality assumptions used to efficiently compute existing A* heuristics (Klein and Manning, 2003; Lewis and Steedman, 2014).

Rather than directly replacing local models, we show that they can simply be augmented with a score from a global model that is constrained to be non-positive and has a trivial upper bound of zero. The global model, in effect, only needs to model the remaining non-local phenomena. In our experiments, we use a recent factored A* CCG parser (Lewis et al., 2016) for the local model, and we train a Tree-LSTM (Tai et al., 2015) to model global structure.

Finding a model that achieves these A* guarantees in practice is a challenging learning problem. Traditional structured prediction objectives focus on ensuring that the gold parse has the highest



(a) The search space in chart parsing, with one node for each labeling of a span.

(b) The search space in this work, with one node for each partial parse.

Figure 2.1: Illustrations of CCG parsing as hypergraph search, showing partial views of the search space. Weighted hyperedges from child nodes to a parent node represent rule productions scored by a parsing model. A path starting at ∅, for example the set of bolded hyperedges, represents the derivation of a parse. During decoding, we find the highest scoring path to a complete parse. Both figures show an ideal exploration that efficiently finds the optimal path. Figure 2.1a depicts the traditional search space, and Figure 2.1b depicts the search space in this work. Hyperedge scores can only depend on neighboring nodes, so our model can condition on the entire parse structure, at the price of an exponentially larger search space.

score (Collins, 2002; Huang et al., 2012). This condition is insufficient in our case, since it does not guarantee that the search will terminate in sub-exponential time. We instead introduce a new objective that optimizes efficiency as well as accuracy.

Our loss function is defined over states of the A* search agenda, and it penalizes the model

whenever the top agenda item is not a part of the gold parse. Minimizing this loss encourages the model to return the correct parse as quickly as possible.

The combination of global representations and optimal decoding enables our parser to achieve state-of-the-art accuracy for Combinatory Categorial Grammar (CCG) parsing. Despite being intractable in the worst case, the parser in practice is highly efficient.

It finds optimal parses for 99.9% of held out sentences while exploring just 190 subtrees on average—allowing it to outperform beam search in both speed and accuracy.

2.1 Overview

2.1.1 Parsing as hypergraph search

Many parsing algorithms can be viewed as a search problem, where parses are specified by paths through a hypergraph.

A node y in this hypergraph is a labeled span, representing structures within a parse tree, as shown in Figure 2.1. Each hyperedge e in the hypergraph represents a rule production in a parse. The head node of the hyperedge HEAD(e) is the parent of the rule production, and the tail nodes of the hyperedge are the children of the rule production. For example, consider the hyperedge in Figure 2.1b whose head is *like bananas*. This hyperedge represents a forward application rule applied to its tails, *like* and *bananas*.

To define a path in the hypergraph, we first include a special start node $\emptyset$ that represents an empty parse. $\emptyset$ has outgoing hyperedges that reach every leaf node, representing assignments of labels to words (supertag assignments in Figure 2.1). We define a path to be a set of hyperedges E starting at $\emptyset$ and ending at a single destination node. A path therefore specifies the derivation of the parse constructed from the labeled spans at each node. For example, in Figure 2.1, the set of bolded hyperedges form a path deriving a complete parse.

Each hyperedge e is weighted by a score $s(e)$ from a parsing model. The score of a path E is

the sum of its hyperedge scores:

$$g(E) = \sum_{e \in E} s(e)$$

Viterbi decoding is equivalent to finding the highest scoring path that forms a complete parse.

2.1.2 Search on parse forests

Traditionally, the hypergraph represents a packed parse chart. In this work, our hypergraph instead represents a *forest* of parses. Figure 2.1 contrasts the two representations.

In the parse chart, labels on the nodes represent local properties of a parse, such as the category of a span in Figure 2.1a. As a result, multiple parses that contain the same property include the same node in their path, (e.g. the node spanning the phrase *Fruit flies* with category `NP`). The number of nodes in this hypergraph is polynomial in the sentence length, permitting exhaustive exploration. However, the model scores can only depend on local properties of a parse. We refer to these models as *locally factored* models.

In contrast, nodes in the parse forest are labeled with entire subtrees, as shown in Figure 2.1b. For example, there are two nodes spanning the phrase *Fruit flies* with the same category `NP` but different internal substructures. While the parse forest requires an exponential number of nodes in the hypergraph, the model scores can depend on entire subtrees.

2.1.3 A parsing*

A* parsing has been successfully applied in locally factored models (Klein and Manning, 2003; Lewis and Steedman, 2014; Lewis et al., 2015, 2016). We present a special case of A* parsing that is conceptually simpler, since the parse forest constrains each node to be reachable via a unique path. During exploration, we maintain the unique path to a hyperedge e, which we define as $\textsc{path}(e)$.

Similar to the standard A* search algorithm, we maintain an agenda $\mathcal{A}$ of hyperedges to explore and a forest $\mathcal{F}$ of explored nodes that initially contains only the start node $\emptyset$.

Each hyperedge e in the agenda is sorted by the sum of its inside score $g(\text{PATH}(e))$ and an admissible heuristic $h(e)$. A heuristic $h(e)$ is admissible if it is an upper bound of the sum of hyperedge scores leading to any complete parse reachable from e (the Viterbi outside score). The efficiency of the search improves when this bound is tighter.

At every step, the parser removes the top of the agenda, $e_{max} = \text{argmax}_{e \in \mathcal{A}}(g(\text{PATH}(e)) + h(e))$. e_{max} is expanded by combining $\text{HEAD}(e_{max})$ with previously explored parses from $\mathcal{F}$ to form new hyperedges. These new hyperedges are inserted into $\mathcal{A}$, and $\text{HEAD}(e_{max})$ is added it to $\mathcal{F}$. We repeat these steps until the first complete parse y^* is explored. The bounds provided by $h(e)$ guarantee that the path to y^* has the highest possible score. Figure 2.1b shows an example of the agenda and the explored forest at the end of perfectly efficient search, where only the optimal path is explored.

2.1.4 Approach

The enormous search space described above presents a challenge for an A* parser, since computing a tight and admissible heuristic is difficult when the model does not decompose locally.

Our key insight in addressing this challenge is that existing locally factored models with an informative A* heuristic can be augmented with a global score (Section 2.2). By constraining the global score to be non-positive, the A* heuristic from the locally factored model is still admissible.

While the heuristic from the local model offers some estimate of the future cost, the efficiency of the parser requires learning a well-calibrated global score, since the heuristic becomes looser as the global score provides stronger penalties (Section 2.4).

As we explore the search graph, we incrementally construct a neural network, which computes representations of the parses and allows backpropagation of errors from bad search steps (Section 2.3).

In the following sections, we present our approach in detail, assuming an existing locally factored model $s_{local}(e)$ for which we can efficiently compute an admissible A* heuristic $h(e)$.

In the experiments, we apply our model to CCG parsing, using the locally factored model and A* heuristic from Lewis et al. (2016).

2.2 Model

Our model scores a hyperedge e by combining the score from the local model with a global score that conditions on the entire parse at the head node:

$$s(e) = s_{local}(e) + s_{global}(e)$$

In $s_{global}(e)$, we first compute a hidden representation encoding the parse structure of $y = \text{HEAD}(e)$. We use a variant of the Tree-LSTM (Tai et al., 2015) connected to a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) at the leaves. The combination of linear and tree LSTMs allows the hidden representation of partial parses to condition on both the partial structure and the full sentence. Figure 2.2 depicts the neural network that computes the hidden representation for a parse.

Formally, given a sentence $\langle w_1, w_2, \ldots, w_n \rangle$, we compute hidden states h_t and cell states c_t in the forward LSTM for $1 < t \leq n$:

$$\begin{aligned}
i_t &= \sigma(W_i[c_{t-1}, h_{t-1}, x_t] + b_i) \\
o_t &= \sigma(W_o[\tilde{c}_t, h_{t-1}, x_t] + b_o) \\
\tilde{c}_t &= \tanh(W_c[h_{t-1}, x_t] + b_c) \\
c_t &= i_t \circ \tilde{c}_t + (\mathbf{1} - i_t) \circ c_{t-1} \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}$$

where σ is the logistic sigmoid, $\circ$ is the component-wise product, and x_t denotes a learned word embedding for w_t. We also construct a backward LSTM, which produces the analogous hidden and cell states starting at the end of the sentence, which we denote as c'_t and h'_t respectively. The start and end latent states, c_{-1}, h_{-1}, c'_{n+1}, and h'_{n+1}, are learned embeddings. This variant of the LSTM includes peephole connections and couples the input and forget gates.

The bidirectional LSTM over the words serves as a base case when we recursively compute a

hidden representation for the parse y using the tree-structured generalization of the LSTM:

$$
\begin{aligned}
i_y &= \sigma(W_i^R[c_l, h_l, c_r, h_r, x_y] + b_i^R) \\
f_y &= \sigma(W_f^R[c_l, h_l, c_r, h_r, x_y] + b_f^R) \\
o_y &= \sigma(W_o^R[\widetilde{c}_y, h_l, h_r, x_y] + b_o^R) \\
c_{lr} &= f_y \circ c_l + (\mathbf{1} - f_y) \circ c_r \\
\widetilde{c}_y &= \tanh(W_c^R[h_l, h_r, x_y] + b_c^R) \\
c_y &= i_y \circ \widetilde{c}_y + (\mathbf{1} - i_y) \circ c_{lr} \\
h_y &= o_y \circ \tanh(c_y)
\end{aligned}
$$

where the weights and biases are parametrized by the rule R that produces y from its children, and x_y denotes a learned embedding for the category at the root of y. For example, in CCG, the rule would correspond to the CCG combinator, and the label would correspond to the CCG category.

We assume that nodes are binary, unary, or leaves. Their left and right latent states, c_l, h_l, c_r, and h_r are defined as follows:

- In a binary node, c_l and h_l are the cell and hidden states of the left child, and c_r and h_r are the cell and hidden states of the right child.
- In a unary node, c_l and h_l are learned embeddings, and c_r and h_r are the cell and hidden states of the singleton child.
- In a leaf node, let w denote the index of the corresponding word. Then c_l and h_l are c_w and h_w from the forward LSTM, and c_r and h_r are c'_w and h'_w from the backward LSTM.

The cell state of the recursive unit is a linear combination of the intermediate cell state $\widetilde{c}_y$, the left cell state c_l, and the right cell state c_r. To preserve the normalizing property of coupled gates, we perform coupling in a hierarchical manner: the input gate i_y decides the weights for $\widetilde{c}_y$, and the forget gate f_y shares the remaining weights between c_l and c_r.

Given the hidden representation h_y at the root, we score the global component as follows:

$$s_{global}(e) = \log(\sigma(W \cdot h_y))$$

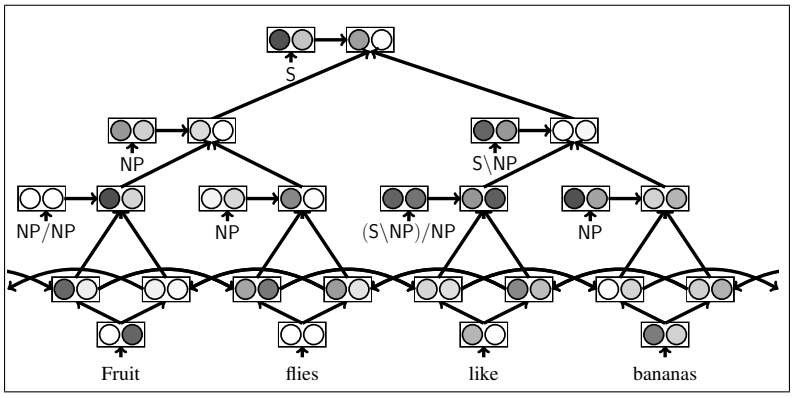


Figure 2.2: Visualization of the Tree-LSTM which computes vector embeddings for each parse node. The leaves of the Tree-LSTM are connected to a bidirectional LSTM over words, encoding lexical information within and outside of the parse.

This definition of the global score ensures that it is non-positive—an important property for inference.

2.3 Inference

Using the hyperedge scoring model $s(e)$ described in Section 2.2, we can find the highest scoring path that derives a complete parse tree by using the A* parsing algorithm described in Section 2.1.

2.3.1 Admissible A* heuristic

Since our global model adds non-positive scores to the existing local scores, path scores under the full model cannot be greater than path scores under the local model. Upper bounds for path scores under the local model also hold for path scores under the full model, and we reuse the A* heuristic from the local model to guide the full model during parsing without sacrificing optimality guarantees.

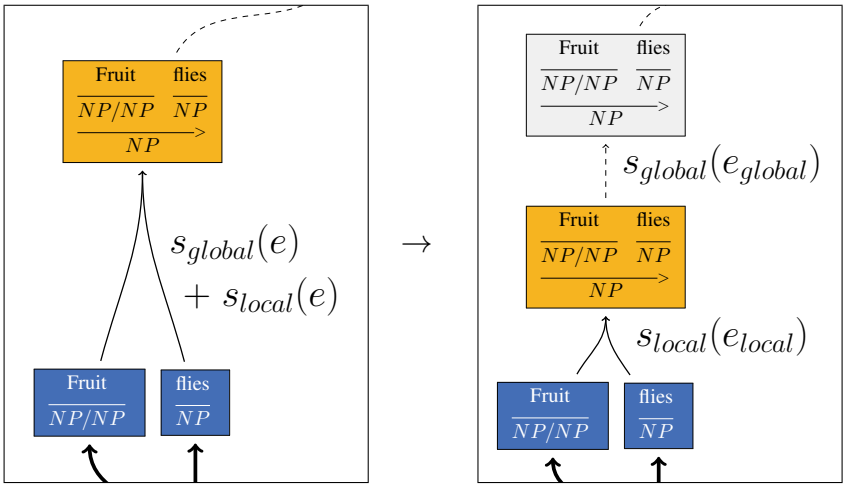


Figure 2.3: The hyperedge on the left requires computing both the local and global score when placed on the agenda. Splitting the hyperedge, as shown on the right, saves expensive computation of the global score if the local score alone indicates that the parse is not worth exploring.

2.3.2 Incremental neural network construction

The recursive hidden representations used in $s_{global}(e)$ can be computed in constant time during parsing. When scoring a new hyperedge, its children must have been previously scored. Instead of computing the full recursion, we reuse the existing latent states of the children and compute $s_{global}(e)$ with an incremental forward pass over a single recursive unit. By maintaining the latent states of each parse, we incrementally build a single DAG-structured LSTM mirroring the explored subset of the hypergraph. This not only enables quick forward passes during decoding, but also allows backpropagation through the search space after decoding, which is crucial for efficient learning (see Section 2.4).

2.3.3 Lazy global scoring

The global score is expensive to compute. We introduce an optimization to avoid computing it when provably unnecessary. We split each hyperedge e into two successive hyperedges, e_{local} and e_{global}, as shown in Figure 2.3. The score for e, previously $s(e) = s_{local}(e) + s_{global}(e)$, is also split

between the two new hyperedges:

$$s(e_{local}) = s_{local}(e_{local})$$
$$s(e_{global}) = s_{global}(e_{global})$$

Intuitively, this transformation requires A* to verify that the local score is good enough before computing the global score, which requires an incremental forward pass over a recursive unit in the neural network. In the example, this involves first summing the supertag scores of *Fruit* and *flies* and inserting the result back into the agenda. The score for applying the forward application rule to the recursive representations is only computed if that item appears again at the head of the agenda. In practice, the lazy global scoring reduces the number of recursive units by over 91%, providing a 2.4X speed up.

2.4 Learning

During training, we assume access to sentences labeled with gold parse trees $\hat{y}$ and gold derivations $\hat{E}$. The gold derivation $\hat{E}$ is a path from $\emptyset$ to $\hat{y}$ in the parse forest.

A* search with our global model is not guaranteed to terminate in sub-exponential time. This creates challenges for learning—for example, it is not possible in practice to use the standard structured perceptron update (Collins, 2002), because the search procedure rarely terminates early in training. Other common loss functions assume inexact search (Huang et al., 2012), and do not optimize efficiency.

Instead, we optimize a new objective that is tightly coupled with the search procedure. During parsing, we would like hyperedges from the gold derivation to appear at the top of the agenda $\mathcal{A}$. When this condition does not hold, A* is searching inefficiently, and we refer to this as a *violation* of the agenda, which we formally define as:

$$v(\hat{E}, \mathcal{A}) = \max_{e \in \mathcal{A}} (g(\text{PATH}(e)) + h(e)) - \max_{e \in \mathcal{A} \cap \hat{E}} (g(\text{PATH}(e)) + h(e))$$

Update	**Loss($\mathcal{V}$)**
Greedy	$\mathcal{V}_1$
Max violation	$\max_{t=1}^{T} \mathcal{V}_t$
All violations	$\sum_{t=1}^{T} \mathcal{V}_t$

Table 2.1: Loss functions optimized by the different update methods. The updates depend on the list of T non-zero violations, $\mathcal{V} = \langle \mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_T \rangle$, as defined in Section 2.4.

where $g(\text{PATH}(e))$ is the score of the unique path to e, and $h(e)$ is the A* heuristic. If all violations are zero, we find the gold parse without exploring any incorrect partial parses—maximizing both accuracy and efficiency. Figure 2.1b shows such a case—if any other nodes were explored, they would be violations.

In existing work on violation-based updates, comparisons are only made between derivations with the same number of steps (Huang et al., 2012; Clark et al., 2015)—whereas our definition allows subtrees of arbitrary spans to compete with each other, because hyperedges are not explored in a fixed order. Our violations also differ from Huang et al.'s in that we optimize efficiency as well as accuracy.

We define loss functions over these violations, which are minimized to encourage correct and efficient search. During training, we parse each sentence until either the gold parse is found or we reach computation limits. We record $\mathcal{V}$, the list of non-zero violations of the agenda $\mathcal{A}$ observed:

$$\mathcal{V} = \langle v(\hat{E}, \mathcal{A}) \mid v(\hat{E}, \mathcal{A}) > 0 \rangle$$

We can optimize several loss functions over $\mathcal{V}$, as defined in Table 2.1. The greedy and max-violation updates are roughly analogous to the violation-fixing updates proposed by Huang et al. (2012), but adapted to exact agenda-based parsing. We also introduce a new *all-violations* update, which minimizes the sum of all observed violations. The all-violations update encourages correct parses to be explored early (similar to the greedy update) while being robust to parses with multiple

deviations from the gold parse (similar to the max-violation update).

The violation losses are optimized with subgradient descent and backpropagation. For our experiments, $s_{local}(e)$ and $h(e)$ are kept constant. Only the parameters θ of $s_{global}(e)$ are updated. Therefore, a subgradient of a violation $v(\hat{E}, \mathcal{A})$ can be computed by summing subgradients of the global scores.

$$\frac{\partial v(\hat{E}, \mathcal{A})}{\partial \theta} = \sum_{e \in \text{PATH}(e_{max})} \frac{\partial s_{global}(e)}{\partial \theta} - \sum_{e \in \text{PATH}(\hat{e}_{max})} \frac{\partial s_{global}(e)}{\partial \theta}$$

where e_{max} denotes the hyperedge at the top of the agenda $\mathcal{A}$ and $\hat{e}_{max}$ denotes the hyperedge in the gold derivation $\hat{E}$ that is closest to the top of $\mathcal{A}$.

2.5 Experiments

2.5.1 Data

We trained our parser on Sections 02-21 of CCGbank (Hockenmaier and Steedman, 2007), using Section 00 for development and Section 23 for test. To recover a single gold derivation for each sentence to use during training, we find the right-most branching parse that satisfies the gold dependencies.

2.5.2 Experimental Setup

For the local model, we use the *supertag-factored* model of Lewis et al. (2016). Here, $s_{local}(e)$ corresponds to a supertag score if a $\text{HEAD}(e)$ is a leaf and zero otherwise. The outside score heuristic is computed by summing the maximum supertag score for every word outside of each span. In the reported results, we back off to the supertag-factored model after the forest size exceeds 500,000, the agenda size exceeds 2 million, or we build more than 200,000 recursive units in the neural network.

Our full system is trained with all-violations updates. During training, we lower the forest size limit to 2000 to reduce training times. The model is trained for 30 epochs using ADAM (Kingma and Ba, 2014), and we use early stopping based on development F1. The LSTM cells and hidden

states have 64 dimensions. We initialize word representations with pre-trained 50-dimensional embeddings from Turian et al. (2010). Embeddings for categories have 16 dimensions and are randomly initialized. We also apply dropout with a probability of 0.4 at the word embedding layer during training. Since the structure of the neural network is dynamically determined, we do not use mini-batches. The neural networks are implemented using the CNN library,[1] and the CCG parser is implemented using the EasySRL library.[2] The code is available online.[3]

2.5.3 Baselines

We compare our parser to several baseline CCG parsers: the C&C parser (Clark and Curran, 2007); C&C + RNN (Xu et al., 2015), which is the C&C parser with an RNN supertagger; Xu (2016), a LSTM shift-reduce parser; Vaswani et al. (2016) who combine a bidirectional LSTM supertagger with a beam search parser using global features (Clark et al., 2015); and *supertag-factored* (Lewis et al., 2016), which uses deterministic A* decoding and an LSTM supertagging model.

2.5.4 Parsing Results

Table 2.2 shows parsing results on the test set. Our global features let us improve over the supertag-factored model by 0.6 F1. Vaswani et al. (2016) also use global features, but our optimal decoding leads to an improvement of 0.4 F1.

Although we observed an overall improvement in parsing performance, the supertag accuracy was not significantly different after applying the parser.

On the test data, the parser finds the optimal parse for 99.9% sentences before reaching our computational limits. On average, we parse 27.1 sentences per second,[4] while exploring only 190.2 subtrees.

---

[1] `https://github.com/clab/cnn`

[2] `https://github.com/mikelewis0/EasySRL`

[3] `https://github.com/kentonl/neuralccg`

[4] We use a single 3.5GHz CPU core.

Model	Dev F1	Test F1
C & C	83.8	85.2
C & C + RNN	86.3	87.0
Xu (2016)	87.5	87.8
Vaswani et al. (2016)	87.8	88.3
Supertag-factored	87.5	88.1
Global A*	**88.4**	**88.7**

Table 2.2: Labeled F1 for CCGbank dependencies on the CCGbank development and test set for our system **Global A*** and the baselines.

2.6 Model Ablations

We ablate various parts of the model to determine how they contribute to the accuracy and efficiency of the parser, as shown in Table 2.3. For each model, the comparisons include the average number of parses explored and the percentage of sentences for which an optimal parse can be found without backing off.

2.6.1 Structure ablation

We first ablate the global score, $s_{global}(y)$, from our model, thus relying entirely on the local supertag-factors that do not explicitly model the parse structure. This ablation allows dynamic programming and is equivalent to the backoff model (*supertag-factored* in Table 2.3). Surprisingly, even in the exponentially larger search space, the global model explores *fewer* nodes than the supertag-factored model—showing that the global model efficiently prune large parts of the search space. This effect is even larger when not using dynamic programming in the supertag-factored model.

Model	Dev F1	Optimal	Explored
Supertag-factored	87.5	100.0%	402.5
− dynamic program	87.5	97.1%	17119.6
Span-factored	87.9	99.9%	176.5
− dynamic program	87.8	99.5%	578.5
Global A*	**88.4**	99.8%	309.6
− lexical inputs	87.8	99.6%	538.5
− lexical context	88.1	99.4%	610.5

Table 2.3: Ablations of our full model (**Global A***) on the development set. *Explored* refers to the size of the parse forest. Results show the importance of global features and lexical information in context.

2.6.2 Global structure ablation

To examine the importance of global features, we ablate the recursive hidden representation (*span-factored* in Table 2.3). The model in this ablation decomposes over labels for spans, as in Durrett and Klein (2015). In this model, the recursive unit uses, instead of latent states from its children, the latent states of the backward LSTM at the start of the span and the latent states of the forward LSTM at the end of the span. Therefore, this model encodes the lexical information available in the full model but does not encode the parse structure beyond the local rule production. While the dynamic program allows this model to find the optimal parse with fewer explorations, the lack of global features significantly hurts its parsing accuracy.

2.6.3 Lexical ablation

We also show lexical ablations instead of structural ablations. We remove the bidirectional LSTM at the leaves, thus delexicalizing the global model. This ablation degrades both accuracy and

Update	Dev F1	Optimal	Explored
Greedy	87.9	99.2%	2313.8
Max-violation	88.1	99.9%	217.3
All-violations	**88.4**	99.8%	309.6

Table 2.4: Parsing results trained with different update methods. Our system uses **all-violations** updates and is the most accurate.

efficiency, showing that the model uses lexical information to discriminate between parses.

To understand the importance of contextual information, we also perform a partial lexical ablation by using word embeddings at the leaves instead of the BiLSTM, thus propagating only lexical information from within the span of each parse. The degradation in F1 is about half of the degradation from the full lexical ablation, suggesting that a significant portion of the lexical cues comes from the context of a parse.

2.7 Update Comparisons

Table 2.4 compares the different learning objectives, as discussed in Section 2.4. Our novel *all-violation* updates outperform the alternatives. We attribute this improvement to the robustness over poor search spaces, which the greedy update lacks, and the incentive to explore good parses early, which the max-violation update lacks. Learning curves in Figure 2.4 show that the all-violations update also converges more quickly.

2.8 Decoder Comparisons

Lastly, to show that our parser is both more accurate and efficient than other decoding methods, we decode our full model using best-first search, reranking, and beam search. Table 2.5 shows the F1 scores with and without the backoff model, the portion of the sentences that each decoder is able to parse, and the time spent decoding relative to the A^* parser.

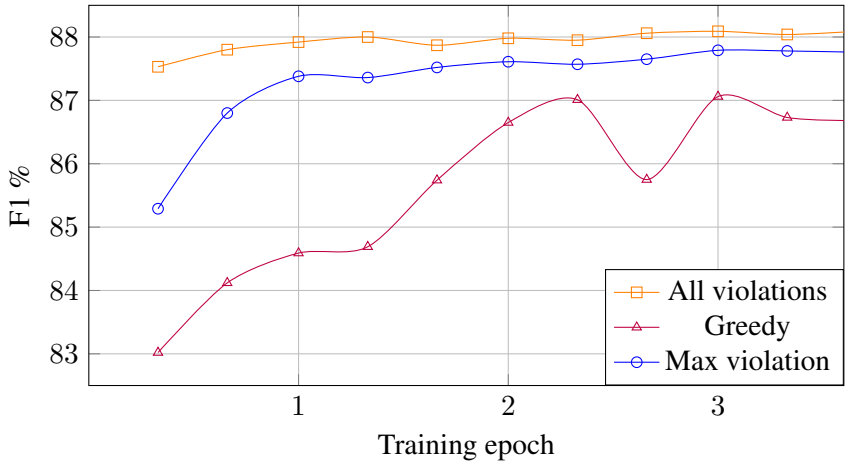


Figure 2.4: Learning curves for the first 3 training epochs on the development set when training with different updates strategies. The all-violations update shows the fastest convergence.

In the best-first search comparison, we do not include the informative A* heuristic, and the parser completes very few parses before reaching computational limits—showing the importance of heuristics in large search spaces. In the reranking comparison, we obtain n-best lists from the backoff model and rerank each result with the full model. In the beam search comparison, we use the approach from Clark et al. (2015) which greedily finds the top-n parses for each span in a bottom-up manner. Results indicate that both approximate methods are less accurate and slower than A*.

2.9 Related Work

Many structured prediction problems are based around dynamic programs, which are incompatible with recursive neural networks because of their real-valued latent variables. Some recent models have neural factors (Durrett and Klein, 2015), but these cannot condition on global parse structure, making them less expressive. Our search explores fewer nodes than dynamic programs, despite an exponentially larger search space, by allowing the recursive neural network to guide the search.

Previous work on structured prediction with recursive or recurrent neural models has used

Decoder	Dev F1	Dev F1 − backoff	Relative Time
Global A*	**88.4**	88.4 (99.8%)	1X
Best-first	87.5	2.8 (6.7%)	293.4X
10-best reranking	87.9	87.9 (99.7%)	8.5X
100-best reranking	88.2	88.0 (99.4%)	72.3X
2-best beam search	88.2	85.7 (94.0%)	2.0X
4-best beam search	88.3	88.1 (99.2%)	6.7X
8-best beam search	88.2	86.8 (98.1%)	26.3X

Table 2.5: Comparison of various decoders using the same model from our full system (**Global A***). We report F1 with and without the backoff model, the percentage of sentences that the decoder can parse, and the time spent decoding relative to A*.

beam search–e.g. in shift reduce parsing (Dyer et al., 2015), string-to-tree transduction (Vinyals et al., 2015c), or reranking (Socher et al., 2013)–at the cost of potentially recovering suboptimal solutions. For our model, beam search is both less efficient and less accurate than optimal A* decoding. In the non-neural setting, Zhang et al. (2014) showed that global features with greedy inference can improve dependency parsing. The CCG beam search parser of Clark et al. (2015), most related to this work, also uses global features. By using neural representations and exact search, we improve over their results.

A* parsing has been previously proposed for locally factored models (Klein and Manning, 2003; Pauls and Klein, 2009; Auli and Lopez, 2011; Lewis and Steedman, 2014). We generalize these methods to enable global features. Vaswani and Sagae (2016) apply best-first search to an unlabeled shift-reduce parser. Their use of error states is related to our global model that penalizes local scores. We demonstrated that best-first search is infeasible in our setting, due to the larger search space.

A close integration of learning and decoding has been shown to be beneficial for structured

prediction. SEARN (Daumé III et al., 2009) and DAGGER (Ross et al., 2011) learn greedy policies to predict structure by sampling classification examples over actions from single states. We similarly generate classification examples over hyperedges in the agenda, but actions from multiple states compete against each other. Other learning objectives that update parameters based on a beam or agenda of partial structures have also been proposed (Collins and Roark, 2004; Daumé III and Marcu, 2005b; Huang et al., 2012; Andor et al., 2016; Wiseman et al., 2016), but the impact of search errors is unclear.

2.10 Conclusion

We have shown for the first time that a parsing model with global features can be decoded with optimality guarantees. This enables the use of powerful recursive neural networks for parsing without resorting to approximate decoding methods. Experiments show that this approach is effective for CCG parsing, resulting in a new state-of-the-art parser. The result indicates the importance of structured learning and inference for parsing, where the output structure is highly compositional. In the following section, we discuss a less structured approach for coreference resolution, where the output structure is conceptually simpler, and state-of-the-art performance can be achieved without modeling full clusters.

Chapter 3

LEARNING RECURRENT SPAN REPRESENTATIONS FOR EXTRACTIVE QUESTION ANSWERING

A primary goal of natural language processing is to develop systems that can answer questions about the contents of documents. The reading comprehension task is of practical interest – we want computers to be able to read the world's text and then answer our questions – and, since we believe it requires deep language understanding, it has also become a flagship task in NLP research.

A number of reading comprehension datasets have been developed that focus on answer selection from a small set of alternatives defined by annotators (Richardson et al., 2013b) or existing NLP pipelines that cannot be trained end-to-end (Hill et al., 2016; Hermann et al., 2015). Subsequently, the models proposed for this task have tended to make use of the limited set of candidates, basing their predictions on mention-level attention weights (Hermann et al., 2015), or centering classifiers (Chen et al., 2016), or network memories (Hill et al., 2016) on candidate locations.

Recently, Rajpurkar et al. (2016) released the less restricted SQuAD dataset[1] that does not place any constraints on the set of allowed answers, other than that they should be drawn from the evidence document. Rajpurkar et al. proposed a baseline system that chooses answers from the constituents identified by an existing syntactic parser. This allows them to prune the $O(N^2)$ answer candidates in each document of length N, but it also effectively renders 20.7% of all questions unanswerable.

Subsequent work by Wang and Jiang (2016) significantly improves upon this baseline by using an end-to-end neural network architecture to identify answer spans by labeling either individual words, or the start and end of the answer span. Both of these methods do not make independence assumptions about substructures, but they are susceptible to search errors due to greedy training

[1] http://stanford-qa.com

and decoding.

In contrast, here we argue that it is beneficial to simplify the decoding procedure by enumerating all possible answer spans. By explicitly representing each answer span, our model can be globally normalized during training and decoded exactly during evaluation. A naive approach to building the $O(N^2)$ spans of up to length N would require a network that is cubic in size with respect to the passage length, and such a network would be untrainable. To overcome this, we present a novel neural architecture called RASOR that builds fixed-length span representations, *reusing* recurrent computations for shared substructures. We demonstrate that directly classifying each of the competing spans, and training with global normalization over all possible spans, leads to a significant increase in performance. In our experiments, we show an increase in performance over Wang and Jiang (2016) of 5% in terms of exact match to a reference answer, and 3.6% in terms of predicted answer F1 with respect to the reference. On both of these metrics, we close the gap between Rajpurkar et al.'s baseline and the human-performance upper-bound by $> 50\%$.

3.1 Extractive Question Answering

3.1.1 Task Definition

Extractive question answering systems take as input a question $\mathbf{q} = \{q_0, \ldots, q_n\}$ and a passage of text $\mathbf{p} = \{p_0, \ldots, p_m\}$ from which they predict a single answer span $\mathbf{a} = \langle a_{start}, a_{end} \rangle$, represented as a pair of indices into $\mathbf{p}$. Machine learned extractive question answering systems, such as the one presented here, learn a predictor function $f(\mathbf{q}, \mathbf{p}) \rightarrow \mathbf{a}$ from a training dataset of $\langle \mathbf{q}, \mathbf{p}, \mathbf{a} \rangle$ triples.

3.1.2 Related Work

For the SQUAD dataset, the original paper from Rajpurkar et al. (2016) implemented a linear model with sparse features based on n-grams and part-of-speech tags present in the question and the candidate answer. Other than lexical features, they also used syntactic information in the form of dependency paths to extract more general features. They set a strong baseline for following work and also presented an in depth analysis, showing that lexical and syntactic features contribute

most strongly to their model's performance. Subsequent work by Wang and Jiang (2016) use an end-to-end neural network method that uses a Match-LSTM to model the question and the passage, and uses pointer networks (Vinyals et al., 2015a) to extract the answer span from the passage. This model resorts to greedy decoding and falls short in terms of performance compared to our model (see Section 3.4 for more detail). While we only compare to published baselines, there are other unpublished competitive systems on the SQuAD leaderboard, as listed in footnote 4.

A task that is closely related to extractive question answering is the Cloze task (Taylor, 1953), in which the goal is to predict a concealed span from a declarative sentence given a passage of supporting text. Recently, Hermann et al. (2015) presented a Cloze dataset in which the task is to predict the correct entity in an incomplete sentence given an abstractive summary of a news article. Hermann et al. also present various neural architectures to solve the problem. Although this dataset is large and varied in domain, recent analysis by Chen et al. (2016) shows that simple models can achieve close to the human upper bound. As noted by the authors of the SQuAD paper, the annotated answers in the SQuAD dataset are often spans that include non-entities and can be longer phrases, unlike the Cloze datasets, thus making the task more challenging.

Another, more traditional line of work has focused on extractive question answering on sentences, where the task is to extract a sentence from a document, given a question. Relevant datasets include datasets from the annual TREC evaluations (Voorhees and Tice, 2000) and WikiQA (Yang et al., 2015), where the latter dataset specifically focused on Wikipedia passages. There has been a line of interesting recent publications using neural architectures, focused on this variety of extractive question answering (Tymoshenko et al., 2016; Wang et al., 2016, *inter alia*). These methods model the question and a candidate answer sentence, but do not focus on possible candidate answer *spans* that may contain the answer to the given question. In this work, we focus on the more challenging problem of extracting the precise answer span.

3.2 Model

We propose a model architecture called RASOR[2] illustrated in Figure 3.1, that explicitly computes embedding representations for candidate answer spans. In most structured prediction problems (e.g. sequence labeling or parsing), the number of possible output structures is exponential in the input length, and computing representations for every candidate is prohibitively expensive. However, we exploit the simplicity of our task, where we can trivially and tractably enumerate all candidates. This facilitates an expressive model that computes joint representations of every answer span, that can be globally normalized during learning.

In order to compute these span representations, we must aggregate information from the passage and the question for every answer candidate. For the example in Figure 3.1, RASOR computes an embedding for the candidate answer spans: *fixed to*, *fixed to the*, *to the*, etc. A naive approach for these aggregations would require a network that is cubic in size with respect to the passage length. Instead, our model reduces this to a quadratic size by reusing recurrent computations for shared substructures (i.e. common passage words) from different spans.

Since the choice of answer span depends on the original question, we must incorporate this information into the computation of the span representation. We model this by augmenting the passage word embeddings with additional embedding representations of the question.

In this section, we motivate and describe the architecture for RASOR in a top-down manner.

3.2.1 Scoring Answer Spans

The goal of our extractive question answering system is to predict the single best answer span among all candidates from the passage $\mathbf{p}$, denoted as $\mathbf{A}(\mathbf{p})$. Therefore, we define a probability distribution over all possible answer spans given the question $\mathbf{q}$ and passage $\mathbf{p}$, and the predictor function finds the answer span with the maximum likelihood:

$$f(\mathbf{q}, \mathbf{p}) := \underset{\mathbf{a} \in \mathbf{A}(\mathbf{p})}{\operatorname{argmax}} P(\mathbf{a} \mid \mathbf{q}, \mathbf{p}) \tag{3.1}$$

[2] An abbreviation for Recurrent Span Representations, pronounced as *razor*.

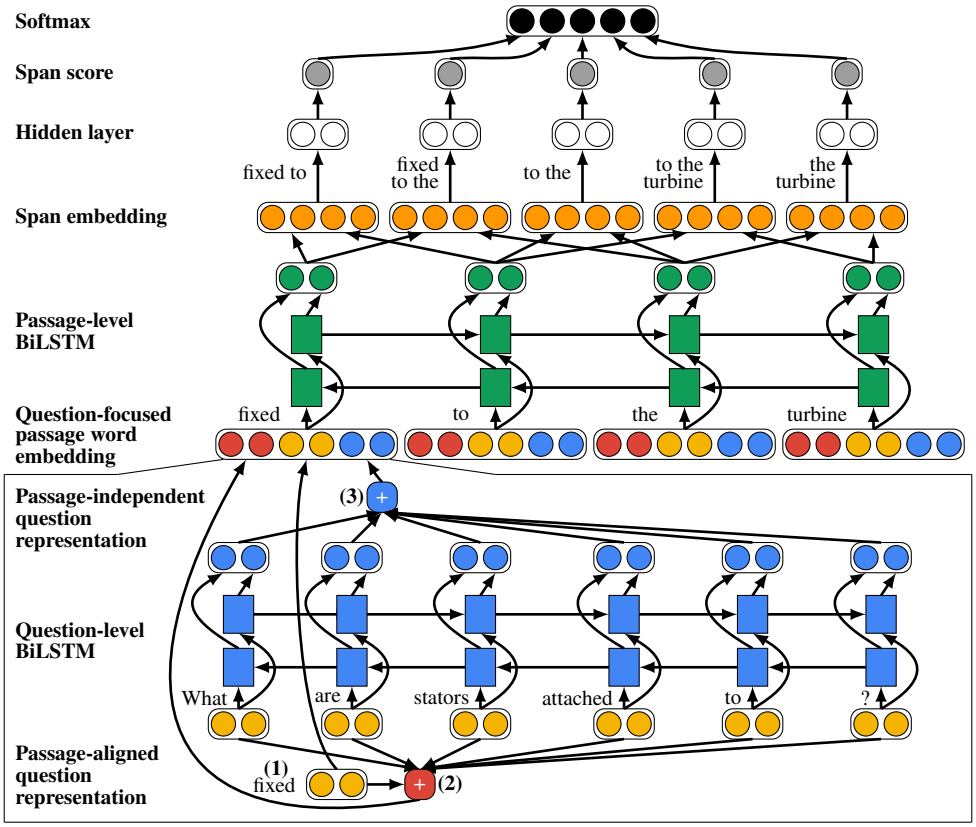


Figure 3.1: A visualization of RASOR, where the question is *"What are the stators attached to?"* and the passage is *"...fixed to the turbine ..."*. The model constructs question-focused passage word embeddings by concatenating **(1)** the original passage word embedding, **(2)** a passage-aligned representation of the question, and **(3)** a passage-independent representation of the question shared across all passage words. We use a BiLSTM over these concatenated embeddings to efficiently recover embeddings of all possible spans, which are then scored by the last layer of the model.

One might be tempted to introduce independence assumptions that would enable cheaper decoding. For example, this distribution can be modeled as (1) a product of conditionally independent distributions (binary) for every word or (2) a product of conditionally independent distributions (over words) for the start and end indices of the answer span. However, we show in Section 3.4.2 that such independence assumptions hurt the accuracy of the model, and instead we only assume a

fixed-length representation $h_{\mathbf{a}}$ of each candidate span that is scored and normalized with a softmax layer (**Span score** and **Softmax** in Figure 3.1):

$$s_{\mathbf{a}} = w_a \cdot \text{FFNN}(h_{\mathbf{a}}) \qquad \mathbf{a} \in \mathbf{A}(\mathbf{p}) \tag{3.2}$$

$$P(\mathbf{a} \mid \mathbf{q}, \mathbf{p}) = \frac{\exp(s_{\mathbf{a}})}{\sum_{\mathbf{a}' \in \mathbf{A}(\mathbf{p})} \exp(s_{\mathbf{a}'})} \qquad \mathbf{a} \in \mathbf{A}(\mathbf{p}) \tag{3.3}$$

where $\text{FFNN}(\cdot)$ denotes a fully connected feed-forward neural network that provides a non-linear mapping of its input embedding, and w_a denotes a learned vector for scoring the last layer of the feed-forward neural network.

3.2.2 RASOR*: Recurrent Span Representation*

The previously defined probability distribution depends on the answer span representations, $h_{\mathbf{a}}$. When computing $h_{\mathbf{a}}$, we assume access to representations of individual passage words that have been augmented with a representation of the question. We denote these question-focused passage word embeddings as $\{p_1^*, \ldots, p_m^*\}$ and describe their creation in Section 3.2.3. In order to reuse computation for shared substructures, we use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to encode the left and right context of every p_i^* (**Passage-level BiLSTM** in Figure 3.1). This allows us to simply concatenate the bidirectional LSTM (BiLSTM) outputs at the endpoints of a span to jointly encode its inside and outside information (**Span embedding** in Figure 3.1):

$$\{p_1^{*\prime}, \ldots, p_m^{*\prime}\} = \text{BILSTM}(\{p_1^*, \ldots, p_m^*\}) \tag{3.4}$$

$$h_{\mathbf{a}} = [p_{a_{start}}^{*\prime}, p_{a_{end}}^{*\prime}] \qquad \langle a_{start}, a_{end} \rangle \in \mathbf{A}(\mathbf{p}) \tag{3.5}$$

where $\text{BILSTM}(\cdot)$ denotes a BiLSTM over its input embedding sequence and $p_i^{*\prime}$ is the concatenation of forward and backward outputs at time-step i. While the visualization in Figure 3.1 shows a single layer BiLSTM for simplicity, we use a multi-layer BiLSTM in our experiments. The concatenated output of each layer is used as input for the subsequent layer, allowing the upper layers to depend on the entire passage.

3.2.3 Question-focused Passage Word Embedding

Computing the question-focused passage word embeddings $\{p_1^*, \ldots, p_m^*\}$ requires integrating question information into the passage. The architecture for this integration is flexible and likely depends on the nature of the dataset. For the SQuAD dataset, we find that both passage-aligned and passage-independent question representations are effective at incorporating this contextual information, and experiments will show that their benefits are complementary. To incorporate these question representations, we simply concatenate them with the passage word embeddings (**Question-focused passage word embedding** in Figure 3.1).

We use fixed pretrained embeddings to represent question and passage words. Therefore, in the following discussion, notation for the words are interchangeable with their embedding representations.

Question-independent passage word embedding The first component simply looks up the pretrained word embedding for the passage word, p_i.

Passage-aligned question representation In this dataset, the question-passage pairs often contain large lexical overlap or similarity near the correct answer span. To encourage the model to exploit these similarities, we include a fixed-length representation of the question based on soft alignments with the passage word. The alignments are computed via neural attention (Bahdanau et al., 2014), and we use the variant proposed by Parikh et al. (2016), where attention scores are dot products between non-linear mappings of word embeddings.

$$s_{ij} = \text{FFNN}(p_i) \cdot \text{FFNN}(q_j) \qquad 1 \leq j \leq n \tag{3.6}$$

$$a_{ij} = \frac{\exp(s_{ij})}{\sum_{k=1}^{n} \exp(s_{ik})} \qquad 1 \leq j \leq n \tag{3.7}$$

$$q_i^{align} = \sum_{j=1}^{n} a_{ij} q_j \tag{3.8}$$

Passage-independent question representation We also include a representation of the question that does not depend on the passage and is shared for all passage words.

Similar to the previous question representation, an attention score is computed via a dot-product, except the question word is compared to a universal learned embedding rather any particular passage word. Additionally, we incorporate contextual information with a BiLSTM before aggregating the outputs using this attention mechanism.

The goal is to generate a coarse-grained summary of the question that depends on word order. Formally, the passage-independent question representation q^{indep} is computed as follows:

$$\{q'_1, \ldots, q'_n\} = \text{BILSTM}(\mathbf{q}) \tag{3.9}$$

$$s_j = w_q \cdot \text{FFNN}(q'_j) \qquad 1 \leq j \leq n \tag{3.10}$$

$$a_j = \frac{\exp(s_j)}{\sum_{k=1}^{n} \exp(s_k)} \qquad 1 \leq j \leq n \tag{3.11}$$

$$q^{indep} = \sum_{j=1}^{n} a_j q'_j \tag{3.12}$$

where w_q denotes a learned vector for scoring the last layer of the feed-forward neural network.

This representation is a bidirectional generalization of the question representation recently proposed by Li et al. (2016) for a different question-answering task.

Given the above three components, the complete question-focused passage word embedding for p_i is their concatenation: $p_i^* = [p_i, q_i^{align}, q^{indep}]$.

3.2.4 Learning

Given the above model specification, learning is straightforward. We simply maximize the log-likelihood of the correct answer candidates and backpropagate the errors end-to-end.

3.3 Experimental Setup

We represent each of the words in the question and document using 300 dimensional GloVe embeddings trained on a corpus of 840bn words (Pennington et al., 2014). These embeddings cover 200k words and all out of vocabulary (OOV) words are projected onto one of 1m randomly initialized 300d embeddings. We couple the input and forget gates in our LSTMs, as described in Greff

et al. (2016), and we use a single dropout mask to apply dropout across all LSTM time-steps as proposed by Gal and Ghahramani (2016a). Hidden layers in the feed-forward neural networks use rectified linear units (Nair and Hinton, 2010). Answer candidates are limited to spans with at most 30 words.

To choose the final model configuration, we ran grid searches over: the dimensionality of the LSTM hidden states $(25, 50, 100, 200)$; the number of stacked LSTM layers $(1, 2, 3)$; the width $(50, 100, 150, 200)$ and depth $(1, 2)$ of the feed-forward neural networks; the dropout rate $(0, 0.1, 0.2)$; and the decay multiplier $(0.9, 0.95, 1.0)$ with which we multiply the learning rate every 10k steps. The best model uses a single 150d hidden layer in all feed-forward neural networks; 50d LSTM states; two-layer BiLSTMs for the span encoder and the passage-independent question representation; dropout of 0.1 throughout; and a learning rate decay of 5% every 10k steps.

All models are implemented using TensorFlow[3] and trained on the SQuAD training set using the ADAM (Kingma and Ba, 2014) optimizer with a mini-batch size of 4 and trained using 10 asynchronous training threads on a single machine.

3.4 Results

We train on the 80k (question, passage, answer span) triples in the SQuAD training set and report results on the 10k examples in the SQuAD development and test sets.

All results are calculated using the official SQuAD evaluation script, which reports exact answer match and F1 overlap of the unigrams between the predicted answer and the closest labeled answer from the 3 reference answers given in the SQuAD development set.

3.4.1 Comparisons to other work

Our model with recurrent span representations (RaSoR) is compared to all previously published systems [4]. Rajpurkar et al. (2016) published a logistic regression baseline as well as human per-

[3] `www.tensorflow.org`

[4] As of submission, other unpublished systems are shown on the SQuAD leaderboard, including *Match-LSTM with Ans-Ptr (Boundary+Ensemble)*, *Co-attention*, *r-net*, *Match-LSTM with Bi-Ans-Ptr (Boundary)*, *Co-attention old*,

	Dev		Test	
System	EM	F1	EM	F1
Logistic regression baseline	39.8	51.0	40.4	51.0
Match-LSTM (Sequence)	54.5	67.7	54.8	68.0
Match-LSTM (Boundary)	60.5	70.7	59.4	70.0
RASOR	66.4	74.9	67.4	75.5
Human	81.4	91.0	82.3	91.2

Table 3.1: Exact match (EM) and span F1 on SQUAD.

formance on the SQUAD task. The logistic regression baseline uses the output of an existing syntactic parser both as a constraint on the set of allowed answer spans, and as a method of creating sparse features for an answer-centric scoring model. Despite not having access to any external representation of linguistic structure, RASOR achieves an error reduction of more than 50% over this baseline, both in terms of exact match and F1, relative to the human performance upper bound.

More closely related to RASOR is the *boundary model* with Match-LSTMs and Pointer Networks by Wang and Jiang (2016). Their model similarly uses recurrent networks to learn embeddings of each passage word in the context of the question, and it can also capture interactions between endpoints, since the end index probability distribution is conditioned on the start index. However, both training and evaluation are greedy, making their system susceptible to search errors when decoding. In contrast, RASOR can efficiently and explicitly model the quadratic number of possible answers, which leads to a 14% error reduction over the best performing Match-LSTM model.

Dynamic Chunk Reader, *Dynamic Chunk Ranker with Convolution layer*, *Attentive Chunker*.

Question representation	EM	F1
Only passage-independent	48.7	56.6
Only passage-aligned	63.1	71.3
RASOR	66.4	74.9

Table 3.2: Ablations results.

3.4.2 Model Variations

We investigate two main questions in the following ablations and comparisons. (1) How important are the two methods of representing the question described in Section 3.2.3? (2) What is the impact of learning a loss function that accurately reflects the span prediction task?

Question representations Table 3.2 shows the performance of RASOR when either of the two question representations described in Section 3.2.3 is removed. The passage-aligned question representation is crucial, since lexically similar regions of the passage provide strong signal for relevant answer spans. If the question is only integrated through the inclusion of a passage-independent representation, performance drops drastically. The passage-independent question representation over the BiLSTM is less important, but it still accounts for over 3% exact match and F1. The input of both of these components is analyzed qualitatively in Section 3.5.

Learning objectives Given a fixed architecture that is capable of encoding the input question-passage pairs, there are many ways of setting up a learning objective to encourage the model to predict the correct span. In Table 3.3, we provide comparisons of some alternatives (learned end-to-end) given only the passage-level BiLSTM from RASOR. In order to provide clean comparisons, we restrict the alternatives to objectives that are trained and evaluated with exact decoding.

The simplest alternative is to consider this task as binary classification for every word (*Membership prediction* in Table 3.3). In this baseline, we optimize the logistic loss for binary labels

indicating whether passage words belong to the correct answer span. At prediction time, a valid span can be recovered in linear time by finding the maximum contiguous sum of scores.

Li et al. (2016) proposed a sequence-labeling scheme that is similar to the above baseline (*BIO sequence prediction* in Table 3.3). We follow their proposed model and learn a conditional random field (CRF) layer after the passage-level BiLSTM to model transitions between the different labels. At prediction time, a valid span can be recovered in linear time using Viterbi decoding, with hard transition constraints to enforce a single contiguous output.

We also consider a model that independently predicts the two endpoints of the answer span (*Endpoints prediction* in Table 3.3). This model uses the softmax loss over passage words during learning. When decoding, we only need to enforce the constraint that the start index is no greater than the end index. Without the interactions between the endpoints, this can be computed in linear time. Note that this model has the same expressivity as RASOR if the span-level FFNN were removed.

Lastly, we compare with a model using the same architecture as RASOR but is trained with a binary logistic loss rather than a softmax loss over spans (*Span prediction w/ logistic loss* in Table 3.3).

The trend in Table 3.3 shows that the model is better at leveraging the supervision as the learning objective more accurately reflects the fundamental task at hand: determining the best answer span. First, we observe general improvements when using labels that closely align with the task. For example, the labels for *membership prediction* simply happens to provide single contiguous spans in the supervision. The model must consider far more possible answers than it needs to (the power set of all words). The same problem holds for *BIO sequence prediction*– the model must do additional work to learn the semantics of the BIO tags. On the other hand, in RASOR, the semantics of an answer span is naturally encoded by the set of labels.

Second, we observe the importance of allowing interactions between the endpoints using the span-level FFNN. RASOR outperforms the *endpoint prediction* model by 1.1 in exact match, The interaction between endpoints enables RASOR to enforce consistency across its two substructures. While this does not provide improvements for predicting the correct *region* of the answer (captured

Learning objective	EM	F1
Membership prediction	57.9	69.7
BIO sequence prediction	63.9	73.0
Endpoints prediction	65.3	75.1
Span prediction w/ log loss	65.2	73.6

Table 3.3: Results for variations of the model architecture presented in Section 3.2.

by the F1 metric, which drops by 0.2), it is more likely to predict a clean answer span that matches human judgment exactly (captured by the exact-match metric).

3.5 Analysis

Figure 3.2 shows how the performances of RASOR and the endpoint predictor introduced in Section 3.4.2 degrade as the lengths of their predictions increase. The endpoint predictor underpredicts single word answer spans, while overpredicting answer spans with more than 8 words.

Since the endpoints predictor does not explicitly model the interaction between the start and end of any given answer span, it is susceptible to choosing the span start and end points from separate answer candidates. For example, consider the following endpoints prediction that is most different in length from a correct span classification. Here, the span classifier correctly answers the question 'Where did the Meuse flow before the flood?' with 'North Sea' but the endpoints prediction is:

> 'south of today's line Merwede-Oude Maas to the North Sea and formed an archipelago-like estuary with Waal and Lek. This system of numerous bays, estuary-like extended rivers, many islands and constant changes of the coastline, is hard to imagine today. From 1421 to 1904, the Meuse and Waal merged further upstream at Gorinchem to form Merwede. For flood protection reasons, the Meuse was separated from the Waal through a lock and diverted into a new outlet called "Bergse Maas", then Amer and then flows into the former bay Hollands Diep'

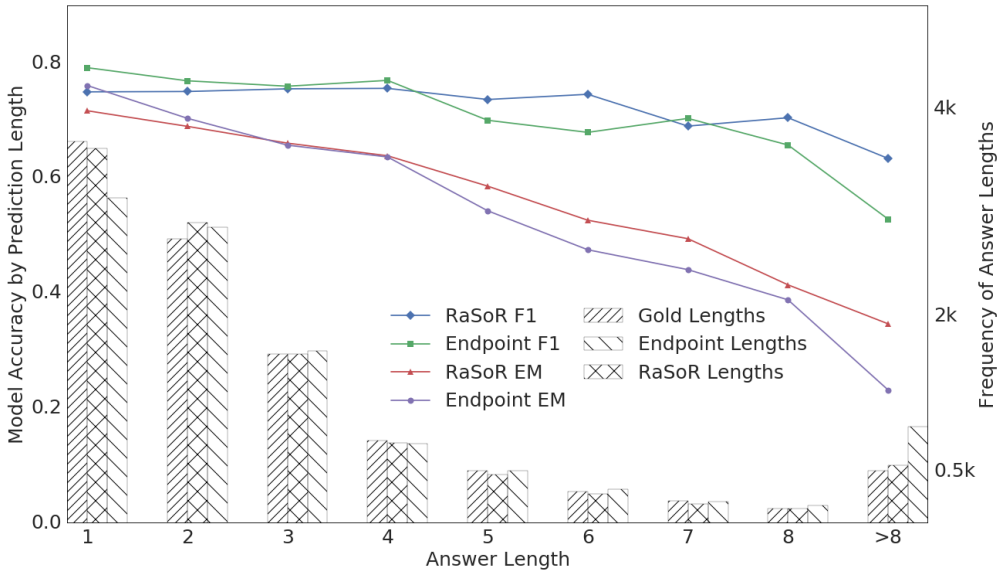


Figure 3.2: F1 and Exact Match accuracy of RASOR and the endpoint predictor over different predictions lengths, along with the distribution of both models' prediction lengths and the gold answer lengths.

In this prediction, we can see that both the start 'south of . . . ' and the end '. . . Hollands Diep' have a reasonable answer type. However, the endpoints predictor has failed to model the fact that they cannot resonably be part of the same answer, a common error case. The endpoints predictor predicts 514 answers with > 25 more words than the gold answer, but the span classifier never does this.

Figure 3.3 shows attention masks for both of RASOR's question representations. The passage-independent question representation pays most attention to the words that could attach to the answer in the passage ('brought', 'against') or describe the answer category ('people'). Meanwhile, the passage-aligned question representation pays attention to similar words. The top predictions for both examples are all valid syntactic constituents, and they all have the correct semantic category. However, RASOR assigns almost as much probability mass to its incorrect third prediction 'British' as it does to the top scoring correct prediction 'Egyptian'. This showcases a common fail-

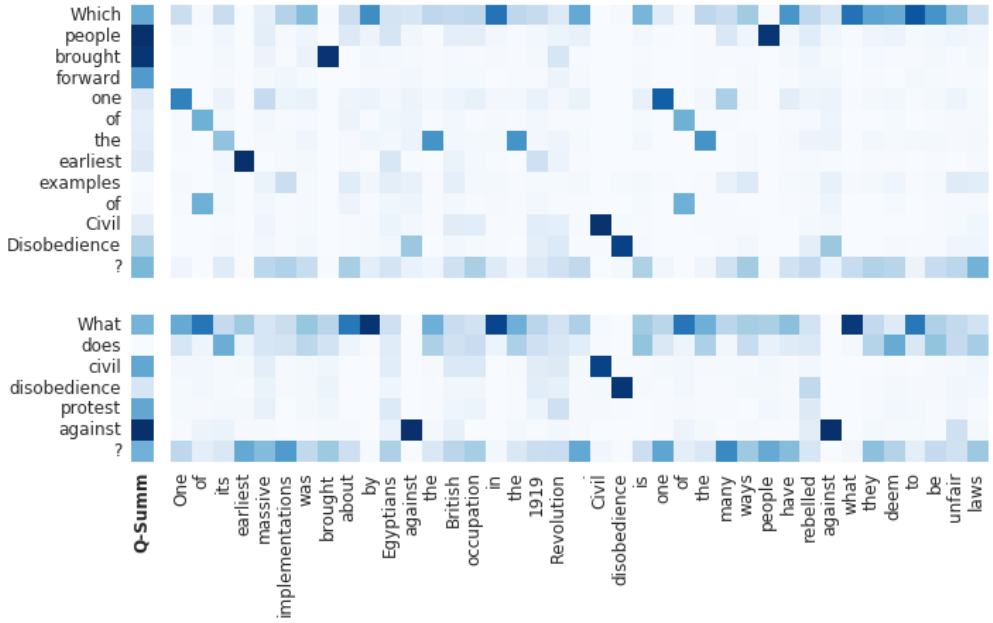


Figure 3.3: Attention masks from RASOR. Top predictions are 'Egyptians', 'Egyptians against the British', and 'British' in the first example and 'unjust laws', 'what they deem to be unjust laws', and 'laws' in the second.

ure case for RASOR, where it can find an answer of the correct type close to a phrase that overlaps with the question – but it cannot accurately represent the semantic dependency on that phrase.

A significant architectural difference from other neural models for the SQUAD dataset, such as Wang and Jiang (2016), is the use of the question-independent passage representation (Equation 3.12). Table 3.4 shows examples in the development set where the model paid the most attention to a single word in the question. The attention mechanism tends to seek words in the question that indicate the answer type, e.g. '**language**' from the question: 'What **language** did the Court of Justice accept ...' This pattern provides insight for the necessity of using *both* question representations, since the answer type information is orthogonal to passage alignment information.

Most attended	Question
conditions	In Gebhard v Consiglio...Milano, the requirements to be registered in Milan before being able to practice law would be allowed under what **conditions**?
Were	**Were** the restored tapes able to have color added to them to enhance the picture or did they remain black and white?
Did	**Did** the European Court of Justice rule the defendant in the case of Commission v. Edith Cresson broke any laws?
whom	The church holds that they are equally bound to respect the sacredness of the life and well-being of **whom**?
Whose	**Whose** thesis states that the solution to a problem is solvable with reasonable resources assuming it allows for a polynomial time algorithm ?
language	What **language** did the Court of Justice accept to be required to teach in a Dublin college in Groner v Minister for Education?
term	Income not from the creation of wealth but by grabbing a larger share of it is know to economists by what **term**?

Table 3.4: Example questions and their most attended words in the passage-independent question representation (Equation 3.11). These examples have the greatest attention (normalized by the question length) in the development set. The attention mechanism typically seeks words in the question that indicate the answer type.

3.6 Conclusion

We have shown a novel approach for performing extractive question answering on the SQuAD dataset by explicitly representing and scoring answer span candidates. The core of our model relies on a recurrent network that enables shared computation for the shared substructure across span candidates. We explore different methods of encoding the passage and question, showing the benefits of including both passage-independent and passage-aligned question representations. While we show that this encoding method is beneficial for the task, this is orthogonal to the core contribution of efficiently computing span representation. In future work, we plan to explore alternate architectures that provide input to the recurrent span representations.

Chapter 4

END-TO-END NEURAL COREFERENCE RESOLUTION

We present the first state-of-the-art neural coreference resolution model that is learned end-to-end given only gold mention clusters. All recent coreference models, including neural approaches that achieved impressive performance gains (Wiseman et al., 2016; Clark and Manning, 2016a,b), rely on syntactic parsers, both for head-word features and as the input to carefully hand-engineered mention proposal algorithms. We demonstrate for the first time that these resources are not required, and in fact performance can be improved significantly without them, by training an end-to-end neural model that jointly learns which spans are entity mentions and how to best cluster them.

Our model reasons over the space of all spans up to a maximum length and directly optimizes the marginal likelihood of antecedent spans from gold coreference clusters. It includes a *span*-ranking model that decides, for each span, which of the previous spans (if any) is a good antecedent. At the core of our model are vector embeddings representing spans of text in the document, which combine context-dependent boundary representations with a head-finding attention mechanism over the span. The attention component is inspired by parser-derived head-word matching features from previous systems (Durrett and Klein, 2013), but is less susceptible to cascading errors. In our analyses, we show empirically that these learned attention weights correlate strongly with traditional headedness definitions.

Scoring all span pairs in our end-to-end model is impractical, since the complexity would be quartic in the document length. Therefore we factor the model over unary mention scores and pairwise antecedent scores, both of which are simple functions of the learned span embedding. The unary mention scores are used to prune the space of spans and antecedents, to aggressively reduce the number of pairwise computations.

Our final approach outperforms existing models by 1.5 F1 on the OntoNotes benchmark and by 3.1 F1 using a 5-model ensemble. It is not only accurate, but also relatively interpretable. The model factors, for example, directly indicate whether an absent coreference link is due to low mention scores (for either span) or a low score from the mention ranking component. The head-finding attention mechanism also reveals which mention-internal words contribute most to coreference decisions. We leverage this overall interpretability to do detailed quantitative and qualitative analyses, providing insights into the strengths and weaknesses of the approach.

4.1 Related Work

Machine learning methods have a long history in coreference resolution (see Ng (2010) for a detailed survey). However, the learning problem is challenging and, until very recently, hand-engineered systems built on top of automatically produced parse trees (Raghunathan et al., 2010) outperformed all learning approaches. Durrett and Klein (2013) showed that highly lexical learning approaches reverse this trend, and more recent neural models (Wiseman et al., 2016; Clark and Manning, 2016a,b) have achieved significant performance gains. However, all of these models use parsers for head features and include highly engineered mention proposal algorithms.[1] Such pipelined systems suffer from two major drawbacks: (1) parsing mistakes can introduce cascading errors and (2) many of the hand-engineered rules do not generalize to new languages.

A non-pipelined system that jointly models mention detection and coreference resolution was first proposed by Daumé III and Marcu (2005a). They introduce a search-based system that predicts the coreference structure in a left-to-right transition system that can incorporate global features. In contrast, our approach performs well while making much stronger independence assumptions, enabling straightforward inference.

More generally, a wide variety of approaches for learning coreference models have been proposed. They can typically be categorized as (1) mention-pair classifiers (Ng and Cardie, 2002; Bengtson and Roth, 2008), (2) entity-level models (Haghighi and Klein, 2010; Clark and Manning,

[1]For example, Raghunathan et al. (2010) use rules to remove pleonastic mentions of *it* detected by 12 lexicalized regular expressions over English parse trees.

2015, 2016a; Wiseman et al., 2016), (3) latent-tree models (Fernandes et al., 2012; Björkelund and Kuhn, 2014; Martschat and Strube, 2015), or (4) mention-ranking models (Durrett and Klein, 2013; Wiseman et al., 2015; Clark and Manning, 2016b). Our span-ranking approach is most similar to mention ranking, but we reason over a larger space by jointly detecting mentions and predicting coreference.

4.2 Task

We formulate the task of end-to-end coreference resolution as a set of decisions for every possible span in the document. The input is a document D containing T words along with metadata such as speaker and genre information.

Let $N = \frac{T(T+1)}{2}$ be the number of possible text spans in D. Denote the start and end indices of a span i in D respectively by START(i) and END(i), for $1 \leq i \leq N$. We assume an ordering of the spans based on START(i); spans with the same start index are ordered by END(i).

The task is to assign to each span i an antecedent y_i. The set of possible assignments for each y_i is $\mathcal{Y}(i) = \{\epsilon, 1, \ldots, i-1\}$, a dummy antecedent ϵ and all preceding spans. True antecedents of span i, i.e. span j such that $1 \leq j \leq i-1$, represent coreference links between i and j. The dummy antecedent ϵ represents two possible scenarios: (1) the span is not an entity mention or (2) the span is an entity mention but it is not coreferent with any previous span. These decisions implicitly define a final clustering, which can be recovered by grouping all spans that are connected by a set of antecedent predictions.

4.3 Model

We aim to learn a conditional probability distribution $P(y_1, \ldots, y_N \mid D)$ whose most likely configuration produces the correct clustering. We use a product of multinomials for each span:

$$\begin{aligned} P(y_1, \ldots, y_N \mid D) &= \prod_{i=1}^{N} P(y_i \mid D) \\ &= \prod_{i=1}^{N} \frac{\exp(s(i, y_i))}{\sum_{y' \in \mathcal{Y}(i)} \exp(s(i, y'))} \end{aligned}$$

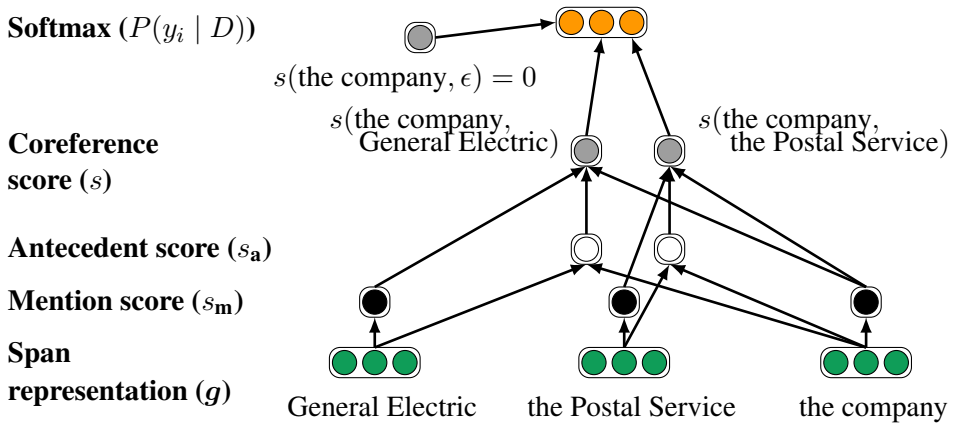


Figure 4.1: Second step of our model. Antecedent scores are computed from pairs of span representations. The final coreference score of a pair of spans is computed by summing the mention scores of both spans and their antecedent score.

where $s(i, j)$ is a pairwise score for a coreference link between span i and span j in document D. We omit the document D from the notation when the context is unambiguous. There are three factors for this pairwise coreference score: (1) whether span i is a mention, (2) whether span j is a mention, and (3) whether j is an antecedent of i:

$$s(i,j) = \begin{cases} 0 & j = \epsilon \\ s_{\mathrm{m}}(i) + s_{\mathrm{m}}(j) + s_{\mathrm{a}}(i,j) & j \neq \epsilon \end{cases}$$

Here $s_{\mathrm{m}}(i)$ is a unary score for span i being a mention, and $s_{\mathrm{a}}(i, j)$ is pairwise score for span j being an antecedent of span i.

By fixing the score of the dummy antecedent ϵ to 0, the model predicts the best scoring antecedent if any non-dummy scores are positive, and it abstains if they are all negative.

A challenging aspect of this model is that its size is $\mathcal{O}(T^4)$ in the document length. As we will see in Section 4.4, the above factoring enables aggressive pruning of spans that are unlikely to belong to a coreference cluster according the mention score $s_{\mathrm{m}}(i)$.

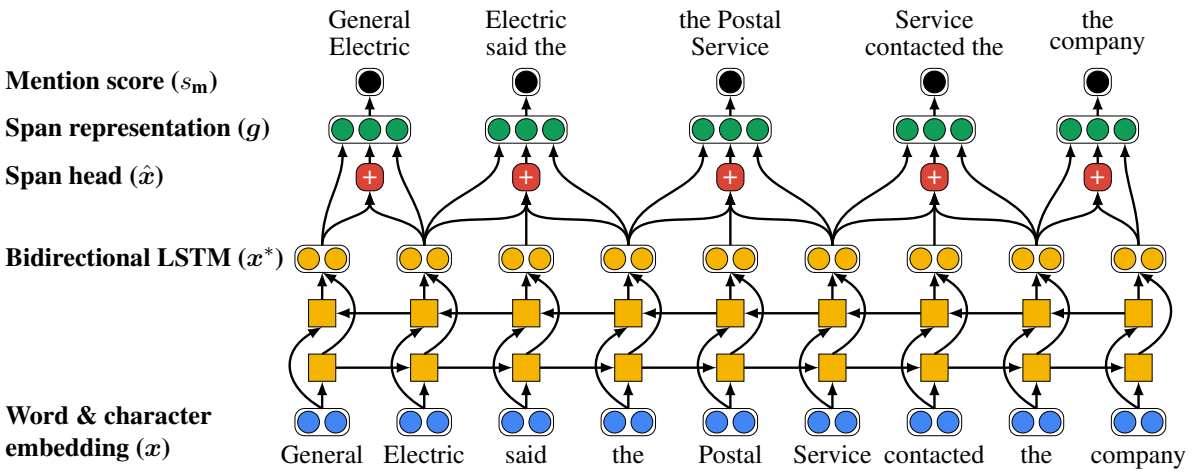


Figure 4.2: First step of the end-to-end coreference resolution model, which computes embedding representations of spans for scoring potential entity mentions. Low-scoring spans are pruned, so that only a manageable number of spans is considered for coreference decisions. In general, the model considers all possible spans up to a maximum width, but we depict here only a small subset.

Scoring Architecture We propose an end-to-end neural architecture that computes the above scores given the document and its metadata.

At the core of the model are vector representations $\boldsymbol{g}_i$ for each possible span i, which we describe in detail in the following section. Given these span representations, the scoring functions above are computed via standard feed-forward neural networks:

$$s_{\text{m}}(i) = \boldsymbol{w}_{\text{m}} \cdot \text{FFNN}_{\text{m}}(\boldsymbol{g}_i)$$

$$s_{\text{a}}(i,j) = \boldsymbol{w}_{\text{a}} \cdot \text{FFNN}_{\text{a}}([\boldsymbol{g}_i, \boldsymbol{g}_j, \boldsymbol{g}_i \circ \boldsymbol{g}_j, \phi(i,j)])$$

where $\cdot$ denotes the dot product, $\circ$ denotes element-wise multiplication, and FFNN denotes a feed-forward neural network that computes a non-linear mapping from input to output vectors.

The antecedent scoring function $s_{\text{a}}(i,j)$ includes explicit element-wise similarity of each span $\boldsymbol{g}_i \circ \boldsymbol{g}_j$ and a feature vector $\phi(i,j)$ encoding speaker and genre information from the metadata and the distance between the two spans.

Span Representations Two types of information are crucial to accurately predicting coreference links: the context surrounding the mention span and the internal structure within the span. We use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to encode the lexical information of both the inside and outside of each span. We also include an attention mechanism over words in each span to model head words.

We assume vector representations of each word $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, which are composed of fixed pre-trained word embeddings and 1-dimensional convolution neural networks (CNN) over characters (see Section 4.6.1 for details)

To compute vector representations of each span, we first use bidirectional LSTMs to encode every word in its context:

$$
\begin{aligned}
\boldsymbol{f}_{t,\delta} &= \sigma(\mathbf{W}_{\mathrm{f}}[\boldsymbol{x}_t, \boldsymbol{h}_{t+\delta,\delta}] + \boldsymbol{b}_{\mathrm{i}}) \\
\boldsymbol{o}_{t,\delta} &= \sigma(\mathbf{W}_{\mathrm{o}}[\boldsymbol{x}_t, \boldsymbol{h}_{t+\delta,\delta}] + \boldsymbol{b}_{\mathrm{o}}) \\
\widetilde{\boldsymbol{c}}_{t,\delta} &= \tanh(\mathbf{W}_{\mathrm{c}}[\boldsymbol{x}_t, \boldsymbol{h}_{t+\delta,\delta}] + \boldsymbol{b}_{\mathrm{c}}) \\
\boldsymbol{c}_{t,\delta} &= \boldsymbol{f}_{t,\delta} \circ \widetilde{\boldsymbol{c}}_{t,\delta} + (\mathbf{1} - \boldsymbol{f}_{t,\delta}) \circ \boldsymbol{c}_{t+\delta,\delta} \\
\boldsymbol{h}_{t,\delta} &= \boldsymbol{o}_{t,\delta} \circ \tanh(\boldsymbol{c}_{t,\delta}) \\
\boldsymbol{x}_t^* &= [\boldsymbol{h}_{t,1}, \boldsymbol{h}_{t,-1}]
\end{aligned}
$$

where $\delta \in \{-1, 1\}$ indicates the directionality of each LSTM, and $\boldsymbol{x}_t^*$ is the concatenated output of the bidirectional LSTM. We use independent LSTMs for every sentence, since cross-sentence context was not helpful in our experiments.

Syntactic heads are typically included as features in previous systems (Durrett and Klein, 2013; Clark and Manning, 2016a,b). Instead of relying on syntactic parses, our model learns a task-specific notion of headedness using an attention mechanism (Bahdanau et al., 2014) over words in

each span:

$$\alpha_t = \boldsymbol{w}_\alpha \cdot \text{FFNN}_\alpha(\boldsymbol{x}^*_t)$$
$$a_{i,t} = \frac{\exp(\alpha_t)}{\sum_{k=\text{START}(i)}^{\text{END}(i)} \exp(\alpha_k)}$$
$$\hat{\boldsymbol{x}}_i = \sum_{t=\text{START}(i)}^{\text{END}(i)} a_{i,t} \cdot \boldsymbol{x}_t$$

where $\hat{\boldsymbol{x}}_i$ is a weighted sum of word vectors in span i. The weights $a_{i,t}$ are automatically learned and correlate strongly with traditional definitions of head words as we will see in Section 4.8.2.

The above span information is concatenated to produce the final representation $\boldsymbol{g}_i$ of span i:

$$\boldsymbol{g}_i = [\boldsymbol{x}^*_{\text{START}(i)}, \boldsymbol{x}^*_{\text{END}(i)}, \hat{\boldsymbol{x}}_i, \phi(i)]$$

This generalizes the recurrent span representations recently proposed for question-answering (Lee et al., 2016b), which only include the boundary representations $\boldsymbol{x}^*_{\text{START}(i)}$ and $\boldsymbol{x}^*_{\text{END}(i)}$. We introduce the soft head word vector $\hat{\boldsymbol{x}}_i$ and a feature vector $\phi(i)$ encoding the size of span i.

4.4 Inference

The size of the full model described above is $\mathcal{O}(T^4)$ in the document length T. To maintain computation efficiency, we prune the candidate spans greedily during both training and evaluation.

We only consider spans with up to L words and compute their unary mention scores $s_\text{m}(i)$ (as defined in Section 4.3). To further reduce the number of spans to consider, we only keep up to λT spans with the highest mention scores and consider only up to K antecedents for each. We also enforce non-crossing bracketing structures with a simple suppression scheme.[2] We accept spans in decreasing order of the mention scores, unless, when considering span i, there exists a previously accepted span j such that $\text{START}(i) < \text{START}(j) \leq \text{END}(i) < \text{END}(j) \vee \text{START}(j) < \text{START}(i) \leq \text{END}(j) < \text{END}(i)$.

[2]The official CoNLL-2012 evaluation only considers predictions without crossing mentions to be valid. Enforcing this consistency is not inherently necessary in our model.

Despite these aggressive pruning strategies, we maintain a high recall of gold mentions in our experiments (over 92% when $\lambda = 0.4$).

For the remaining mentions, the joint distribution of antecedents for each document is computed in a forward pass over a single computation graph. The final prediction is the clustering produced by the most likely configuration.

4.5 Learning

In the training data, only clustering information is observed. Since the antecedents are latent, we optimize the marginal log-likelihood of all correct antecedents implied by the gold clustering:

$$\log \prod_{i=1}^{N} \sum_{\hat{y} \in \mathcal{Y}(i) \cap \text{GOLD}(i)} P(\hat{y})$$

where $\text{GOLD}(i)$ is the set of spans in the gold cluster containing span i. If span i does not belong to a gold cluster or all gold antecedents have been pruned, $\text{GOLD}(i) = \{\epsilon\}$.

By optimizing this objective, the model naturally learns to prune spans accurately. While the initial pruning is completely random, only gold mentions receive positive updates. The model can quickly leverage this learning signal for appropriate credit assignment to the different factors, such as the mention scores s_m used for pruning.

Fixing score of the dummy antecedent to zero removes a spurious degree of freedom in the overall model with respect to mention detection. It also prevents the span pruning from introducing noise. For example, consider the case where span i has a single gold antecedent that was pruned, so $\text{GOLD}(i) = \{\epsilon\}$. The learning objective will only correctly push the scores of non-gold antecedents lower, and it cannot incorrectly push the score of the dummy antecedent higher.

This learning objective can be considered a span-level, cost-insensitive analog of the learning objective proposed by Durrett and Klein (2013). We experimented with these cost-sensitive alternatives, including margin-based variants (Wiseman et al., 2015; Clark and Manning, 2016b), but a simple maximum-likelihood objective proved to be most effective.

	MUC			B^3			$CEAF_{\phi_4}$			
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Avg. F1
Our model (ensemble)	**81.4**	**73.3**	**77.1**	**72.6**	**61.5**	**66.6**	**65.7**	**59.9**	**62.6**	**68.8**
Our model (single)	78.8	72.7	75.6	68.9	60.8	64.6	62.9	58.1	60.4	66.9
Clark and Manning (2016b)	79.2	70.4	74.6	69.9	58.0	63.4	63.5	55.5	59.2	65.7
Clark and Manning (2016a)	79.9	69.3	74.2	71.0	56.5	63.0	63.8	54.3	58.7	65.3
Wiseman et al. (2016)	77.5	69.8	73.4	66.8	57.0	61.5	62.1	53.9	57.7	64.2
Wiseman et al. (2015)	76.2	69.3	72.6	66.2	55.8	60.5	59.4	54.9	57.1	63.4
Clark and Manning (2015)	76.1	69.4	72.6	65.6	56.0	60.4	59.4	53.0	56.0	63.0
Martschat and Strube (2015)	76.7	68.1	72.2	66.1	54.2	59.6	59.5	52.3	55.7	62.5
Durrett and Klein (2014)	72.6	69.9	71.2	61.2	56.4	58.7	56.2	54.2	55.2	61.7
Björkelund and Kuhn (2014)	74.3	67.5	70.7	62.7	55.0	58.6	59.4	52.3	55.6	61.6
Durrett and Klein (2013)	72.9	65.9	69.2	63.6	52.5	57.5	54.3	54.4	54.3	60.3

Table 4.1: Results on the test set on the English data from the CoNLL-2012 shared task. The final column (Avg. F1) is the main evaluation metric, computed by averaging the F1 of MUC, B^3, and $CEAF_{\phi_4}$. We improve state-of-the-art performance by 1.2 F1 for the single model and by 3.1 F1.

4.6 Experiments

We use the English coreference resolution data from the CoNLL-2012 shared task (Pradhan et al., 2012) in our experiments. This dataset contains 2802 training documents, 343 development documents, and 348 test documents. The training documents contain on average 454 words and a maximum of 4009 words.

4.6.1 Hyperparameters

Word representations The word embeddings are a fixed concatenation of 300-dimensional GloVe embeddings (Pennington et al., 2014) and 50-dimensional embeddings from Turian et al. (2010), both normalized to be unit vectors. Out-of-vocabulary words are represented by a vector of zeros. In the character CNN, characters are represented as learned 8-dimensional embeddings. The convolutions have window sizes of 3, 4, and 5 characters, each consisting of 50 filters.

Hidden dimensions The hidden states in the LSTMs have 200 dimensions. Each feed-forward neural network consists of two hidden layers with 150 dimensions and rectified linear units (Nair and Hinton, 2010).

Feature encoding We encode speaker information as a binary feature indicating whether a pair of spans are from the same speaker. Following Clark and Manning (2016a), the distance features are binned into the following buckets [1, 2, 3, 4, 5-7, 8-15, 16-31, 32-63, 64+]. All features (speaker, genre, span distance, mention width) are represented as learned 20-dimensional embeddings.

Pruning We prune the spans such that the maximum span width $L = 10$, the number of spans per word $\lambda = 0.4$, and the maximum number of antecedents $K = 250$. During training, documents are randomly truncated to up to 50 sentences.

Learning We use ADAM (Kingma and Ba, 2014) for learning with a minibatch size of 1. The LSTM weights are initialized with random orthonormal matrices as described in Saxe et al. (2013). We apply 0.5 dropout to the word embeddings and character CNN outputs. We apply 0.2 dropout to all hidden layers and feature embeddings. Dropout masks are shared across timesteps to preserve long-distance information as described in Gal and Ghahramani (2016b). The learning rate is decayed by 0.1% every 100 steps. The model is trained for up to 150 epochs, with early stopping based on the development set.

All code is implemented in TensorFlow (Abadi et al., 2015) and is publicly available. [3]

4.6.2 Ensembling

We also report ensemble experiments using five models trained with different random initializations. Ensembling is performed for both the span pruning and antecedent decisions.

At test time, we first average the mention scores $s_{\mathrm{m}}(i)$ over each model before pruning the spans. Given the same pruned spans, each model then computes the antecedent scores $s_{\mathrm{a}}(i, j)$ separately, and they are averaged to produce the final scores.

4.7 Results

We report the precision, recall, and F1 for the standard MUC, B^3, and $CEAF_{\phi_4}$metrics using the official CoNLL-2012 evaluation scripts. The main evaluation is the average F1 of the three metrics.

4.7.1 Coreference Results

Table 4.1 compares our model to several previous systems that have driven substantial improvements over the past several years on the OntoNotes benchmark. We outperform previous systems in all metrics. In particular, our single model improves the state-of-the-art average F1 by 1.5, and our 5-model ensemble improves it by 3.1.

The most significant gains come from improvements in recall, which is likely due to our end-to-end setup. During training, pipelined systems typically discard any mentions that the mention detector misses, which for Clark and Manning (2016b) consists of more than 9% of the labeled mentions in the training data. In contrast, we only discard mentions that exceed our maximum mention width of 10, which accounts for less than 2% of the training mentions. The contribution of joint mention scoring is further discussed in Section 4.7.3

	Avg. F1	Δ
Our model (ensemble)	68.8	+1.4
Our model (single)	67.4	
– distance and width features	63.5	-3.9
– GloVe embeddings	64.9	-2.5
– speaker and genre metadata	65.9	-1.5
– head-finding attention	66.1	-1.3
– character CNN	66.4	-1.0
– Turian embeddings	66.8	-0.6

Table 4.2: Comparisons of our single model on the development data. The 5-model ensemble provides a 1.4 F1 improvement. The head-finding attention, features, and components of the word representations all contribute to the full model.

4.7.2 Ablations

To show the importance of each component in our proposed model, we ablate various parts of the architecture and report the average F1 on the development set of the data (see Figure 4.2).

Features The distance between spans and the width of spans are crucial signals for coreference resolution, consistent with previous findings from other coreference models. They contribute 3.8 F1 to the final result.

Word representations Since our word embeddings are fixed, having access to a variety of word embeddings allows for a more expressive model without overfitting. We hypothesis that the different learning objectives of the GloVe and Turian embeddings provide orthogonal information

[3] `https://github.com/kentonl/e2e-coref`

	Avg. F1	Δ
Our model (joint mention scoring)	67.7	
w/ rule-based mentions	66.7	-1.0
w/ oracle mentions	85.2	+17.5

Table 4.3: Comparisons of of various mention proposal methods with our model on the development data. The rule-based mentions are derived from the mention detector from Raghunathan et al. (2010), resulting in a 1 F1 drop in performance. The oracle mentions are from the labeled clusters and improve our model by over 17.5 F1.

(the former is word-order insensitive while the latter is word-order sensitive). Both embeddings contribute to some improvement in development F1.

The character CNN provides morphological information and a way to backoff for out-of-vocabulary words. Since coreference decisions often involve rare named entities, we see a contribution of 0.9 F1 from character-level modeling.

Metadata Speaker and genre indicators many not be available in downstream applications. We show that performance degrades by 1.4 F1 without them, but is still on par with previous state-of-the-art systems that assume access to this metadata.

Head-finding attention Ablations also show a 1.3 F1 degradation in performance without the attention mechanism for finding task-specific heads. As we will see in Section 4.8.4, the attention mechanism should not be viewed as simply an approximation of syntactic heads. In many cases, it is beneficial to pay attention to multiple words that are useful specifically for coreference but are not traditionally considered to be syntactic heads.

4.7.3 Comparing Span Pruning Strategies

To tease apart the contributions of improved mention scoring and improved coreference decisions, we compare the results of our model with alternate span pruning strategies. In these experiments, we use the alternate spans for both training and evaluation. As shown in Table 4.3, keeping mention candidates detected by the rule-based system over predicted parse trees (Raghunathan et al., 2010) degrades performance by 1 F1. We also provide oracle experiment results, where we keep exactly the mentions that are present in gold coreference clusters. With oracle mentions, we see an improvement of 17.5 F1, suggesting an enormous room for improvement if our model can produce better mention scores and anaphoricity decisions.

4.8 Analysis

To highlight the strengths and weaknesses of our model, we provide both quantitative and qualitative analyses. In the following discussion, we use predictions from the single model rather than the ensembled model.

4.8.1 Mention Recall

The training data only provides a weak signal for spans that correspond to entity mentions, since singleton clusters are not explicitly labeled. As a by product of optimizing marginal likelihood, our model automatically learns a useful ranking of spans via the unary mention scores from Section 4.3.

The top spans, according to the mention scores, cover a large portion of the mentions in gold clusters, as shown in Figure 4.3. Given a similar number of spans kept, our recall is comparable to the rule-based mention detector (Raghunathan et al., 2010) that produces 0.26 spans per word with a recall of 89.2%. As we increase the number of spans per word (λ in Section 4.4), we observe higher recall but with diminishing returns. In our experiments, keeping 0.4 spans per word results in 92.7% recall in the development data.

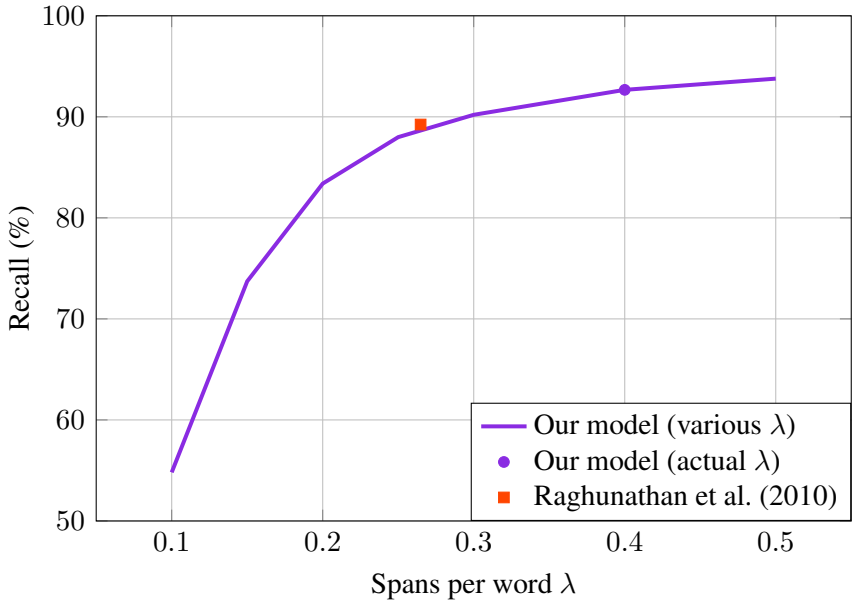


Figure 4.3: Proportion of gold mentions covered in the development data as we increase the number of spans kept per word. Recall is comparable to the mention detector of previous state-of-the-art systems given the same number of spans. Our model keeps 0.4 spans per word in our experiments, achieving 92.7% recall of gold mentions.

4.8.2 Mention Precision

While the training data does not offer a direct measure of mention precision, we can use the gold syntactic structures provided in the data as a proxy. Spans with high mention scores should correspond to syntactic constituents.

In Figure 4.4, we show the precision of top-scoring spans when keeping 0.4 spans per word. For spans with 2–5 words, 75–90% of the predictions are constituents, indicating that the vast majority of the mentions are syntactically plausible. Longer spans, which are all relatively rare, prove more difficult for the model, and precision drops to 46% for spans with 10 words.

1	**(A fire in a Bangladeshi garment factory)** has left at least 37 people dead and 100 hospitalized. Most of the deceased were killed in the crush as workers tried to flee **(the blaze)** in the four-story building.
	A fire in **(a Bangladeshi garment factory)** has left at least 37 people dead and 100 hospitalized. Most of the deceased were killed in the crush as workers tried to flee the blaze in **(the four-story building)**.
2	We are looking for **(a region of central Italy bordering the Adriatic Sea)**. **(The area)** is mostly mountainous and includes Mt. Corno, the highest peak of the Apennines. **(It)** also includes a lot of sheep, good clean-living, healthy sheep, and an Italian entrepreneur has an idea about how to make a little money of them.
3	**(The flight attendants)** have until 6:00 today to ratify labor concessions. **(The pilots')** union and ground crew did so yesterday.
4	**(Prince Charles and his new wife Camilla)** have jumped across the pond and are touring the United States making **(their)** first stop today in New York. It's Charles' first opportunity to showcase his new wife, but few Americans seem to care. Here's Jeanie Mowth. What a difference two decades make. **(Charles and Diana)** visited a JC Penney's on the prince's last official US tour. Twenty years later here's the prince with his new wife.
5	Also such location devices, **(some ships)** have smoke floats **(they)** can toss out so the man overboard will be able to use smoke signals as a way of trying to, let the rescuer locate **(them)**.

Table 4.4: Examples predictions from the development data. Each row depicts a single coreference cluster predicted by our model. Bold, parenthesized spans indicate mentions in the predicted cluster. The redness of each word indicates the weight of the head-finding attention mechanism ($a_{i,t}$ in Section 4.3).

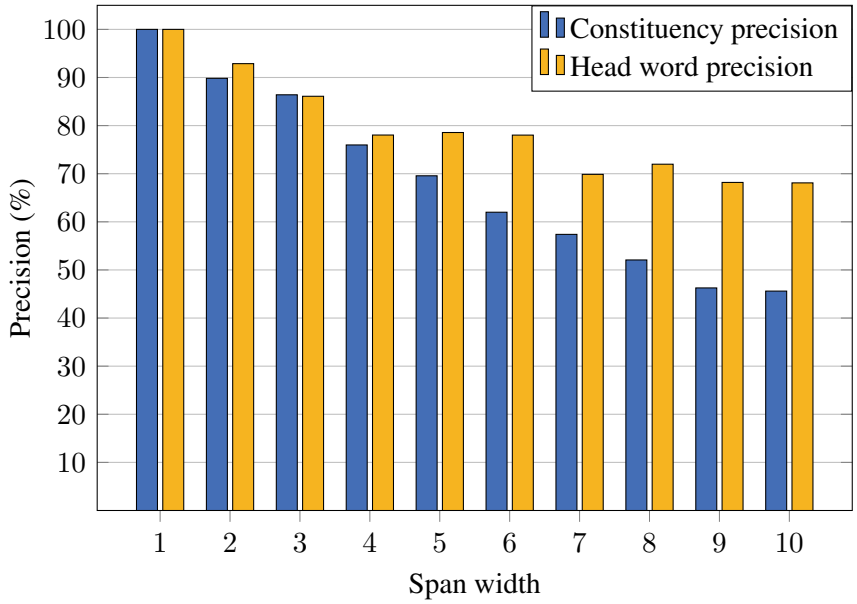


Figure 4.4: Indirect measure of mention precision using agreement with gold syntax. Blue bars show the proportion of unpruned spans that match syntactic constituents. Yellow bars show the proportion of unpruned constituents whose syntactic head word matches the most attended word from the learned head-finding attention mechanism.

4.8.3 Head Agreement

We also investigate how well the learned head preferences correlate with syntactic heads. For each of the top-scoring spans in the development data that correspond to gold constituents, we compute the word with the highest attention weight.

We plot in Figure 4.4 the proportion of these words that match syntactic heads. Agreement ranges between 68-93%, which is surprisingly high, since no explicit supervision of syntactic heads is provided. The model simply learns from the clustering data that these head words are useful for making coreference decisions.

4.8.4 Qualitative Analysis

Our qualitative analysis in Table 4.4 highlights the strengths and weaknesses of our model. Each row is a visualization of a single coreference cluster predicted by the model. Bolded spans in parentheses belong to the predicted cluster, and the redness of a word indicates its weight from the head-finding attention mechanism ($a_{i,t}$ in Section 4.3).

Strengths The effectiveness of the attention mechanism for making coreference decisions can be seen in Example 1. The model pays attention to *fire* in the span *A fire in a Bangladeshi garment factory*, allowing it to successfully predict the coreference link with *the blaze*. For a subspan of that mention, *a Bangladeshi garment factory*, the model pays most attention instead to *factory*, allowing it successfully predict the coreference link with *the four-story building*.

The task-specific nature of the attention mechanism is also illustrated in Example 4. The model generally pays attention to coordinators more than the content of the coordination, since coordinators, such as *and*, provide strong cues for plurality.

The model is capable of detecting relatively long and complex noun phrases, such as *a region of central Italy bordering the Adriatic Sea* in Example 2. It also appropriately pays attention to *region*, showing that the attention mechanism provides more than content-word classification. The context encoding provided by the bidirectional LSTMs is critical to making informative head word decisions.

Weaknesses A benefit of using neural models for coreference resolution is their ability to use word embeddings to capture similarity between words, a property that many traditional feature-based models lack. While this can dramatically increase recall, as demonstrated in Example 1, it is also prone to predicting false positive links when the model conflates paraphrasing with relatedness or similarity. In Example 3, the model mistakenly predicts a link between *The flight attendants* and *The pilots'*. The predicted head words *attendants* and *pilots* likely have nearby word embeddings, which is a signal used—and often overused—by the model. The same type of error is made in Example 4, where the model predicts a coreference link between *Prince Charles and his new*

wife Camilla and *Charles and Diana*, two non-coreferent mentions that are similar in many ways. These mistakes suggest substantial room for improvement with word or span representations that can cleanly distinguish between equivalence, entailment, and alternation.

Unsurprisingly, our model does little in the uphill battle of making coreference decisions requiring world knowledge. In Example 5, the model incorrectly decides that *them* (in the context of *let the rescuer locate them*) is coreferent with *some ships*, likely due to plurality cues. However, an ideal model that uses common-sense reasoning would instead correctly infer that a rescuer is more likely to look for *the man overboard* rather than the ship from which he fell. This type of reasoning would require either (1) models that integrate external sources of knowledge with more complex inference or (2) a vastly larger corpus of training data to overcome the sparsity of these patterns.

4.9 Conclusion

We presented a state-of-the-art coreference resolution model that is trained end-to-end for the first time. Our final model ensemble improves performance on the OntoNotes benchmark by over 3 F1 without external preprocessing tools used by previous systems. We showed that our model implicitly learns to generate useful mention candidates from the space of all possible spans. A novel head-finding attention mechanism also learns a task-specific preference for head words, which we empirically showed correlate strongly with traditional head-word definitions.

While our model substantially pushes the state-of-the-art performance, the improvements are potentially complementary to a large body of work on various strategies to improve coreference resolution, including entity-level inference and incorporating world knowledge, which are important avenues for future work.

Chapter 5

HIGHER-ORDER COREFERENCE RESOLUTION WITH COARSE-TO-FINE INFERENCE

5.1 Introduction

Recent coreference resolution systems have heavily relied on first order models (Clark and Manning, 2016b; Lee et al., 2017), where only pairs of entity mentions are scored by the model. These models are computationally efficient and scalable to long documents. However, because they make independent decisions about coreference links, they are susceptible to predicting clusters that are locally consistent but globally inconsistent. Figure 5.1 shows an example from Wiseman et al. (2016) that illustrates this failure case. The plurality of **[you]** is underspecified, making it locally compatible with both **[I]** and **[all of you]**, while the full cluster would have mixed plurality, resulting in global inconsistency.

We introduce an approximation of higher-order inference by iteratively refining span representations using the span-ranking architecture from Lee et al. (2017). To alleviate computational challenges from this higher-order inference, we also introduce a coarse-to-fine approach that uses a less accurate but more efficient bilinear factor to prune the set of possible antecedents.

Our experiments show that both of the above contributions improve the performance of coreference resolution on the English OntoNotes benchmark. We observe a significant increase in average F1 with a second-order model, but returns quickly diminish with a third-order model. Additionally, our analysis shows that the coarse-to-fine approach makes the model performance relatively insensitive to more aggressive antecedent pruning, compared to the distance-based heuristic pruning from previous work.

Speaker 1: Um and **[I]** think that is what's - Go ahead Linda.
Speaker 2: Well and uh thanks goes to **[you]** and to the media to help us... So our hat is off to **[all of you]** as well.

Figure 5.1: Example of consistency errors to which first-order span-ranking models are susceptible. Span pairs (**I**, **you**) and (**you**, **all of you**) are locally consistent, but the span triplet (**I**, **you**, **all of you**) is globally inconsistent. Avoiding this error requires modeling higher-order structures.

5.2 Background

Task definition We formulate the coreference resolution tasks as a set of antecedent assignments y_i for each of span i in the given document, following Lee et al. (2017). The set of possible assignments for each y_i is $\mathcal{Y}(i) = \{\epsilon, 1, \ldots, i-1\}$, a dummy antecedent ϵ and all preceding spans. Non-dummy antecedents represent coreference links between i and y_i. The dummy antecedent ϵ represents two possible scenarios: (1) the span is not an entity mention or (2) the span is an entity mention but it is not coreferent with any previous span. These decisions implicitly define a final clustering, which can be recovered by grouping all spans that are connected by antecedent predictions.

Baseline We describe the baseline model (Lee et al., 2017), which we will improve to address the modeling and computational limitations discussed previously. The goal is to learn a distribution $P(y_i)$ over antecedents for each span i :

$$P(y_i) = \frac{e^{s(i,y_i)}}{\sum_{y' \in \mathcal{Y}(i)} e^{s(i,y')}}$$

where $s(i, j)$ is a pairwise score for a coreference link between span i and span j. The baseline model includes three factors for this pairwise coreference score: (1) $s_{\text{m}}(i)$, whether span i is a mention, (2) $s_{\text{m}}(j)$, whether span j is a mention, and (3) $s_{\text{a}}(i, j)$ whether j is an antecedent of i:

$$s(i, j) = s_{\text{m}}(i) + s_{\text{m}}(j) + s_{\text{a}}(i, j)$$

In the special case of the dummy antecedent, the score $s(i, \epsilon)$ is instead fixed to 0. A common component used throughout the model is the vector representations $\boldsymbol{g}_i$ for each possible span i. These are computed via bidirectional LSTMs (Hochreiter and Schmidhuber, 1997) that learn context-dependent boundary and head representations. The scoring functions take these span representations as input:

$$s_{\mathrm{m}}(i) = \boldsymbol{w}_{\mathrm{m}}^{\top} \mathrm{FFNN}_{\mathrm{m}}(\boldsymbol{g}_i)$$
$$s_{\mathrm{a}}(i, j) = \boldsymbol{w}_{\mathrm{a}}^{\top} \mathrm{FFNN}_{\mathrm{a}}([\boldsymbol{g}_i, \boldsymbol{g}_j, \boldsymbol{g}_i \circ \boldsymbol{g}_j, \phi(i, j)])$$

where $\circ$ denotes element-wise multiplication, FFNN denotes a feed-forward neural network, and the antecedent scoring function $s_{\mathrm{a}}(i, j)$ includes explicit element-wise similarity of each span $\boldsymbol{g}_i \circ \boldsymbol{g}_j$ and a feature vector $\phi(i, j)$ encoding speaker and genre information from the metadata and the distance between the two spans.

The model above is factored to enable a two-stage beam search. A beam of up to M potential mentions is computed (where M is proportional to the document length) based on the spans with the highest mention scores $s_{\mathrm{m}}(i)$.

Given supervision of gold coreference clusters, the model is learned by optimizing the marginal log-likelihood of the possibly correct antecedents. This marginalization is required since the best antecedent for each span is a latent variable.

5.3 Higher-order Coreference Resolution

The baseline above is a first-order model, since it only considers pairs of spans. First-order models are susceptible to consistency errors demonstrated in Figure 5.1 that can only be detected when considering higher-order structures.

We propose an inference procedure that allows the model to condition on higher-order structures, while being fully differentiable. This inference involves N iterations of refining span representations, denoted as $\boldsymbol{g}_i^n$ for the representation of span i at iteration n. The baseline model is used to initialize the span representation at $\boldsymbol{g}_i^1$. The refined span representations allow the model to also

iteratively refine the antecedent distributions $P_n(y_i)$:

$$P_n(y_i) = \frac{e^{s(\boldsymbol{g}_i^n, \boldsymbol{g}_{y_i}^n))}}{\sum_{y \in \mathcal{Y}(i)} e^{s(\boldsymbol{g}_i^n, \boldsymbol{g}_y^n))}}$$

where s is the coreference scoring function from the baseline, which is given a different span representation at every iteration.

At each iteration, we first compute the expected antecedent representation $\boldsymbol{a}_i^n$ of each span i by using the current antecedent distribution $P_n(y_i)$ as an attention mechanism:

$$\boldsymbol{a}_i^n = \sum_{y_i \in \mathcal{Y}(i)} P_n(y_i) \cdot \boldsymbol{g}_y^n$$

The current span representation $\boldsymbol{g}_i^n$ is then updated via interpolation with its expected antecedent representation $\boldsymbol{a}_i^n$:

$$\boldsymbol{f}_i^n = \sigma(\mathbf{W}_\mathrm{f}[\boldsymbol{g}_i^n, \boldsymbol{a}_i^n])$$
$$\boldsymbol{g}_i^{n+1} = \boldsymbol{f}_i^n \circ \boldsymbol{g}_i^n + (\mathbf{1} - \boldsymbol{f}_i^n) \circ \boldsymbol{a}_i^n$$

The learned gate vector $\boldsymbol{f}_i^n$ determines for each dimension whether to keep the current span information or to integrate new information from its expected antecedent. At iteration n, $\boldsymbol{g}_i^n$ is an element-wise weighted average of approximately n span representations (assuming $P_n(y_i)$ is peaked), allowing $P_n(y_i)$ to softly condition on up to n other spans in the predicted cluster.

Span-ranking can be viewed as predicting latent antecedent trees (Fernandes et al., 2012; Martschat and Strube, 2015), where the predicted antecedent is the parent of a span and each tree is a predicted cluster. By iteratively refining the span representations and antecedent distributions, another way to interpret this model is that the joint distribution $\prod_i P_N(y_i)$ implicitly models every directed path of up to length $N + 1$ in the latent antecedent tree.

5.4 Coarse-to-fine Inference

The model described above scales poorly to long documents. Despite heavy pruning of potential mentions, the space of possible antecedents for every surviving span is still too large to fully

consider. The bottleneck is in the antecedent score $s_{\text{a}}(i, j)$, which requires computing a tensor of size $M \times M \times (3|\boldsymbol{g}| + |\phi|)$. This computational challenge is even more problematic with the iterative inference from Section 5.3, which requires recomputing this tensor at every iteration.

Heuristic antecedent pruning To reduce computation, Lee et al. (2017) heuristically consider only the nearest K antecedents of each span, resulting in a smaller input of size $M \times K \times (3|\boldsymbol{g}| + |\phi|)]$. The main drawback to this solution is that it imposes an a priori limit on the maximum distance of a coreference link, which can be quite large in natural language discourse.

Coarse-to-fine antecedent pruning We instead propose a coarse-to-fine approach that can be learned end-to-end and does not establish an a priori maximum coreference distance. The key component of this coarse-to-fine approach is an alternate bilinear scoring function:

$$s_{\text{c}}(i, j) = \boldsymbol{g}_i^{\top} \mathbf{W}_{\text{c}} \, \boldsymbol{g}_j$$

In contrast to the feed-forward $s_{\text{a}}(i, j)$, the bilinear $s_{\text{c}}(i, j)$ is far less accurate. A direct replacement of $s_{\text{a}}(i, j)$ with $s_{\text{c}}(i, j)$ results in a performance loss of over 3 F1 on the development set. However, $s_{\text{c}}(i, j)$ is much more efficient to compute. Computing $s_{\text{c}}(i, j)$ only requires manipulating matrices of size $M \times |\boldsymbol{g}|$.

We leverage this more efficient but less-accurate coarse score without loss in accuracy by instead including it as an additional factor in the model:

$$s(i, j) = s_{\text{m}}(i) + s_{\text{m}}(j) + s_{\text{c}}(i, j) + s_{\text{a}}(i, j)$$

With this additional factor, more efficient inference can be achieved with three-stage beam search. In the first stage, we keep the top M spans based on mention scores. In the second stage, we keep the top K antecedents of each remaining span i based on the first three factors, $s_{\text{m}}(i) + s_{\text{m}}(j) + s_{\text{c}}(i, j)$. In the third and final stage, the overall coreference $s(i, j)$ is computed based on the remaining span pairs. This coarse-to-fine approach effectively expands the set of coreference links that the model is capable of learning and can achieve better performance while using a much smaller K (see Figure 5.2).

	MUC			B^3			$CEAF_{\phi_4}$			
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Avg. F1
Martschat and Strube (2015)	76.7	68.1	72.2	66.1	54.2	59.6	59.5	52.3	55.7	62.5
Clark and Manning (2015)	76.1	69.4	72.6	65.6	56.0	60.4	59.4	53.0	56.0	63.0
Wiseman et al. (2015)	76.2	69.3	72.6	66.2	55.8	60.5	59.4	54.9	57.1	63.4
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 (2016a)	79.9	69.3	74.2	71.0	56.5	63.0	63.8	54.3	58.7	65.3
Clark and Manning (2016b)	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
+ ELMo	80.1	77.2	78.6	69.8	66.5	68.1	66.4	62.9	64.6	70.4
+ hyperparameter tuning	80.7	78.8	79.8	71.7	68.7	70.2	67.2	66.8	67.0	72.3
+ coarse-to-fine inference	80.4	**79.9**	80.1	71.0	**70.0**	70.5	67.5	**67.2**	67.3	72.6
+ second-order inference	**81.4**	79.5	**80.4**	**72.2**	69.5	**70.8**	**68.2**	67.1	**67.6**	**73.0**

Table 5.1: Results on the test set on the English CoNLL-2012 shared task. The average F1 of MUC, B^3, and $CEAF_{\phi_4}$is the main evaluation metric. We show only non-ensembled models for fair comparison.

5.5 Experimental Setup

We use the English coreference resolution data from the CoNLL-2012 shared task (Pradhan et al., 2012) in our experiments.

Our models reuse the hyperparameters from Lee et al. (2017), with a few exceptions mentioned below. In our results, we report two improvements that are orthogonal to our contributions:

- including embedding representations from a language model (Peters et al., 2018) at the input to the LSTMs (ELMo in the results).

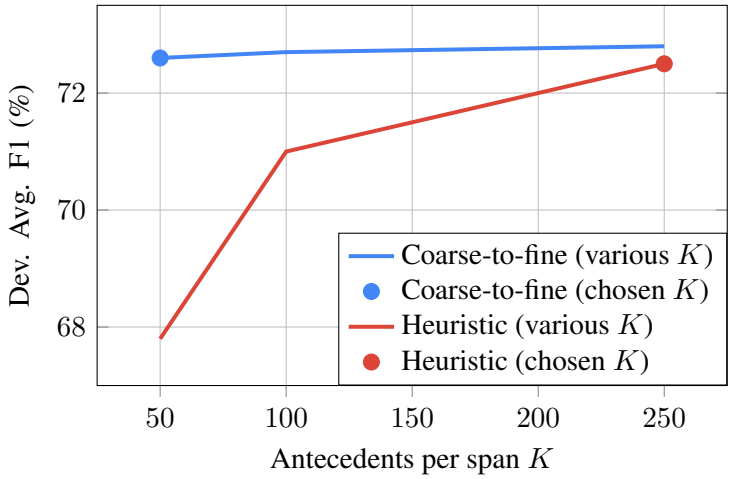


Figure 5.2: Comparison of accuracy on the development set for the two antecedent pruning strategies with various beams sizes K. The distance-based heuristic pruning performance drops by almost 5 F1 when reducing K from 250 to 50, while the coarse-to-fine pruning results in an insignificant drop of less than 0.2 F1.

- improving several hyperparameters:

    1. increasing the maximum span width from 10 to 30 words.
    2. using 3 highway LSTMs instead of 1.
    3. using GloVe word embeddings (Pennington et al., 2014) with a window size of 2 for the head word embeddings and a window size of 10 for the LSTM inputs.

The baseline model considers up to 250 antecedents per span. As shown in Figure 5.2, the coarse-to-fine model is quite insensitive to more aggressive pruning. Therefore, our final model considers only 50 antecedents per span.

On the development set, the second-order model ($N = 2$) outperforms the first-order model by 0.8 F1, but the third order model only provides an additional 0.1 F1 improvement. Therefore, we only report test results for the second-order model.

5.6 Results

We report the precision, recall, and F1 of the the MUC, B^3, and $CEAF_{\phi_4}$ metrics using the official CoNLL-2012 evaluation scripts. The main evaluation is the average F1 of the three metrics.

Results on the test set are shown in Table 5.1. We include performance of systems proposed in the past 3 years for reference. The baseline relative to our contributions is the span-ranking model from Lee et al. (2017) augmented with both ELMo and hyperparameter tuning, which achieves 72.3 F1. Our full approach achieves 73.0 F1, setting a new state of the art for coreference resolution.

Compared to the heuristic pruning with up to 250 antecedents, our coarse-to-fine model only computes the expensive scores $s_{\mathrm{a}}(i, j)$ for 50 antecedents. Despite using far less computation, it outperforms the baseline because the coarse scores $s_{\mathrm{c}}(i, j)$ can be computed for all antecedents, enabling the model to potentially predict a coreference link between any two spans in the document. As a result, we observe a much higher recall when adopting the coarse-to-fine approach.

We also observe another improvement by including the second-order inference (Section 5.3). The improvement is largely driven by the overall increase in precision, which is expected since the higher-order inference mainly serves to rule out inconsistent clusters. It is also consistent with findings from Martschat and Strube (2015) who report mainly improvements in precision when modeling latent trees to achieve a similar goal.

5.7 Related Work

In addition to the end-to-end span-ranking model (Lee et al., 2017) that our proposed model builds upon, there is a large body of literature on coreference resolvers that fundamentally rely on scoring span pairs (Ng and Cardie, 2002; Bengtson and Roth, 2008; Denis and Baldridge, 2008; Fernandes et al., 2012; Durrett and Klein, 2013; Wiseman et al., 2015; Clark and Manning, 2016b).

Motivated by structural consistency issues discussed above, significant effort has also been devoted towards cluster-level modeling. Since global features are notoriously difficult to define (Wiseman et al., 2016), they often depend heavily on existing pairwise features or architectures (Björkelund and Kuhn, 2014; Clark and Manning, 2015, 2016a). We similarly use an existing

pairwise span-ranking architecture as a building block for modeling more complex structures. In contrast to Wiseman et al. (2016) who use highly expressive recurrent neural networks to model clusters, we show that the addition of a relatively lightweight gating mechanism is sufficient to effectively model higher-order structures.

5.8 Conclusion

We presented a state-of-the-art coreference resolution system that models higher order interactions between spans in predicted clusters. Additionally, our proposed coarse-to-fine approach alleviates the additional computational cost of higher-order inference, while maintaining the end-to-end learnability of the entire model.

Chapter 6

CONCLUSION

This thesis presented a series of neural structured prediction techniques that involved model spans of text. We showed that neural span representations are effective for a wide variety of natural language processing tasks, such as syntactic parsing (Chapter 2), extractive question answering (Chapter 3), and coreference resolution (Chapters 4 and 5). We also demonstrated a common pattern for enabling efficient inference by decomposing neural models into a components of varying complexity while being end-to-end learnable. For example, a local model can be used to guide A* inference through a small sliver of an enormous search space (Chapter 2). Unary mention scores can be used to prune the space of likely mentions for coreference resolution (Chapter 4). Similarly, efficient but less accurate pairwise antecedents scores can also be used to prune the space of likely antecedents in a coarse-to-fine approach (Chapter 5).

The empirical effectiveness of the proposed methods were shown for three core NLP tasks in English. While it remains to be seen how well these methods will generalize to other tasks and languages, they are in theory widely applicable. Though not discussed in this thesis, preliminary results show that minor modifications to the end-to-end coreference resolution model can also achieve promising results on other semantic span-based tasks, including semantic role labeling and named entity recognition, and on other languages, including Chinese and Arabic.

On a more speculative note, we expect ideas from this thesis to drive two important long term directions. First, while significant progress has been made on learning representations at the sentence level, document-level representations are still in its infancy. Our neural coreference resolution architecture provides the first step in using neural representations to predict document-level structures in an end-to-end manner, and we believe that further steps in this direction is vital for automatic natural language understanding with a broader notion of context. Second, it is unclear

whether fully supervised structured prediction tasks will remain relevant as the amount of available data continues to increase. With weaker supervision, we will need to generalize these approaches such that they can learn more complex latent structure. Our neural coreference system takes small steps towards this goal, by demonstrating that it can recover latent singleton mentions and latent antecedent trees with end-to-end learning. As we move further along the spectrum towards using more weakly supervised or unsupervised data, it will be important for our models to retain the most fundamental inductive bias: natural language is inherently structured.

BIBLIOGRAPHY

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-scale Machine Learning on Heterogeneous Systems. *Software available from tensorflow.org*, 2015.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2442–2452, 2016.

Y. Artzi and L.S. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62, 2013.

Michael Auli and Adam Lopez. Efficient CCG parsing: A* versus Adaptive Supertagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011.

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

Eric Bengtson and Dan Roth. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics, 2008.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, 2013.

Anders Björkelund and Jonas Kuhn. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *ACL*, 2014.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*, 2013.

Danqi Chen and Christopher D Manning. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 740–750, 2014.

Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of ACL*, 2016.

Kevin Clark and Christopher D. Manning. Entity-centric coreference resolution with model stacking. In *ACL*, 2015.

Kevin Clark and Christopher D. Manning. Improving coreference resolution by learning entity-level distributed representations. In *Association for Computational Linguistics (ACL)*, 2016a.

Kevin Clark and Christopher D. Manning. Deep reinforcement learning for mention-ranking coreference models. In *Empirical Methods on Natural Language Processing (EMNLP)*, 2016b.

Stephen Clark and James R Curran. Wide-coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4), 2007.

Stephen Clark, Darren Foong, Luana Bulat, and Wenduan Xu. The Java Version of the C&C Parser: Version 0.95. Technical report, University of Cambridge Computer Laboratory, August 2015.

Michael Collins. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002.

Michael Collins and Brian Roark. Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics, 2004.

James Cross and Liang Huang. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *EMNLP*, 2016.

Hal Daumé III and Daniel Marcu. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 97–104. Association for Computational Linguistics, 2005a.

Hal Daumé III and Daniel Marcu. Learning as search optimization: Approximate Large Margin Methods for Structured Prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176. ACM, 2005b.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

Pascal Denis and Jason Baldridge. Specialized models and ranking for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 660–669. Association for Computational Linguistics, 2008.

Li Dong and Mirella Lapata. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*, 2016.

Timothy Dozat and Christopher D Manning. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*, 2016.

Greg Durrett and Dan Klein. Easy victories and uphill battles in coreference resolution. In *EMNLP*, 2013.

Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. *TACL*, 2:477–490, 2014.

Greg Durrett and Dan Klein. Neural CRF Parsing. In *Proceedings of the Association for Computational Linguistics*, 2015.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based Dependency Parsing with Stack Long Short-Term Memory. In *Proc. ACL*, 2015.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*, pages 199–209, 2016.

Eraldo Rezende Fernandes, Cícero Nogueira Dos Santos, and Ruy Luiz Milidiú. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 41–48. Association for Computational Linguistics, 2012.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. *Proceedings of NIPS*, 2016a.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027, 2016b.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–11, 2016.

Aria Haghighi and Dan Klein. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of*

the Association for Computational Linguistics, pages 385–393. Association for Computational Linguistics, 2010.

Luheng He, Kenton Lee, Mike Lewis, and Zettlemoyer Luke. Deep semantic role labeling: What works and whats next. In *Association for Computational Linguistics (ACL)*, 2017.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of NIPS*, 2015.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of ICLR*, 2016.

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

Julia Hockenmaier and Mark Steedman. CCGbank: a Corpus of CCG derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3), 2007.

Liang Huang, Suphan Fayong, and Yang Guo. Structured Perceptron with Inexact Search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics, 2012.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*, 2017.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Chloé Kiddon, Luke S. Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *EMNLP*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *TACL*, 4:313–327, 2016.

Dan Klein and Christopher D Manning. A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, 2003.

Lingpeng Kong, Chris Dyer, and Noah A Smith. Segmental recurrent neural networks. *arXiv preprint arXiv:1511.06018*, 2015.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. Neural amr: Sequence-to-sequence models for parsing and generation. *arXiv preprint arXiv:1704.08381*, 2017.

Jayant Krishnamurthy, Oyvind Tafjord, and Aniruddha Kembhavi. Semantic parsing to probabilistic programs for situated question answering. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 160–170. The Associ-

ation for Computational Linguistics, 2016. URL http://aclweb.org/anthology/D/D16/D16-1016.pdf.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. Context-dependent semantic parsing for time expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-1135.

Kenton Lee, Mike Lewis, and Luke S. Zettlemoyer. Global neural ccg parsing with optimality guarantees. In *EMNLP*, 2016a.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*, 2016b.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

Mike Lewis and Mark Steedman. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

Mike Lewis, Luheng He, and Luke Zettlemoyer. Joint A* CCG Parsing and Semantic Role Labelling. In *Empirical Methods in Natural Language Processing*, 2015.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. LSTM CCG Parsing. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2016.

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *CoRR*, abs/1607.06275, 2016.

Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. Segmental recurrent neural networks for end-to-end speech recognition. *arXiv preprint arXiv:1603.00223*, 2016.

Sebastian Martschat and Michael Strube. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418, 2015.

Dipendra Misra and Yoav Artzi. Neural shift-reduce ccg semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1775–1786. Association for Computational Linguistics, 2016. URL `http://aclweb.org/anthology/D16-1183`.

Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, 2008.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of ICML*, 2010.

Vincent Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1396–1411. Association for Computational Linguistics, 2010.

Vincent Ng and Claire Cardie. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of EMNLP*, 2016.

Adam Pauls and Dan Klein. K-best A* Parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 958–966. Association for Computational Linguistics, 2009.

Hao Peng, Sam Thomson, and Noah A Smith. Deep multitask learning for semantic dependency parsing. *arXiv preprint arXiv:1704.06855*, 2017.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, 2014.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S1p31z-Ab`.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 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, 2012.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501. Association for Computational Linguistics, 2010.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100, 000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, 2016.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, 2013a.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of EMNLP*, 2013b.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 627–635, 2011. URL `http://www.jmlr.org/proceedings/papers/v15/ross11a/ross11a.pdf`.

Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Noah A. Smith. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, May 2011.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with Compositional Vector Grammars. In *Proceedings of the ACL conference*, 2013.

Mark Steedman. *Surface Structure and Interpretation*. The MIT Press, 1996.

Mark Steedman. *The Syntactic Process*. The MIT Press, 2000.

Mitchell Stern, Jacob Andreas, and Dan Klein. A minimal span-based neural constituency parser. In *ACL*, 2017.

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

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. Greedy, joint syntactic-semantic parsing with stack lstms. *arXiv preprint arXiv:1606.08954*, 2016.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved Semantic Representations from Tree-structured Long Short-term Memory Networks. *arXiv preprint arXiv:1503.00075*, 2015.

Wilson Taylor. Cloze procedure: A new tool for measuring readability. *Journalism Quarterly*, 30: 415–433, 1953.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of NAACL*, 2016.

Ashish Vaswani and Kenji Sagae. Efficient Structured Inference for Transition-Based Parsing with Neural Networks and Error States. *Transactions of the Association for Computational Linguistics*, 2016.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. Supertagging With LSTMs. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2016.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proceedings of NIPS*, 2015a.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781, 2015b.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a Foreign Language. In *Advances in Neural Information Processing Systems*, 2015c.

Ellen M Voorhees. Question answering in trec. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 535–537. ACM, 2001.

Ellen M. Voorhees and Dawn M. Tice. Building a question answering test collection. In *Proceedings of SIGIR*, 2000.

Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *Proceedings of ACL*, 2016.

Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016.

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. Learning anaphoricity and antecedent ranking features for coreference resolution. In *ACL*, 2015.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. Learning global features for coreference resolution. *arXiv preprint arXiv:1604.03035*, 2016.

Wenduan Xu. LSTM Shift-Reduce CCG Parsing . In *Empirical Methods in Natural Language Processing*, 2016.

Wenduan Xu, Michael Auli, and Stephen Clark. CCG Supertagging with a Recurrent Neural Network. *Volume 2: Short Papers*, page 250, 2015.

Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, 2015.

J.M. Zelle and R.J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055, 1996.

L.S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2005.

Luke Zettlemoyer and Michael Collins. Learning Context-dependent Mappings from Sentences to Logical Form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and*

the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2, 2009. ISBN 978-1-932432-46-6. URL http://dl.acm.org/citation.cfm?id=1690219.1690283.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Greed is Good if Randomized: New Inference for Dependency Parsing. In *Proceedings of EMNLP*, 2014.