Doctoral Dissertation

Studies on Efficient Parsing and Logic-based Inference based on Combinatory Categorial Grammar

Masashi Yoshikawa

February 19, 2020

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Masashi Yoshikawa

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Satoshi Nakamura	(Co-supervisor)
Associate Professor Masashi Shimbo	(Co-supervisor)
Assistant Professor Hiroyuki Shindo	(Co-supervisor)

Studies on Efficient Parsing and Logic-based Inference based on Combinatory Categorial Grammar*

Masashi Yoshikawa

Abstract

Combinatory Categorial Grammar (CCG) is a lexicalized grammatical formalism, which has been used in theoretical linguistics to provide explanations to various linguistic phenomena. It has also attracted attentions in the field of Natural Language Processing, for developping systems to solve natural language inference tasks by utilizing the linguistic insights. The goal of this thesis is to develop and extend inference systems that perform reasoning using logical formulas obtained from CCG trees. Specifically, our contributions are as follows: developing (1) an efficient and accurate CCG parsing method, (2) domain adaptation method of CCG parsing, and (3) knowledge insertion method for logic-based natural language inference system.

The first contribution is the development of an accurate and efficient CCG parser. Because pre-terminal categories (supertags) are highly informative in the lexicalized grammar, the vast majority of the decisions made during parsing involve assigning the correct syntactic role given by a supertag. Our method exploits these characteristics, and models the probability of a tree through the supertags, resolving the remaining ambiguities by syntactic dependencies. This strongly factored model allows the computation of the most probable CCG tree using an efficient A* parsing algorithm. We conduct various experiments, including evaluating the effect of the improved parsing accuracy on Recognizing Textual Entailment (RTE) tasks, by integrating our parser within logic-based reasoning systems. We observe that our parser leads to improved performance in English RTE experiments.

Second, we work on the domain adaptation issue of CCG parsing, since we are interested in applications of CCG-based inference systems in various domains such as scientific papers and speech conversation. We propose a new domain adaptation method,

*Doctoral Dissertation, Graduate School of Science and Technology, Nara Institute of Science and Technology, February 19, 2020.

based on the idea of automatic generation of CCG corpora using cheaper dependency-based linguistic resources. Our solution is simple and not relying on a specific parser architecture, making it applicable to the current best-performing parsers. We conduct parsing experiments on four different domains: biomedical texts, question sentences, speech conversation, and math problems, for the latter two of which we constructed the experimental datasets.

Our last contribution is on logic-based reasoning systems. We tackle the issue of adding external knowledge to these systems; there is a tension in extending knowledge base in these systems and their efficiency in solving a problem. We show that the processing speed of a state-of-the-art logic-based RTE system can be significantly improved by using techniques of Knowledge Base Completion. Additionally, we integrate this mechanism in a Coq plugin that provides a proof automation tactic for natural language inference. We show empirically that adding new knowledge data contributes to better RTE performance while not harming the processing speed in this framework.

Keywords:

Natural Language Processing, Combinatory Categorial Grammar, Syntax, Parsing, Domain Adaptation, Logical Inference

組み合わせ範疇文法に基づく効率の良い構文解析手法と論理推論に関する研究*

吉川 将司

内容梗概

組み合わせ範疇文法（Combinatory Categorial Grammar, 以降 CCG）は、理論言語学において、様々な言語現象に対する適切な説明を与えることが可能である一方で、理論の数学的な性質から、それらの理論的知見をプログラムとして実装可能であるため、自然言語処理の文脈においても注目されている。本論文の目的は、CCG に基づく構文木を論理式に変換し推論を行う自然言語推論システムの開発と拡張を行うことである。本論文の貢献は具体的に、(1) 高速かつ高精度な CCG 構文解析手法、(2) CCG 解析の分野適応手法、(3) 論理推論システムにおける自然言語推論のための公理生成技術の開発である。

最初の貢献は、高速かつ高精度な CCG 構文解析器の開発である。語彙化文法である CCG においては、前終端記号であるカテゴリ（あるいはスーパータグ）が文の構造に関して多くの情報を保持しており、CCG に基づく構文解析では、スーパータグにより決定する統語関係を解決することが大部分を占める。本研究における解析手法ではこの特徴を活かし、CCG 木の確率をスーパータグの確率に基づいてモデリングし、それだけでは決まらない曖昧性は CCG 木から取り出される係り受け構造によって解消する。これにより、木構造の確率を局所的な要素の積として表せ、高速な A*構文解析を用いて文に対する最尤な CCG 木を計算することが可能になる。実験では、日英語の CCG ツリーバンクにおける性能評価に加え、提案法の解析器を論理に基づく自然言語推論システムに統合することで、提案法の解析性能の含意関係認識タスクに対する影響も検証し、有用性を示した。

本研究では科学技術論文や対話を含む様々なドメインに対する CCG と論理推論に基づく自然言語処理システムの開発を目指すが、そのためにはこれらの分野のテキストに対して頑健に CCG 解析ができる必要がある。本論文の 2 つ目の研究では、CCG 解析のための新たな分野適応技術を提案する。技術の要となるの

*奈良先端科学技術大学院大学 先端科学技術研究科情報科学領域 博士論文, 2020 年 2 月 19 日.

は、比較的安価な係り受け構造に基づく言語リソースを用いたCCGツリーバンクの自動生成である。提案法は、分野適応を行う解析器の構成に依らないものであり、近年の高性能な解析器にも適用可能である。実験では、生物医学論文、疑問文、対話テキスト、数学問題の４つの分野において提案法の有用性を検証した。中でも対話テキスト、数学問題において解析性能の著しい向上が見られた。

本研究最後の貢献は、論理推論システムに対するものである。論理推論システムに対して、知識ベースなどから得られる外部知識を導入する場合、知識データを増やすにつれ問題を解く速度が低下してしまうという緊張関係が存在する。そこで、本研究では知識グラフ補完の技術を応用することによりこの緊張関係を緩和し、最高精度の論理に基づく含意関係認識システムの処理速度を大幅に改善可能であることを示す。さらに、本研究ではその仕組を定理証明支援系のCoqのプラグインとして実装することにより、自然言語推論のための知識推論を含めた自動推論機構を構築する。実験において、提案法では処理速度を損なわずに知識データを追加することで含意関係認識の精度を向上させることができた。

キーワード

自然言語処理、組み合わせ範疇文法, 統語論, 構文解析, 意味解析, 分野適応, 論理推論

Acknowledgments

主指導教官の松本裕治教授には、大学院の日々を通して終始暖かくご指導していただき、研究内容に関する様々な助言をいただきました。なによりもまず、情報科学の入門者だった自分に、自然言語処理に挑戦する機会を与えていただいたことについて感謝しきれません。それに加え、外部の研究室での長期滞在など、自由な研究生活を許して頂いたことも深く感謝致します。

中村哲教授には、お忙しい中審査委員をお引き受けいただき、公聴会などで研究に関する助言をいただきました。有難うございました。

新保仁准教授、進藤裕之助教には、お忙しい中審査委員をお引き受けいただき、研究に関する様々な助言をいただいたきました。また、勉強会や個別でお話したときの議論はいつも興趣が尽きず、非常に有意義な学生生活となりました。有難うございました。

学部時代に指導を受けました大阪大学外国語学部の藤家洋昭准教授には、計算言語学や自然言語処理といった形式的に言語を扱う分野の存在を教えていただきました。先生との出会いがなければ、学問の世界には進めていなかったと思い、感謝しきれません。また当時、先生のもとに長居し、言語学について色々お話を聞かせていただいたことにも感謝しています。

松本研究室の方々や秘書の北川祐子さんには、研究面や生活面で助けていただきました。北川さんには東京奈良間の書類手続きのやり取り等でかなりお手数をおかけしました。有難うございました。

産業総合研究所の能地宏先生には、私の研究を通して終始指導していただき大変お世話になりました。研究上の鋭い指摘はもちろんのこと、読みやすい論文の書き方やプレゼンテーションの心がけまで、自身の研究者としての成長において大きな影響を与えていただきました。有難うございました。

また、博士後期課程の多くの時間をお世話になりましたお茶の水女子大学の戸次大介准教授、峯島宏次特任准教授にも多くの影響を受けました。まず、２年半という長期滞在を許していただき有難うございました。論文紹介を通した議論がいつも楽しく、また普段の何気ない雑談における言語に対する深い造詣にいつも感動していました。大規模の研究プロジェクトや学校での課題などで多くの活躍

の場を与えていただいたことにも感謝しています。学生でありながら、少し進んで研究者としての経験を多く積むことができました。

お茶の水女子大学戸次研究室の学生の方々にも大変お世話になりました。皆様とゼミ、勉強会で活発な議論を通して勉強した時間はとても有意義でした。また不定期開催の勉強会もとても楽しかったです。研究に関しては、馬目華奈さん、鈴木莉子さんと一緒に研究をさせていただいたことは学ぶところの多い経験となりました。また、研究室で知り合った東京大学の渡邉知樹くんにも、大いに感謝しています。数学について教えてもらったのみならず、難しい本をグループで読みすすめることの生産性の高さなど学問一般の取り組み方について多く学ばせていただきました。博士後期課程は、奈良の山で修行僧のように過ごすのだろうと覚悟していましたが、思いもよらず、皆様のおかげで楽しく、精神的にもとても過ごしやすい時期となりました。大いに感謝しています。

最後に、これまで私を支えてくれた両親と二人の弟に感謝致します。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As a theory of human language, a *syntactic theory* should provide explanations to a range of linguistic phenomena of natural languages, those arising from the way words of a sentence are concatenated:

1. *John* [[*saw* [*a girl*]] [*with a telescope*]]]
2. *Mary* [[*cooked*] *and* [*might eat*] *the beans*]

The first example above is a case of the widely known *PP-attachment*, where a prepositional phrase *with a telescope* can compose a larger phrase by attaching to either *saw* or *a girl*, resulting in different readings of the sentence. The second example is of *coordination* construction, which also has the similar sort of ambiguity. Examples of such phenomena are numerous to mention; not only *wh*-constructions, control verb, complement, active/passive voices, scope, etc., which are observable cross-linguistically, but also those specific to some languages, such as so-called "eel sentences",[1] double subject construction,[2] and indirect passive voice,[3] peculiar to some languages such as Japanese.

[1] The construction in Japanese sentence 僕はうなぎだ。(*Boku-wa unagi da*) is basically used to express copula "X is Y" , according to which the sentence is understood as meaning "I am an eel". The same sentence is also used to express that the speaker intends he or she will have an eel for dinner, where the particle *wa* is used as topic marker.

[2] ジョンがメアリーが嫌い。(*John-ga Mary-ga kirai*), "It is in the case of John that Mary is disliked." The primary usage of the particle *ga* is marking a subject.

[3] 太郎が妻に倒れられた。(*Taroo-ga tuma-ni taore-rare-ta.*), "Taro was in trouble because his wife got sick in bed."

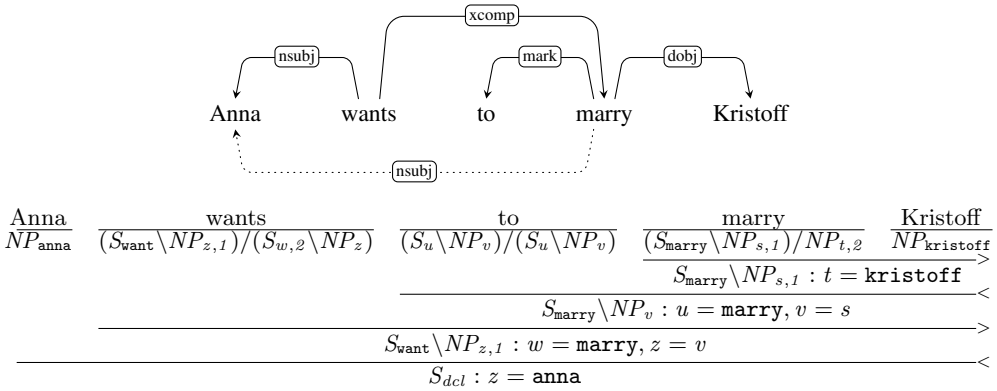


Figure 1.1: Syntactic trees for sentence *Anna wants to marry Kristoff* based on UD (above, borrowed from Reddy et al. (2017)) and CCG (below). A QA system based on UD tree (Reddy et al., 2017) has an issue in handling control verb, due to the lack of information marking *Anna* as the subject of *marry*. Using CCG, such information is available by reading the chain of variables $s = v = z =$ `anna`, constructed according to the CCG tree. See Section 2.4 for the mechanism of predicting such a relation.

Combinatory Categorial Grammar (CCG; Steedman (2000, 2012)) is a grammatical formalism, that has its root in mathematical logic (Ajdukiewicz, 1997), and one of its central characteristics is its formally defined transparent interface between syntax and semantics. The syntactic theory has been long used as an analytical tool for theoretical linguists to explain a wide range of linguistic phenomena, while its mathematical formality and the special care taken on the computational complexity through the theory's development facilitate Natural Language Processing (NLP) researchers to implement the theory, along with theoretical linguists' insights, as a syntactic/semantic parsing component of their NLP systems.

In the literature, there have been several applications of CCG in NLP researches such as induction of meaning representations of a sentence (Artzi et al. (2015); Artzi and Zettlemoyer (2013); Reddy et al. (2014), among others). However, it is fair to say that dependency-based syntactic formalisms have attracted considerable attention of the community, and their effort has been devoted to developing linguistic resources and parsing algorithms for dependencies, whose most prominent case being the Universal Dependencies (UD) project (Nivre et al., 2016), which is an ongoing program to assign the consistent annotation of syntactic dependencies across different human languages. We would like begin our thesis by arguing the attractiveness of CCG, sometimes by

comparing it with the UD syntax, and give the motivation of this entire thesis.

Firstly, CCG provides a unified platform for treating various linguistic constructions. Previous work by Reddy et al. (2017) for developing a Question Answering (QA) system uses a query language converted from the UD-based dependency structure of an input sentence. Reportedly, they had a difficulty in handling some constructions, due to the lack of some structural information in the UD trees. For example, Figure 1.1 above (borrowed from Reddy et al. (2017)) is a UD-based syntactic representation for sentence *Marry wants to marry Kristoff*, where it does not make explicit the subject-verb relation between the subject *Anna* in the matrix clause and the infinitive verb *to marry* (where *want* plays a role of so-called *control verb*). According to the UD project, there will be an *enhanced* version of the UD, where this sort of relations are made explicit in the annotation. However, we suspect this will in turn become problematic in terms of the parsing algorithm, since existing dependency parsing methods exploit the simplicity of dependency structure (*one word one head*). For CCG, capturing *long-range dependencies* spanning multiple words like this example was the earliest interest of the theory's development, and it provides apparatus that effectively handles the above example (see Figure 1.1 below), as well as other longer dependencies such as relative clauses and *wh*-constructions.

As noted, the fact that the CCG grammar is formally defined makes it possible for NLP researchers to be benefited from the insights of the theoretical linguistics, and to address complex linguistic phenomena, those reside rather in the "long tail" of the distribution, and as such, are difficult to address using purely data-driven machineries. Haruta et al. (2019) is one of such cases, where they address inference problems involving comparative expressions (e.g., *John runs as fast as Michael* entails *Michael runs fast*). By extending the meaning representation of a CCG-based inference system by Mineshima et al. (2015), they achieve high scores on sections related to the comparative expressions of the FraCaS dataset, a collection of linguistically challenging RTE problems (Cooper et al., 1994). A work by Matsuzaki et al. (2017) is another outstanding example, where they extend the Japanese CCG grammar by Bekki (2010) with analyses on sentences containing math expressions, and develop a CCG-based semantic parser for pre-university math problems based on it. CCG-based inference systems can be used to address highly domain-specific constructions as well.

Though the focus of the earliest work of the CCG literature has been wholly on Indo-European languages such as English and Dutch, the seminal work by Bekki (2010) provides detailed and comprehensive analyses for syntactic/semantic phenomena of

the Japanese language, including those listed at the beginning of this chapter, i.e., eel sentences, double subject construction, and indirect/direct passive voice, and those involve complex clauses: sentential noun modification, conditionals, and quotes. The work is done by combining CCG and the latest semantic representation language at the time. This empirically proves the theory's strength in explaining various linguistic phenomena of various human languages. On the other hand, the Parallel Meaning Bank project (Abzianidze et al., 2017) has developed CCG-based treebanks for English, German, Dutch and Italian languages, along with other semantic annotation layers, such as meaning representations based on Discourse Representation Theory (Kamp and Reyle, 1993) and universal semantic tags (Bjerva et al., 2016). The project is still expanding, and other languages such as Japanese will be included in future. CCG can be used to universally annotate diverse human languages, while bringing those merits discussed in this section. CCG and dependency-based syntactic representations may play complementary roles in NLP, according to the granularity of information required for the task to be solved.

To summarize, the CCG-based approach has potential to develop inference systems that, collaborating with theoretical linguistics, address natural language problems which cannot be solved using purely data-driven methods. However, these advantages are off course not for free; predicting more informative syntactic structure naturally incurs solving more complex optimization problem. On top of that, there is an issue in terms of linguistic resources. Annotating a CCG tree to a sentence is costly; the linguistic expertise is required, and the grammar's strict nature does not allow annotation errors. In this thesis, by solving these problems, we aim to bring the advantages of using formal grammar-based NLP systems closer to real applications.

1.2 Contribution

Based on the above arguments, this thesis aims at developing and extending NLP systems that utilize syntactic structure provided by CCG. The contributions of this thesis are summarized as follows.

First, we work on developing an accurate and efficient CCG-based syntactic parser, which is indispensable for developing an accurate inference system based on the grammar. Since CCG is a strongly lexicalized grammatical formalism (Section 2.1), in which pre-terminal categories (*supertags*) are highly informative as to the higher-level structure, the vast majority of the decisions made during parsing involve assigning the

correct syntactic role given by a supertag. In Chapter 3, we propose a new A* parsing model for CCG that exploits this characteristics, by modeling the probability of a tree through the supertags and resolving the remaining ambiguities by its syntactic dependencies. The key of our method is that it predicts the probabilities of supertags and dependency heads independently with a strong unigram model defined over bidirectional LSTMs (Schuster and Paliwal, 1997). The factorization allows the precomputation of the probabilities for all possible trees for a sentence, which enables very efficient decoding combined with A* parsing (Klein and D. Manning, 2003). We conduct various experiments to verify the proposed method; we develop parsers for English and Japanese languages, and evaluate them on the respective CCGbanks (Hockenmaier and Steedman, 2007; Uematsu et al., 2015). Our model achieves the state-of-the-art results on these settings. Additionally, we conduct Recognizing Textual Entailment (RTE) experiments by integrating our parser within logic-based RTE systems (Section 2.3). We observe that our parser leads to improved performance in English RTE experiments, while, in the Japanese RTE experiment, we find out a drawback of our method in handling unary rules, which is then inspected with a detailed analysis. We name the developed parser in this chapter `depccg`, after the fact that the use of dependencies is the key of the method. The `depccg` parser is used in various experiments throughout this thesis.

To use CCG-based inference systems in real applications, such as inference systems on scientific papers and speech conversation, the parsing accuracy on texts from these domains is critical, since the succeeding components heavily rely on its output. Our next focus in Chapter 4 is the domain adaptation issue of CCG parsers. We propose a new domain adaptation method for CCG parsing, based on the idea of automatic generation of CCG corpora exploiting cheaper resources of dependency trees. Our solution is conceptually simple, and not relying on a specific parser architecture, making it applicable to the current best-performing parsers. We conduct extensive parsing experiments with detailed discussion; on top of existing benchmark datasets on (1) biomedical texts and (2) question sentences, we create experimental datasets of (3) speech conversation (on the Switchboard corpus (Godfrey et al., 1992)) and (4) math problems, with applications of a CCG-based inference system on these domains. When applied to the proposed method, our `depccg` parser developed in Chapter 3 shows significant performance gains, improving from 90.7% to 96.6% on speech conversation, and from 88.5% to 96.8% on math problems.

Our last contribution is on inference systems based on CCG and logic. In logic-

based approaches to reasoning tasks such as RTE and QA, it is important for a system to have a large amount of knowledge data. However, there is a tradeoff between adding more knowledge data for improved RTE performance and maintaining an efficient RTE system, as such a big database is problematic in terms of the memory usage and computational complexity. In Chapter 5, we show the processing time of a state-of-the-art logic-based RTE system can be significantly reduced by replacing its search-based axiom injection (abduction) mechanism by that based on Knowledge Base Completion (e.g., Bordes et al. (2013); Trouillon et al. (2016); Dettmers et al. (2017)). We integrate this mechanism in a Coq plugin (The Coq Development Team, 2017) that provides a proof automation tactic for natural language inference. We show empirically that adding new knowledge data contributes to better RTE performance while not harming the processing speed in this framework.

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2: Preliminaries We first provide basic knowledge about CCG. Then, we introduce how the grammar is used to derive the semantic structures of a sentence, and how the structures are used in CCG parsing evaluation and existing logic-based systems solving RTE tasks.

Chapter 3: A* CCG Parsing In this chapter, we propose a probabilistic model of a CCG tree and an efficient parsing algorithm based on the A* algorithm. We evaluate our method in terms of the wide range of settings; we develop parsers for English and Japanese languages, each evaluated on the respective CCGbanks. We also evaluate the parsers in terms of the performance gains they contribute to existing logic-based RTE systems, which heavily rely a CCG parser to derive their meaning representations.

Chapter 4: Domain Adaptation In this chapter, we focus on the domain adaptation issue of a CCG parser. We propose a domain adaptation method for CCG parsing, which exploits cheaper and abundant dependency-based treebanks to obtain the corresponding CCG treebanks (CCGbanks). We evaluate the method by performing parsing experiments on new domains including biomedical texts, question sentences, speech

conversation and math problems, for the two latter of which we constructed the experimental datasets.

Chapter 5: Axiom Injection In this chapter, we turn our focus to an issue of logic-based RTE systems. We propose to leverage the techniques of Knowledge Base Completion to enable efficient insertion of lexical knowledge during solving an RTE problem. We evaluate the effectiveness of the method in terms of the RTE.

Chapter 6: Conclusion This chapter summarizes the thesis and discusses the direction of further research.

1.4 Programs and Resources

The programs and resources developed in our work in publicly available from the following links:

- Chapter 3: `depccg` parser
  - https://github.com/masashi-y/depccg
- Chapter 4: CCG treebanks on speech conversation (the Switchboard corpus (Godfrey et al., 1992)) and math problems
  - https://github.com/masashi-y/ud2ccg
- Chapter 5: Coq plugin for knowledge insertion
  - https://github.com/masashi-y/abduction_kbc

Chapter 2

Preliminaries

First, we introduce CCG and describe English and Japanese grammars adopted in this work.

2.1 Combinatory Categorial Grammar

CCG is characterized by categories with rich internal structures, and a limited number of derivational rules, called *combinatory rules*. A CCG category is either atomic (S, N, NP, etc.) or functional (complex), where a complex category of the form A/B (or $B\backslash A$) is equivalent to a functional type $B \rightarrow A$, representing a function that returns a result of type A when applied to an argument of type B. A forward slash $/$ (or backward $\backslash$) indicates that the argument of type B must appear to the right (or left) of that functor. A category can be marked with *feature values*, such as `adj` in $S_{\texttt{adj}}$, which are rather language-specific, fine-grained controls over which pairs of categories can be combined (See the documentation of respective CCGbanks for detailed specifications).

Combinatory rules are meta rules, in that they contain variables such as X and Y, that matches any category (cf. rules such as $S \Rightarrow NP\ \ VP$ in a context-free grammar does not contain any). Table 2.1 shows a set of combinatory rules. The forward application, symbolically denoted as $>$, is a basic combinatory rule that combines categories X/Y and Y (occurring in this order) into X. Note that, in Table 2.1, a λ-term is assigned to each category, as in "$X/Y\ :\ \lambda x.f(x)$". One of the notable features of CCG is that it provides a transparent interface to semantic theories, i.e., a meaning representation of a sentence is constructed concurrently with a derivation. This is the fundamental principle that most of the logic-based RTE systems, some of which

$$\frac{X/Y : \lambda x.f(x) \quad Y : a}{X : f(a)}{>} \qquad \frac{Y : a \quad X\backslash Y : \lambda x.f(x)}{X : f(a)}{<}$$

$$\frac{X/Y : \lambda x.f(x) \quad Y/Z : \lambda x.g(x)}{X/Z : \lambda x.f(g(x))}{>\mathbf{B}^1} \qquad \frac{Y\backslash Z : \lambda x.g(x) \quad X\backslash Y : \lambda x.f(x)}{X\backslash Z : \lambda x.f(g(x))}{<\mathbf{B}^1}$$

$$\frac{X/Y : \lambda x.f(x) \quad Y\backslash Z : \lambda x.g(x)}{X\backslash Z : \lambda x.f(g(x))}{>\mathbf{B}^1_\times} \qquad \frac{Y/Z : \lambda x.g(x) \quad X\backslash Y : \lambda x.f(x)}{X/Z : \lambda x.f(g(x))}{<\mathbf{B}^1_\times}$$

$$\frac{X/Y : \lambda x.f(x) \quad Y/Z_1|\ldots|Z_n : \lambda z_n \ldots z_1.g(z_1,\ldots,z_n)}{X/Z_1|\ldots|Z_n : \lambda z_n \ldots z_1.f(g(z_1,\ldots,z_n))}{>\mathbf{B}^n}$$

$$\frac{Y\backslash Z_1|\ldots|Z_n : \lambda z_n \ldots z_1.g(z_1,\ldots,z_n) \quad X\backslash Y : \lambda x.f(x)}{X\backslash Z_1|\ldots|Z_n : \lambda z_n \ldots z_1.f(g(z_1,\ldots,z_n))}{<\mathbf{B}^n}$$

$$\frac{X/Y : \lambda x.f(x) \quad Y\backslash Z_1|\ldots|Z_n : \lambda z_n \ldots z_1.g(z_1,\ldots,z_n)}{X/Z_1|\ldots|Z_n : \lambda z_n \ldots z_1.f(g(z_1,\ldots,z_n))}{>\mathbf{B}^n_\times}$$

$$\frac{Y/Z_1|\ldots|Z_n : \lambda z_n \ldots z_1.g(z_1,\ldots,z_n) \quad X\backslash Y : \lambda x.f(x)}{X\backslash Z_1|\ldots|Z_n : \lambda z_n \ldots z_1.f(g(z_1,\ldots,z_n))}{<\mathbf{B}^n_\times}$$

Table 2.1: Set of common combinatory rules. Each English and Japanese parser developed in this work uses a subset of these rules. A vertical | bar can match both forward and backward slashes. All /, \ and | are left-associative. See the text for details.

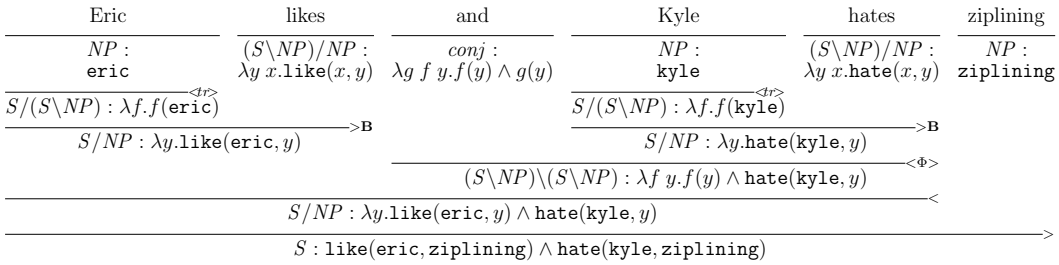


Figure 2.1: Example CCG derivation for a complex coordination structure, where type raising $<tr>$ and forward composition $>\mathbf{B}^1$ play important roles.

we evaluate in the RTE experiments, are based on (Section 3.7). Figure 1.1 shows an example CCG derivation using forward (and similarly defined backward $<$) application. Conventionally, a CCG derivation is drawn top-down, resembling a proof diagram in natural deduction (Prawitz, 1965). Since we are particularly interested in the construction algorithm of derivations, λ-terms are omitted when they are not relevant. Figure 2.1 shows a derivation (using combinatory rules explained below), where a meaning representation is constructed according to the derivation.

Forward and backward composition rules ($>\mathbf{B}^1$ and $<\mathbf{B}^1$) and their crossed variants ($>\mathbf{B}^1_\times$ and $<\mathbf{B}^1_\times$) compose a pair of complex categories; you can see from the operation performed on the λ-terms in Table 2.1 that they literally correspond to function composition $f \circ g$. They can be generalized to $n \geq 2$ (in the last four rows in the table), where a vertical bar $|$ means that it can be either of $/$ or $\backslash$. Many elegant CCG analyses for complex syntactic phenomena employ composition rules; Figure 2.1 is one of such cases, where forward composition married with a unary *type raising rule* ($X \Rightarrow Y\backslash(Y/X)$, denoted as $<tr>$) explains a complex coordination structure. Composition rules are also employed to capture phenomena that involve long-range dependencies, such as *wh*-extraction.

While the strong expressivity of CCG plays various crucial roles in the analysis of complex constructions, it also comes at a cost, i.e., it introduces *spurious ambiguity*. For example, "*Kyle hates*" is licensed as a constituent (Figure 2.1); thus, it can be combined with "*ziplining*" to derive exactly the same interpretation as a derivation that combines "*hates ziplining*" first. To avoid combinatorial explosion during parsing, a technique called *normal form parsing* (Eisner, 1996a; Hockenmaier and Bisk, 2010), which imposes constraints such that only "normal-form" derivations are produced, has been developed. One of the findings of Chapter 3 is that the proposed dependency-

augmented model can automatically learn softened versions of such constraints. Experimental results demonstrate that, compared to baseline methods, even though the constraints are not given explicitly, the proposed parser violates the constraints significantly fewer times (Section 3.6).

2.1.1 CCG Grammar for English Language

Following Lewis and Steedman (2014a), our English parser uses the standard binary rules from Steedman (2012): *forward and backward application* ($>$ and $<$), *(generalized) forward composition* ($>\mathbf{B}^n$ for $n = 1, 2$) and *(generalized) backward crossed composition* ($<\mathbf{B}^n_\times$ for $n = 1, 2$). In addition, the parser uses five binary rules that include a coordination-specific *conjunction* rule ($<\Phi>$) and exactly the same set of unary rules in (Lewis and Steedman, 2014a), including type raising. The complete set of combinatory rules is summarized in Appendix A.

2.1.2 CCG Grammar for Japanese Language

In Japanese CCG parsing, we use the set of binary rules found in the annotation of Japanese CCGbank, which basically follows Bekki (2010): *forward and backward application* ($>$, $<$), *forward and backward composition* ($>\mathbf{B}^1$, $<\mathbf{B}^1$), *generalized backward composition* ($<\mathbf{B}^n$ for $n = 2, 3, 4$), *generalized forward crossed composition* ($>\mathbf{B}^n_\times$ for $n = 2, 3$), and the "*SSEQ*" rule. *SSEQ* is rather specific to the corpus, where multiple sentences can occur within a *sentence* when they are quoted:

> 「 [$_S$ [$_S$ 突然家が崩れ、着のみ着のまま飛びだした。] [$_S$ 生きているのが不思議]] 」と話した。(translation: "The house collapsed all of a sudden and I ran out with nothing but the clothes on my back. I cannot believe that I am still alive", he said.)

In this example, the SSEQ rule performs $S\ S \Rightarrow S$. For the unary rules, we select rules based on frequencies in the CCGbank. Most rules are related to sentential noun modifiers that turn a sentence-like category (S and $S\backslash NP$) into a nominal modifier (N/N). A complete list of the rules used is provided in Appendix B.

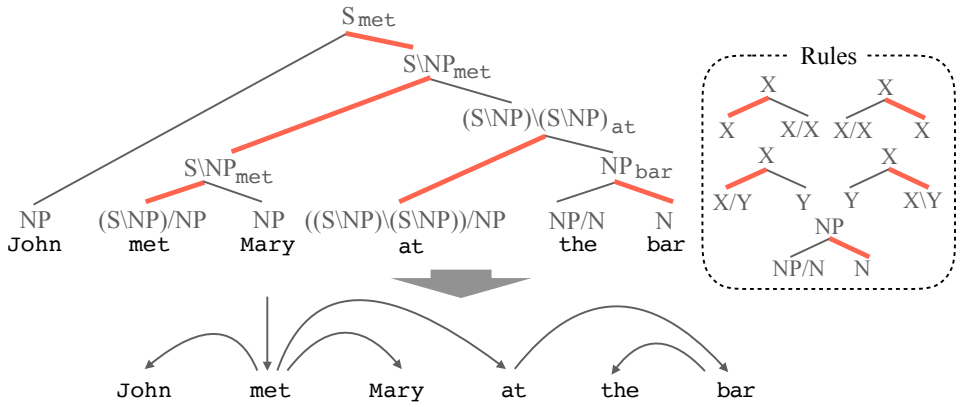


Figure 2.2: Example dependency extarction algorithm. Extraction rules are shown at the righthand side, where, given a subtree with two children, the child with the red edge is specified as the head of the other. In the smallest subtree containing *at the bar*, subtree with *at* is the head of *the bar* phrase (whose head is marked at the category as *bar*), recovering *at* $\rightarrow$ *bar* $\rightarrow$ *the* relations.

2.2 CCG and Dependency Grammar

Since we will be using *dependency trees* (interchangeably called *dependency structures* or *dependencies*) in some of the following chapters, we briefly describe the grammar here.

Formally, a dependency tree for sentence $\boldsymbol{x}$ of length N is a pair of $\boldsymbol{d} = (d_1, ..., d_N)$ and $\boldsymbol{\ell} = (\ell_1, ..., \ell_N)$, where $d_i \in \{0, ..., N\}$ is an index of its dependency parent of i'th word (0 is the index for the *root token*), and ℓ_i is the label of the corresponding dependency edge (e.g., *nsubj*). Sometimes labels $\boldsymbol{\ell}$ is not specified, giving *unlabeled* dependency structure. Example dependency trees are shown in Figure 1.1 and Figure 2.2 (the latter is unlabeled).

Under dependency-based formalisms, there are some representational variants (or grammars), differing on how they express particular syntactic constructions. For example, some grammars such as the Stanford dependencies (de Marneffe et al., 2006) regard prepositions as gluing its argument to the predicate (e.g., *met* $\rightarrow$ *at* $\rightarrow$ *bar*), meanwhile in others like UD, the argument is directly linked to the verb, reminiscent of predicate argument structure. Such a syntactic representation of a sentence can also be extracted from constituency-based syntactic representations such as CCG. We can construct an extraction algorithm by defining, for all possible (binary) nodes, which of left

or right children becomes the parent of the other. The Figure 2.2 shows the algorithm used in (Lewis and Steedman, 2014a). They define a subtree with a functor category as the head of the other, with some exceptions such as determiners and modifiers (note modifiers are characterized as having the X/X patterns), which are naturally attached to their modifiees. Conversely, it is generally not possible to recover a constituency-like syntactic representation from a dependency tree, given the information-losing nature of the above algorithm. However, we will show modeling the dependency structure of a CCG tree is effective to build an accurate CCG parser (Chapter 3).

2.3 Existing CCG Parsers

Here we summarize existing CCG parsers that will be used in experiments throughout this thesis. One of the main contributions of this thesis is to add our own parser to the list (see Chapter 3), which we call `depccg`, based on the fact that the key technique of the parser is the introduction of *dependency* terms into the probabilistic model.

- `C&C` (Clark and Curran, 2007)[1] is a well-known CCG parser. It is based on a log-linear model defined over an entire parse tree and finds an optimal tree using the CKY algorithm. The supertagger of this parser, which is also defined as a log-linear model, first assigns the most probable supertags to the input sentence to help reduce the search space in the succeeding CKY stage.

- `EasyCCG` (Lewis et al., 2016)[2] implements the A* parser introduced in Section 3.2, which our work in Chapter 3 is primarily based on. `EasyCCG` adopts a supertag-factored model, where supertag probabilities are computed using a feedforward neural network with fixed window context, and parsing ambiguities are resolved heuristically prioritizing trees involving longer dependencies (Section 3.2.1).

- `EasySRL` (Lewis et al., 2016)[3] extends `EasyCCG` by replacing its supertagger with one based on bidirectional LSTMs (Schuster and Paliwal, 1997). This parser resolves parsing ambiguities using attach-low heuristics described in Section 3.2.1.

[1] https://www.cl.cam.ac.uk/~sc609/candc-1.00.html
[2] https://github.com/mikelewis0/easyccg
[3] https://github.com/uwnlp/EasySRL

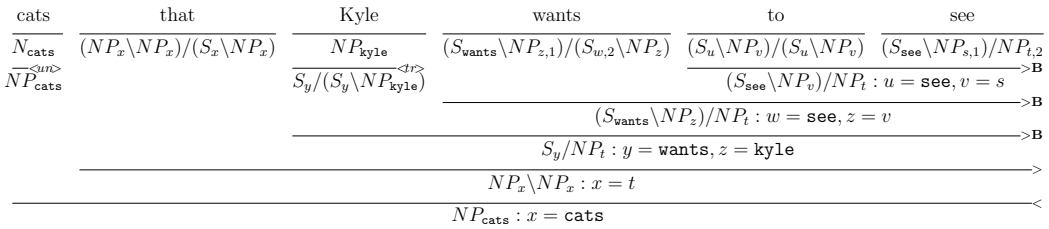


Figure 2.3: Example CCG derivation tree for phrase *cats that Kyle wants to see*.

- `neuralccg` (Lee et al., 2016)[4] extends `EasySRL` with scores from a TreeLSTM (Tai et al., 2015). Using globally conditioned score, `neuralccg` achieves the current state-of-the-art results in the CCGBank parsing. `neuralccg` also uses A* parsing to obtain an optimal parse tree; however, calculating the scores incurs significant computational costs thereby reducing processing speed (Section 3.6.3).

- `Jigg` (Noji and Miyao, 2016)[5] is a shift-reduce CCG parser for Japanese language, implemented in an NLP tool Jigg (Noji and Miyao, 2016) (for simplicity, we refer to the CCG parser as `Jigg`). `Jigg` is based on a linear model to predict an action for each time step, and uses beam search to reduce possible parse errors.

2.4 CCG Parsing Evaluation

The semantic structure of a sentence can be extracted using the functional nature of CCG categories. Figure 2.3 shows an example CCG derivation of a phrase *cats that Kyle wants to see*, where categories are marked with variables and constants (e.g., `kyle` in $NP_{\texttt{kyle}}$), and argument ids in the case of verbs (subscripts in $(S_{\texttt{see}}\backslash NP_{s,1})/NP_{t,2}$). Unification is performed on these variables and constants in the course of derivation, resulting in chains of equations $s = v = z = \texttt{kyle}$, and $t = x = \texttt{cats}$, successfully recovering the first and second argument of *see*: *Kyle* and *cats* (i.e., capturing *long-range dependencies*). What is demonstrated here is performed in the standard evaluation of CCG parsing, where the number of such correctly predicted predicate-argument relations is calculated. In the parsing experiments, labeled and unlabeled F1

[4] https://github.com/uwnlp/neuralccg
[5] https://github.com/mynlp/jigg

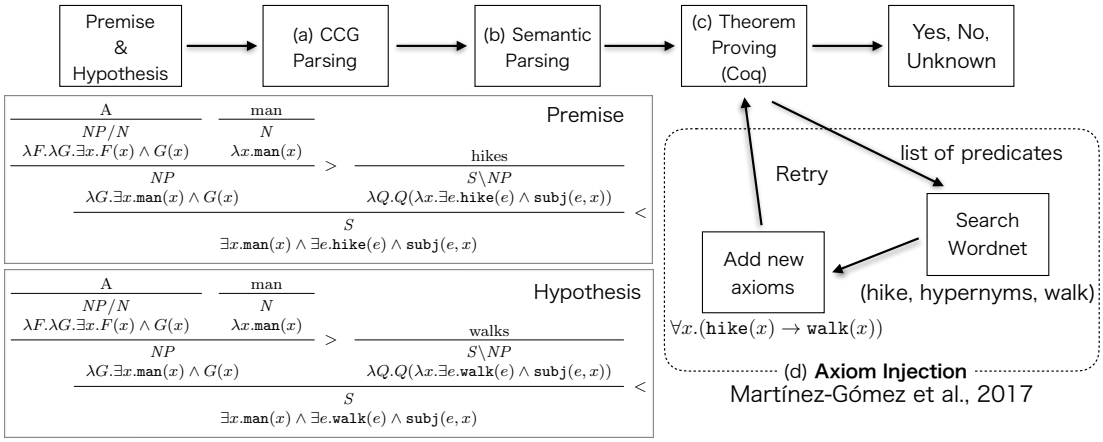


Figure 2.4: Pipeline of `ccg2lambda`. First, it applies CCG parser to premise (P) and hypothesis (H) sentences (a), and then converts them to logical formulas (b). It attempts to prove if entailment (contradiction) can be established by applying Coq to the theorem $P \rightarrow H$ ($P \rightarrow \neg H$) (c). If the theorem proving fails, it tries axiom injection (d).

scores of the extracted CCG semantic dependencies are reported, where the "labeled" metric takes into account the head category with which a predicate-argument relation is constructed, while the unlabeled one only considers the relations alone (for the detail, see (Clark et al., 2002)). Remarkably, it is also the basis of CCG-based semantic parsing (Abzianidze, 2017; Martínez-Gómez et al., 2017; Matsuzaki et al., 2017). In these systems, the above simple unification rule is replaced with more sophisticated techniques such as λ-calculus, which are described in the next subsection.

2.5 CCG-based Natural Language Inference Systems

We introduce logic-based systems for solving the Recognizing Textual Entailment (RTE) task. RTE is a challenging NLP task where the objective is to judge whether a hypothesis H logically follows from premise(s) P. This task is fundamental, in that advances in this task have positive implications in other areas such as information retrieval, question answering and reading comprehension. In Section 3, we evaluate our developed parser in terms of the performance gains of these systems, and in Section 5, we extend one of them using the proposed method.

2.5.1 ccg2lambda

Mineshima et al. (2015) develops a higher-order logic-based RTE system called `ccg2lambda`. As shown in Figure 2.4, the `ccg2lambda` pipeline first processes premise(s) and a hypothesis using a CCG parser. The parse trees are converted into logical formulas by composing λ-terms assigned to each terminal word in accordance with combinatory rules. For the assignment of λ-terms, `ccg2lambda` adopts a template-based procedure, where closed-class words (logical or functional expressions) are mapped to their specific meaning representations and other words to schematic meaning representations based on CCG categories. In this work, we adopt a semantic template based on Neo-Davidsonian Event Semantics (Parsons, 1990), where a sentence is mapped to a formula involving quantification over events and a verb is analyzed as a one-place predicate over events using auxiliary predicates for semantic roles such as `subj`. The system uses an automated theorem proving in Coq (The Coq Development Team, 2017) to determine whether entailment or contradiction holds between the premise(s) and the hypothesis. The RTE system implements a specialized prover for higher-order features in natural language, which is combined with Coq's built-in efficient first-order inference mechanism.

2.5.2 LangPro

`LangPro` (Abzianidze, 2017) is an analytic tableau system based on natural logic (van Benthem, 2008). Natural logic seeks a formal logic whose formulas are as close as possible to linguistic expressions. `LangPro` develops an efficient RTE system based on natural logic by extending a previously proposed analytic tableau system (Muskens, 2010), where higher-order logic based on a simple type theory is used as natural logic and a version of an analytic tableau method is designed for it.

`LangPro` also constructs a meaning representation based on Lambda Logical Form (LLF; Muskens (2010)) for a sentence in the accordance with CCG categories and combinatory rules. `LangPro` constructs an intermediate representation by first mapping each CCG category in a tree to a function type of λ-terms (e.g., mapping both A/B and $B\backslash A$ to $\langle \mathtt{B}, \mathtt{A} \rangle$, an alternative notation of $B \to A$, which is common in formal semantics). Then, its subtrees are rotated according to the order of the functional application of those types. When this conversion is applied to a sentence "*There is no one*

cutting a tomato", the result is as in Eq. 2.1.[6] This process also involves procedures that normalize the CCG tree (e.g. mapping "*no one*" to "*no person*"). Theorem proving operates on LLFs obtained from this representation by mapping quantified noun phrases, such as "*no person*" and "*a tomato*" to generalized quantifiers (Barwise and Cooper, 1981). Note that Eq. 2.2 represents such LLFs obtained from Eq. 2.1. The `LangPro` system is described in detail in Abzianidze's Ph.D. thesis (Abzianidze, 2016).

$$\mathbf{be}_{\langle \mathrm{np,np,s}\rangle}\ (\mathbf{no}_{\langle \mathrm{n,np}\rangle}\ (\mathbf{which}_{\langle \mathrm{vp,n,n}\rangle}\ (\mathbf{cut}_{\langle \mathrm{np,vp}\rangle}\ (\mathbf{a}_{\langle \mathrm{n,np}\rangle}\ \mathbf{tomato}_{\mathrm{n}}))\ \mathbf{person}_{\mathrm{n}}))\ \mathbf{there}_{\mathrm{np}} \tag{2.1}$$

$$\mathbf{no}_{\langle \mathrm{n,vp,s}\rangle}\ (\mathbf{which}_{\langle \mathrm{vp,n,n}\rangle}\ (\lambda y_{\mathrm{np}}.\ \mathbf{cut}_{\langle \mathrm{np,vp}\rangle}\ y\ x)\ \mathbf{person}_{\mathrm{n}})(\lambda z_{\mathrm{np}}.\ \mathbf{be}_{\langle \mathrm{np,np,s}\rangle}\ z\ \mathbf{there}_{\mathrm{np}}) \tag{2.2}$$

While it implements some techniques to fix parse errors, the accurate CCG parsing is important for the system, as demonstrated by the improved performance of a hybrid approach (Abzianidze, 2015) that combines RTE predictions obtained by parse trees of two different parsers.

[6] Note that, in Abzianidze's method, via subtyping rules, s, n, np, etc. also qualify as semantic types in much the same way as traditional entity "e" and truth value "t" types do.

Chapter 3

A* CCG Parsing

3.1 Introduction

In lexicalized grammar parsing, supertagging is known as *almost parsing* (Bangalore and Joshi, 1999), in that each supertag is syntactically informative and most ambiguities are resolved once a correct supertag (category) is assigned to each word. Recently this property has been effectively exploited in A* CCG parsing (Lewis and Steedman, 2014a; Lewis et al., 2016), in which the probability of a CCG tree $\boldsymbol{y}$ for sentence $\boldsymbol{x}$ of length N is the product of the probabilities of supertags c_i (*locally factored model*):

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{N} p_{tag}(c_i|\boldsymbol{x}). \tag{3.1}$$

By not modeling every combinatory rule in a derivation, this formulation enables us to employ an efficient A* search (Section 3.2) that finds the most probable supertag sequence that can construct a well-formed CCG tree.

Although supertagging can resolve considerable ambiguity, some ambiguity remains. Figure 3.1 shows an example, where two CCG parses are derived from the same supertags. Lewis et al. proposed an approach to this problem that involved a deterministic rule. For example, motivated by the right-branching tendency of English, Lewis et al. (2016) employ the attach low heuristics, and always prioritizes the result shown in Figure 3.1(b) for this type of ambiguity. Though this approach works well for English empirically, an obvious limitation is that it does not always derive the correct parse. For example, consider the phrase "*a house in Paris with a garden*", for which the structure of the correct parse corresponds to Figure 3.1(a).

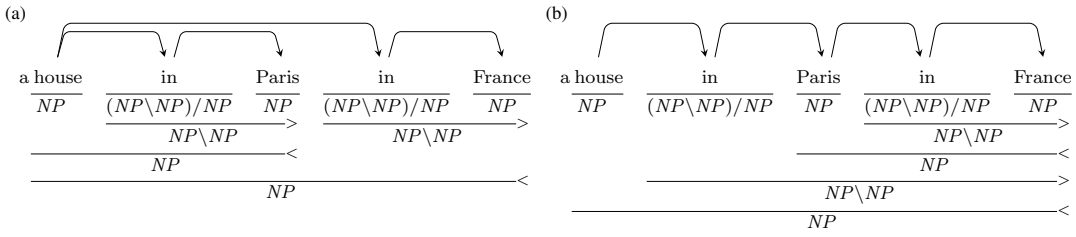


Figure 3.1: CCG trees that are equally likely under Eq. 3.1. The proposed model resolves this ambiguity by modeling the head of each word (dependencies).

In this chapter, we provide a way to resolve these remaining ambiguities under the locally factored model by explicitly modeling bi-lexical dependencies, as shown in Figure 3.1.[1] The proposed model is still locally factored so that an efficient A* search can be applied. The key idea is to predict the head of each word independently, as in Eq. 3.1, using a strong unigram model. Here we utilize the scoring model employed in the recent successful graph-based dependency parsing on LSTMs (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017). Specifically, we extend the bidirectional LSTM architecture of Lewis et al. (2016) to predict the supertag and head of a word simultaneously using a bilinear transformation.

The importance of modeling structures beyond supertags is demonstrated by the performance gain reported by Lee et al. (2016), which adds a recursive component to the model of Eq. 3.1. Unfortunately, compared to the original formulation, this formulation is less efficient because it must compute a recursive neural network each time it searches for a new node. The proposed model does not resort to the recursive networks while modeling tree structures via dependencies.

We also extend the tri-training method of (Lewis et al., 2016) to learn our model with dependencies from unlabeled data. With this technique, on English CCGbank test data, the proposed model achieves 88.8% and 94.0% in terms of labeled and unlabeled F1, which mark the best scores so far.

In addition to English, we conducted experiments on Japanese CCG parsing. In Japanese, word order is relatively free and is dominated by case markers; thus, a deter-

[1] In English CCG parsing literature, the evaluation is performed in terms of "semantic dependencies", i.e., a meaning representation similar to predicate argument structure that is obtainable from a CCG parse tree. Some studies in the literature define "*dependency models*", where the optimization is done in terms of this meaning representation (Section 3.8). Our work is different in that we utilize "syntactic dependencies", a common syntactic representation in dependency parsing literature. In this work, unless otherwise stated, the term "dependencies" stands for syntactic dependencies.

ministic rule, such as the attach low method, may not work well. The proposed method outperforms the simple application of (Lewis et al., 2016) by a large margin, i.e., 10.0 points in terms of bunsetsu[2] dependency accuracy.

CCG parsing is an important intermediate step of logic-based RTE systems (Section 2.5); thus, CCG parsing accuracy significantly affects RTE results. We evaluate the proposed method in terms of RTE by combining it with existing state-of-the-art RTE systems: `ccg2lambda` and `LangPro`. We observe that it consistently leads to better recalls, showing that it can parse a broader range of sentences robustly.

3.2 Background

Our work is built on A* CCG parsing (Section 3.2.1), which we extend in Section 3.3 with a head prediction model based on bidirectional LSTM (Section 3.2.2).

3.2.1 Supertag-factored A* CCG Parsing

CCG has a nice property that every category is highly informative about attachment decisions and assigning it to every word (*supertagging*) resolves most of its syntactic structure. Previous work (Lewis and Steedman, 2014a) utilizes this characteristics of the grammar. Let a CCG tree $\boldsymbol{y}$ be a list of categories $(c_1, \ldots, c_N)$ and a derivation on it. Their model looks for the most probable $\boldsymbol{y}$ given sentence $\boldsymbol{x}$ of length N from the set $\mathcal{Y}(\boldsymbol{x})$ of possible CCG trees under the model of Eq. 3.1:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \sum_{i=1}^{N} \log p_{tag}(c_i | \boldsymbol{x}).$$

Since this score is factored into each supertag, they call the model a *supertag-factored* model.

Exact inference of this problem is possible by A* parsing (Klein and D. Manning,

[2] A *bunsetsu* is a Japanese phrasal unit consisting of one or more adjoining content words (noun, verb, adjective, etc.) and zero or more functional words (postposition, auxiliary verb, etc.) (Hashimoto, 1934).

2003), which uses the following two scores on a chart:

$$b(C_{i,j}) = \sum_{c_k \in \boldsymbol{c}_{i,j}} \log p_{tag}(c_k|\boldsymbol{x}),$$

$$a(C_{i,j}) = \sum_{\substack{k=1 \\ k \notin \{i,...,j\}}}^{N} \max_{c_k} \log p_{tag}(c_k|\boldsymbol{x}),$$

where $C_{i,j}$ is a chart item called an *edge*, which abstracts parses spanning interval $\{i, ..., j\}$ rooted by category C. The chart maps each edge to the derivation with the highest score, i.e., the Viterbi parse for $C_{i,j}$. $\boldsymbol{c}_{i,j}$ ($= \boldsymbol{c}^C_{i,j}$) is the sequence of categories[3] for such a Viterbi parse, and thus b is called the Viterbi inside score, while a is the approximation (upper bound) of the Viterbi outside score.

A* parsing is a kind of CKY chart parsing augmented with an *agenda*, a priority queue that keeps the edges to be explored. At each step it pops the edge e with the highest priority $b(e) + a(e)$, inserts that into the chart, and enqueues any edges that can be built by combining e with other edges in the chart. The algorithm terminates when an edge $C_{1,N}$ is popped from the agenda.

The A* search in this model is quite efficient because both b and a can be obtained from the unigram category distribution on all words, which can be precomputed prior to a search. The heuristics a gives an upper bound on the true Viterbi outside score (i.e., admissible). In addition, the condition that the inside score never increases by expansion (monotonicity) guarantees that the first found derivation on $C_{1,N}$ is always optimal. $a(C_{i,j})$ matches the true outside score if the one-best category assignments on the outside words ($\arg\max_{c_k} \log p_{tag}(c_k|\boldsymbol{x})$) can comprise a well-formed tree with $C_{i,j}$. As such, a tighter upper bound (and hence more efficient parsing) can be achievable if the employed scoring model is more accurate. This is possible with recent neural network-based models described next.

Scoring model For modeling p_{tag}, a log-linear model with features from a fixed window context is used in Lewis and Steedman (2014a). Later, the log-linear model is extended with a bidirectional LSTM, which encode the complete sentence and capture the long range syntactic information (Lewis et al., 2016). We base the proposed model on this bidirectional LSTM architecture and extend it such that a head word can be modeled simultaneously.

[3]We omit the dependence on C for simplicity when it is evident from the context. This rule applies to the variable $\boldsymbol{d}^C_{i,j}$ of a dependency structure, introduced shortly.

Attachment ambiguity In an A* search, an edge with the highest priority $b + a$ is searched first; however, as shown in Figure 3.1 the same categories (with the same priority) may sometimes derive more than one tree. In Lewis and Steedman (2014a), they prioritize the parse with longer dependencies, which they determine using a conversion rule from a CCG tree to a dependency tree (Section 3.4). It is later replaced with another heuristics that prioritizes low attachments of constituencies (Lewis et al., 2016); however inevitably these heuristics cannot be flawless in any situations. We provide a simple solution to this problem by modeling bi-lexical dependencies explicitly.

3.2.2 Bidirectional LSTM Dependency Parsing

To model dependencies, we borrow the idea from the recent graph-based neural dependency parsing (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017) where each dependency arc is scored directly on the outputs of a bidirectional LSTM. Although the model is first-order, the bidirectional LSTM enable conditioning on the entire sentence and lead to the state-of-the-art performance. Note that this mechanism is similar to modeling of the supertag distribution discussed above, in that, for each word, the distribution of the head choice is unigram and can be precomputed. As we will see this keeps our joint model still locally factored and A* search tractable. For score calculation, we use an extended bilinear transformation proposed by Dozat and Manning (2017) that models the headness of each token as well, which they call *biaffine*.

3.3 Proposed Method

3.3.1 A* Parsing with Supertag and Dependency Factored Model

We define a CCG tree $\boldsymbol{y}$ for sentence $\boldsymbol{x} = (x_1, \ldots, x_N)$ as a pair of CCG categories $\boldsymbol{c} = (c_1, \ldots, c_N)$ and dependencies $\boldsymbol{d} = (d_1, \ldots, d_N)$,[4]where $d_i \in \{0, ..., N\}$ is the head index of x_i (0 is the index for the root token). The proposed model is defined as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{N} p_{tag}(c_i|\boldsymbol{x}) \prod_{i=1}^{N} p_{dep}(d_i|\boldsymbol{x}). \tag{3.2}$$

[4]In this work, we assume a pair $\boldsymbol{c}$ and $\boldsymbol{d}$ uniquely defines a CCG tree $\boldsymbol{y}$, if one exists. This is a sound assumption empirically, but strictly not true.

The added term p_{dep} is a unigram distribution of the head choice.

Given that supertags determine syntactic relations among words, c_i and d_i terms naturally interact with each other. However, the independence assumption of these terms is one of the keys of our method, which makes A* search tractable under this model. The search problem is changed as follows:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \left(\sum_{i=1}^{N} \log p_{tag}(c_i|\boldsymbol{x}) + \sum_{i=1}^{N} \log p_{dep}(d_i|\boldsymbol{x}) \right),$$

and the inside score is expressed as follows:

$$b(C_{i,j}) = \sum_{c_k \in \boldsymbol{c}_{i,j}} \log p_{tag}(c_k|\boldsymbol{x}) + \sum_{\substack{d_k \in \boldsymbol{d}_{i,j} \\ k \neq root(\boldsymbol{d}_{i,j})}} \log p_{dep}(d_k|\boldsymbol{x}), \tag{3.3}$$

where $\boldsymbol{d}_{i,j}$ is a dependency subtree for the Viterbi parse on $C_{i,j}$, and $root(\boldsymbol{d})$ returns the root index. We exclude the head score for the subtree root token because its head does not appear in $\{i, ..., j\}$. This causes the discrepancy between the goal inside score $b(C_{1,N})$ and the true model score (log of Eq. 3.2), which we adjust by applying a special unary rule to the goal edge $C_{1,N}$ that adds $\log p_{dep}(d_{root(\boldsymbol{d}_{1,N})} = 0|\boldsymbol{x})$.

We can calculate the dependency terms in Eq. 3.3 on the fly when expanding the chart. Let the currently popped edge be $A_{i,k}$, which will be combined with $B_{k,j}$ to form a new subtree in $C_{i,j}$. The key observation is that only one dependency arc (between $root(\boldsymbol{d}^A_{i,k})$ and $root(\boldsymbol{d}^B_{k,j})$) is determined at every combination (Figure 3.2). For every rule $C \rightarrow A\ B$ we can define the head direction (Section 3.4) and p_{dep} is obtained accordingly. For example, when the right child B becomes the head, $b(C_{i,j}) = b(A_{i,k}) + b(B_{k,j}) + \log p_{dep}(d_l = m|\boldsymbol{x})$, where $l = root(\boldsymbol{d}^A_{i,k})$ and $m = root(\boldsymbol{d}^B_{k,j})$ $(l < m)$.

The Viterbi outside score is changed as follows:

$$a(C_{i,j}) = \sum_{\substack{k=1 \\ k \notin \{i,...,j\}}}^{N} \max_{c_k} \log p_{tag}(c_k|\boldsymbol{x}) + \sum_{\substack{k=1 \\ k \notin \{i,...,j\} \setminus \{root(\boldsymbol{d}_{i,j})\}}}^{N} \max_{d_k} \log p_{dep}(d_k|\boldsymbol{x}), \tag{3.4}$$

We consider $root(\boldsymbol{d}_{i,j})$ an outside word because its head is not yet defined. For every outside word we independently assign the score of its argmax head, which may not comprise a well-formed dependency tree. We initialize the agenda by adding an item for every supertag C and word x_i with the score $a(C_{i,i}) =$

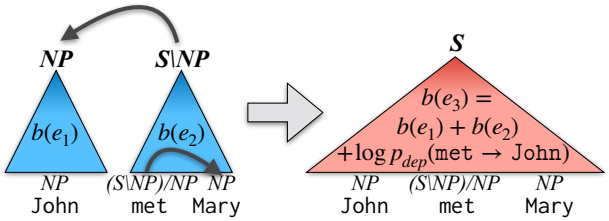


Figure 3.2: In the proposed model, the Viterbi inside score for edge e_3 is the sum of the e_1 and e_2 scores and the score of dependency arc going from the head of e_2 to that of e_1. Note that the head direction changes according to the child categories.

$\sum_{k=1,k\neq i}^{N} \max_{c_k} \log p_{tag}(c_k|\boldsymbol{x}) + \sum_{k=1}^{N} \max_{d_k} \log p_{dep}(d_k|\boldsymbol{x})$. Note that the dependency component of $a(C_{i,i})$ is the same for every word.

Note that the proposed model has the same advantages of the supertag-factored model (Section 3.2.1); $a(C_{i,j})$ defined above is admissible and monotone, guaranteeing the optimality of the first found tree. The upper bound is similarly the tightest if the one-best category and head assignments can form a CCG tree, implying the importance of an accurate scoring model.[5]

3.3.2 Network Architecture

Following Lewis et al. (2016) and Dozat and Manning (2017), we model p_{tag} and p_{dep} using a bidirectional LSTM to exploit the entire sentence to capture the long range phenomena. The overall network architecture is shown in Figure 3.3, where p_{tag} and p_{dep} share common LSTM hidden vectors.

First we map each word x_i to their hidden vector $\boldsymbol{h}_i$ with a bidirectional LSTM. The input to the LSTMs is word embeddings, details of which is described in Section 3.6. We add special start and end tokens to each sentence with trainable parameters following Lewis et al. (2016).

[5]Running a maximum spanning tree (MST) algorithm (e.g., Eisner (1996b)) on p_{dep} terms can give tighter estimates on the Viterbi outside score in case the one-best head assignments do not form a tree. However, given that neural network-based head prediction is so accurate that the output fails to form a tree only rarely, the computational complexity of the MST algorithm is rather harmful.

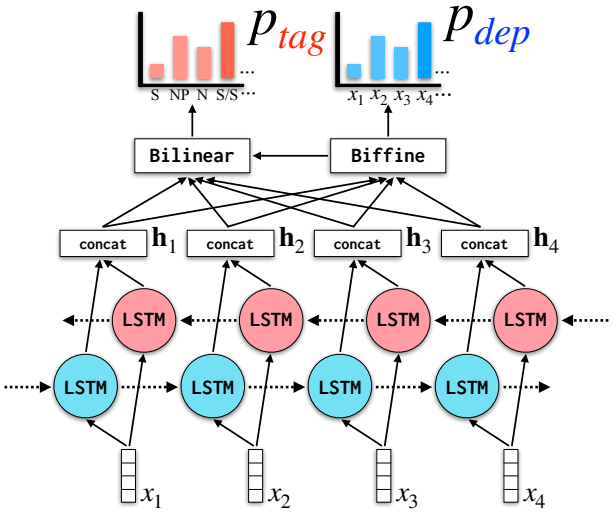


Figure 3.3: Neural networks of the proposed supertag and dependency factored model. First, we map each word x_i to a hidden vector $\boldsymbol{h}_i$ using a bidirectional LSTM, and then apply biaffine (Eq. 3.5) and bilinear (Eq. 3.6) transformations to obtain the distributions of dependency heads p_{dep} and supertags p_{tag}.

For p_{dep}, we use the biaffine transformation in (Dozat and Manning, 2017):

$$\begin{aligned} \boldsymbol{r}_i &= \psi^{dep}_{child}(\boldsymbol{h}_i), \quad \boldsymbol{r}_j = \psi^{dep}_{head}(\boldsymbol{h}_j), \\ s_{i,j} &= \boldsymbol{r}_i^\top W \boldsymbol{r}_j + \boldsymbol{w}^\top \boldsymbol{r}_j, \\ p_{dep}(d_i = j|\boldsymbol{x}) &\propto \exp(s_{i,j}), \end{aligned} \tag{3.5}$$

where ψ is a multi-layer perceptron. Although previous work (Lewis et al., 2016) simply uses a multi-layer perceptron to map $\boldsymbol{h}_i$ to p_{tag}, we also utilize the hidden vector of the most probable head $\hat{d}_i = \arg\max_j p_{dep}(d_i = j|\boldsymbol{x})$ (which is not necessarily identical to the head in the final output tree), and apply $\boldsymbol{h}_i$ and $\boldsymbol{h}_{\hat{d}_i}$ to a bilinear function:[6]

$$\begin{aligned} \boldsymbol{q}_i &= \psi^{tag}_{child}(\boldsymbol{h}_i), \quad \boldsymbol{q}_{\hat{d}_i} = \psi^{tag}_{head}(\boldsymbol{h}_{\hat{d}_i}), \\ s_{i,c} &= \boldsymbol{q}_i^\top W_c \boldsymbol{q}_{\hat{d}_i} + \boldsymbol{v}_c^\top \boldsymbol{q}_i + \boldsymbol{u}_c^\top \boldsymbol{q}_{\hat{d}_i} + b_c, \\ p_{tag}(c_i = c|\boldsymbol{x}) &\propto \exp(s_{i,c}). \end{aligned} \tag{3.6}$$

Formulating the prediction of p_{tag} this way performs the best among other settings that remove or reverse the dependence between the head model and the supertag model

[6]This is inspired by the formulation of label prediction in (Dozat and Manning, 2017).

empirically. Despite the actual dependence of supertags on a dependency structure in the neural network prediction, we formulate our probabilistic model of a CCG tree as in Eq. 3.2, where they are treated as independent. As in Lewis et al., these values can be precomputed prior to search, which makes our A* parsing quite efficient.

3.4 CCG to Dependency Conversion

In this section, we describe our conversion rules from a CCG tree to a dependency one, which we use in two purposes: (1) creation of the training data for the dependency component of our model; and (2) extraction of a dependency arc at each combinatory rule during A* search (Section 3.3.1). A previous work (Lewis and Steedman, 2014a) describes one way to extract dependencies from a CCG tree (LEWISRULE). In addition to LEWISRULE, we describe two simpler alternatives, i.e., HEADFIRST and HEADFINAL, and verify the effects on parsing performance in our experiments (Section 3.6). An overview of the CCG to dependency conversion is shown in Figure 3.4.

3.4.1 LEWISRULE

The LEWISRULE is the same as the conversion rule in (Lewis and Steedman, 2014a). As shown in Figure 3.4c, the output looks a familiar English dependency tree.

For forward application and (generalized) forward composition, we define the head to be the left argument of the combinatory rule, unless it matches either X/X or $X/(X\backslash Y)$, in which case the right argument is the head. For example, on *"Black Monday"* in Figure 3.4a, we select *Monday* as the head of *Black*. For the backward rules, the conversions are defined as the reverse of the corresponding forward rules. For other rules, *remove punctuation* ($<rp>$) selects the non-punctuation argument as the head, while *conjunction* ($<\Phi>$) selects the right argument.[7]

One issue when using this method to obtain the training data is that due to the mismatch between the rule set of our CCG parser, for which we follow Lewis and Steedman (2014a), and the grammar in English CCGbank (Hockenmaier and Steedman,

[7]When applying LEWISRULE to Japanese, we ignore feature values when determining the head argument, which often leads to a more natural dependency structure. For example, in "tabe ta" (eat *PAST*), the category of auxiliary verb "ta" is $S_{f_1}\backslash S_{f_2}$ with $f_1 \neq f_2$, thus $S_{f_1} \neq S_{f_2}$. In this case, we select "tabe" as the head by removing the feature values, which makes the category $X\backslash X$.

2007), we cannot extract dependencies from some of annotated CCG trees.[8] Thus, we instead obtain training data for this method from the original CCGbank dependency annotations. Fortunately, in most cases, the CCGbank dependency annotations match those of the LEWISRULE; thus they can well-approximate LEWISRULE dependencies.

3.4.2 HEADFINAL

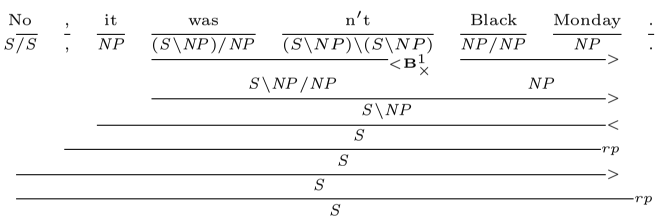
Among SOV languages, Japanese is a strictly head final language, i.e., the head of each word always follows the given word. Japanese dependency parsing (Uchimoto et al., 1999; Kudo and Matsumoto, 2002) has exploited this property explicitly by only allowing left-to-right dependency arcs. Inspired by this tradition, we try a simple HEADFINAL rule in Japanese CCG parsing, where we always select the right argument as the head. For example, we obtain the head final dependency tree shown in Figure 3.4e, from the Japanese CCG tree shown in Figure 3.4b.

3.4.3 HEADFIRST

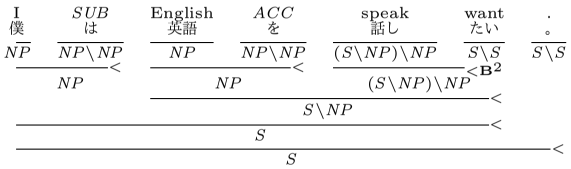
We apply an idea similar to HEADFINAL to English. Differing from Japanese, English is an SVO language; therefore, we define a simple "head first" rule whereby the left argument always becomes the head (Figure 3.4d).

Though this conversion may look odd at first, it has some advantages over the LEWISRULE. First, since a model that employs the LEWISRULE is trained on CCGbank dependencies, at inference, occasionally the two components p_{dep} and p_{tag} result in conflicting predictions. For example, the true Viterbi parse may have a lower score in terms of dependencies, in which case the parser slows down and may degrade the accuracy. HEADFIRST does not suffer from such conflicts. Second, because the arc directionality is fixed, the prediction of heads becomes easier. In Section 3.5, we demonstrate that this is the case for existing dependency parsers, and in practice, we find that the HEADFIRST conversion rule yields higher parsing scores than the LEWISRULE on English (Section 3.6).

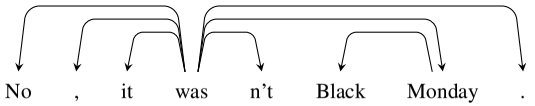
[8] For example, the combinatory rules in (Lewis and Steedman, 2014a) do not contain $N\ N \Rightarrow N_{conj}$ in CCGbank. Another difficulty is that in English CCGbank the name of each combinatory rule is not annotated explicitly.



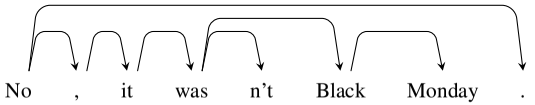
(a) English sentence



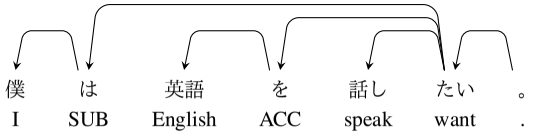
(b) Japanese sentence *"I want to speak English."*



(c) LEWISRULE



(d) HEADFIRST



(e) HEADFINAL

Figure 3.4: Examples of applying conversion rules (Section 3.4) to English and Japanese sentences.

3.5 Tri-training

We extend the existing tri-training method and apply it to our English parsers. Tri-training is one of the semi-supervised methods, where the outputs of two parsers on unlabeled data are intersected to create new (silver) training data. This method has been successfully applied to dependency parsing (Weiss et al., 2015) and CCG supertagging (Lewis et al., 2016).

We simply combine the two previous approaches. Lewis et al. (2016) obtained their silver data annotated with the high quality supertags by parsing One Billion Word Benchmark dataset (Chelba et al., 2014). Since they make this data publicly available,[9] we obtain our silver data by assigning dependency structures on top of them.

We train two very different dependency parsers on two types of training data extracted from CCGbank Section 02-21.[10] The datasets differ depending on our dependency conversion strategy employed (Section 3.4). For the LEWISRULE, we extract the original CCGbank dependency annotations. For the HEADFIRST, we extract the head first dependencies from the CCG trees. Note that we cannot annotate dependency labels; thus, we assign a dummy "none" label to each arc. The first parser is the graph-based `RBGParser` (Lei et al., 2014) with the default settings except that we train an unlabeled parser and use the word embeddings of Turian et al. (2010). The second parser is the transition-based `lstm-parser` (Dyer et al., 2015) with the default parameters.

On the development set (Section 00), with LEWISRULE dependencies the `RBGParser` shows 93.8% unlabeled attachment score while that of the `lstm-parser` is 92.5% using gold POS tags. Interestingly, the parsers with HEADFIRST dependencies achieve higher scores: 94.9% by `RBGParser` and 94.6% by `lstm-parser`, suggesting that HEADFIRST dependencies are easier to parse. For both dependencies, we obtain more than 1.7 million sentences on which two parsers agree.

Following Lewis et al. (2016), we include 15 copies of CCGbank training set when using these silver data. In addition, we reduce the effects of the tri-training samples by multiplying their loss by 0.4.

[9] https://github.com/uwnlp/taggerflow

[10] We annotate POS tags on this data using the Stanford POS tagger (Toutanova et al., 2003).

3.6 Parsing Experiments

In this section, we describe experiments performed to evaluate the parsing performance of the proposed method on English and Japanese CCGbanks.

3.6.1 English Experimental Settings

In the English experiment, we use the CCGBank dataset (Hockenmaier and Steedman, 2007), which was constructed by converting the Wall Street Journal part of the Penn Treebank (Marcus et al., 1994), and hence contains the same set of sentences annotated with CCG trees. The data split is also defined in the same way; Sections 02-21 for training, Section 00 for development, and Section 23 for final evaluation, each of which consists of 39,604 / 1,913 / 2,407 sentences. The English CCGbank dataset contains 1,030,345 running words with 41,478 (lowercased) word types, and the average sentence length is 23.45. We report labeled and unlabeled F1 scores of the extracted CCG semantic dependencies, following the standard CCG parsing evaluation protocol (Section 2.4).

For our models, we adopt the pruning strategies proposed by Lewis and Steedman (2014a) and allow at most 50 categories per word, use a variable-width beam with $\beta = 0.00001$, and utilize a tag dictionary, which maps frequent words to the possible supertags[11]. Unless otherwise stated, we only allow normal form parses (Eisner, 1996a; Hockenmaier and Bisk, 2010), choosing the same subset of the constraints as (Lewis and Steedman, 2014a).

For word representation, we use the concatenation of word vectors initialized to GloVe[12] (Pennington et al., 2014), and randomly initialized prefix and suffix vectors of length 1 to 4, which is inspired by Lewis et al. (2016). Affixes that only appear once in the training data are mapped to "UNK".

Other model configurations include: 4-layer bidirectional LSTM with left and right 300-dimensional LSTMs, 1-layer 100-dimensional multi-layer perceptrons with ELU non-linearity (Clevert et al., 2016) for all ψ^{dep}_{child}, ψ^{dep}_{head}, ψ^{tag}_{child} and ψ^{tag}_{head}, and the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.9$, L2 norm ($10^{-6}$), and learning rate decay with a ratio of 0.75 for every 2,500 iteration starting from 0.002, which has been shown to be effective for training the biaffine parser (Dozat and Manning, 2017).

[11] We use the tag dictionary provided with their bidirectional LSTM model.

[12] http://nlp.stanford.edu/projects/glove/

We compare the proposed method to the following baseline methods: C&C (Clark and Curran, 2007), EasySRL (Lewis et al., 2016), and neuralccg (Lee et al., 2016) (see Section 2.3 for the details). For C&C, we also report the performance of an extension with an LSTM supertagger (Vaswani et al., 2016), and for EasySRL, we also report results obtained with EasySRL_reimpl, our reimplemantation of EasySRL. Note that only C&C and its extension contain the part-of-speech tagging phase (which is implemented internally). The inputs to the other parsers are untagged raw sentences.

3.6.2 Japanese Experimental Settings

In this experiment, we use the Japanese CCGbank (Uematsu et al., 2015). We follow the default train / dev / test splits, each of which contains of 23,375 / 4,645 / 8,948 sentences. The Japanese CCGbank dataset contains 1,004,038 running words and 25,902 word types (both calculated in terms of lemma). The average sentence length is 27.15. As baselines, we use a shift-reduce CCG parser implemented in an NLP tool Jigg (Noji and Miyao, 2016)[13] (for simplicity, we refer to it as Jigg) with beam size $= 64$, and our adaptation of the supertag-factored model using a bidirectional LSTM.

For Japanese word representation, we use the concatenation of word vectors initialized to the Japanese Wikipedia Entity Vector[14], and 100-dimensional vectors computed from randomly initialized 50-dimensional character embeddings through convolution (dos Santos and Zadrozny, 2014). We do not use affix vectors as affixes are less informative in Japanese. Characters that only appear once in the training data are mapped to "UNK". For the bidirectional LSTM, multi-layer perceptrons, and optimization, we use the same parameter settings used in the English experiment.

One issue in Japanese experiments is evaluation. The Japanese CCGbank is encoded in a different format than the English bank, and no stand-alone script for extracting semantic dependencies is available yet. For this reason, we evaluate parser outputs by converting them to *bunsetsu dependencies*, the syntactic representation typically used in Japanese NLP (Kudo and Matsumoto, 2002). Given a CCG tree, we obtain bunsetsu dependencies by first segmenting a sentence into bunsetsu (chunks) using CaboCha[15]. After obtaining the word-level, head final dependencies as in Figure 3.4b, dependencies that cross a bunsetsu boundary are extracted. For example, the sentence in Figure

[13] https://github.com/mynlp/jigg
[14] http://www.cl.ecei.tohoku.ac.jp/~m-suzuki/jawiki_vector/
[15] http://taku910.github.io/cabocha/

Method	Labeled	Unlabeled
CCGbank		
LEWISRULE w/o dep	85.8	91.7
LEWISRULE	86.0	92.5
HEADFIRST w/o dep	85.6	91.6
HEADFIRST	**86.6**	**92.8**
Tri-training		
LEWISRULE	86.9	93.0
HEADFIRST	**87.6**	**93.3**

Table 3.1: Parsing results (F1) on English development set. "w/o dep" means that the model discards dependency components at prediction.

3.4e is segmented as "*Boku wa* | *eigo wo* | *hanashi tai*", from which we extract two dependencies (*Boku wa*) ← (*hanashi tai*) and (*eigo wo*) ← (*hanashi tai*). We perform this conversion for both gold and output CCG trees and calculate the (unlabeled) attachment accuracy (**Bunsetsu Dep.**). We also report the accuracy of pre-terminal supertag assignment (**Category**), because it is crucial in determining the overall parse structure (Section 3.2). Though this process is imperfect, it can detect important parse errors such as attachment errors and thus can be a good proxy to evaluate the performance of a CCG parser.

3.6.3 English Parsing Results

Effect of Dependency

We first see how the added dependency components in our model affect performance. Table 3.1 shows the results on the development set with several configurations. Here, "w/o dep" means the dependency terms of the model are discarded, and the attach low heuristics (Section 3.1) is used instead, i.e., a supertag-factored model (Section 3.2.1). As can be seen, adding dependency terms improves the performance for both LEWISRULE and HEADFIRST methods.

Method	Labeled	Unlabeled	# violations
CCGbank			
LEWISRULE w/o dep	85.8	91.7	2,732
LEWISRULE	85.4	92.2	283
HEADFIRST w/o dep	85.6	91.6	2,773
HEADFIRST	**86.8**	**93.0**	**89**
Tri-training			
LEWISRULE	86.7	92.8	253
HEADFIRST	**87.7**	**93.5**	**66**

Table 3.2: Parsing results (F1) on English development set when normal form constraints are excluded. The # violations column shows the number of combinations that violate the constraints on the outputs.

Choice of Dependency Conversion Rule

To our surprise, our simple HEADFIRST strategy always leads to better results than the linguistically motivated LEWISRULE (except for w/o dep settings). The absolute improvements by tri-training are equally large (approximately 0.5 points), suggesting that the proposed model with dependencies can also benefit from the silver data.

Excluding Normal Form Constraints

One advantage of HEADFIRST is that the arc direction is always right, which makes the structures simpler and more accurately parsable (Section 3.5). From another viewpoint, this fixed direction means that the constituent structure behind a (head first) dependency tree is almost unique.[16] Since the constituent structures of CCGbank trees basically follow the normal form (NF), we hypothesize that the model learned with HEADFIRST has an ability to force the outputs in NF automatically. We summarize the results without the NF constraints in Table 3.2. These results demonstrates that the above argument is correct; with HEADFIRST, the number of times NF rules are violated is significantly less than the number of time such rules are violated using

[16]There is a bijection between a head first dependency structure and an unlabeled binary constituency structure (modulo the use of unary rules). Basically we can recover a constituency tree by considering each dependency edge $i \rightarrow j$ as a constituent that spans from the i'th word to the rightmost descendant of the j'th word (or itself when there is none).

Method	Labeled	Unlabeled
CCGbank		
C&C (Clark and Curran, 2007)	85.5	91.7
w/ LSTMs (Vaswani et al., 2016)	**88.3**	-
EasySRL (Lewis et al., 2016)	87.2	-
EasySRL_reimpl	86.8	92.3
HEADFIRST w/o NF (Ours)	87.7	**93.4**
Tri-training		
EasySRL (Lewis et al., 2016)	88.0	92.9
neuralccg (Lee et al., 2016)	88.7	93.7
HEADFIRST w/o NF (Ours)	**88.8**	**94.0**

Table 3.3: Parsing results (F1) on English test set (Section 23).

LEWISRULE (89 vs. 283). Interestingly the scores of HEADFIRST models increase slightly compared to models with NF (e.g., 86.8 vs. 86.6 for CCGbank), suggesting that NF constraints occasionally hinder HEADFIRST model's search process.

Results on Test Set

Parsing results on the test set (Section 23) are shown in Table 3.3. In Table 3.3, we compare the HEADFIRST dependency model without NF constraints that returns the best results to several existing parsers. In the CCGbank experiment, the proposed parser demonstrates better results than all the baseline parsers, except for C&C with an LSTM supertagger (Vaswani et al., 2016). In terms of labeled F1, the proposed parser outperforms EasySRL and our reimplementation of that parser (EasySRL_reimpl) by 0.5% and 0.9%, respectively. In the tri-training experiment, the proposed parser's performance increases significantly, i.e., 88.8% for labeled F1 and 94.0% for unlabeled F1. Here, the proposed parser outperforms the state-of-the-art neuralccg (Lee et al., 2016), which uses recursive neural networks, by 0.1 and 0.3 points in terms of labeled and unlabeled F1, respectively. To the best of our knowledge, these are the best reported F1 scores for English CCG parsing. When three independent HeadFirst models are trained using the tri-training setup with different initial weight parameters, they have resulted in the labeled and unlabeled F1 scores of 88.46 ± 0.24 and 93.80 ± 0.13, on average.

	EasySRL_reimpl	neuralccg	Ours
Tagging	24.8	21.7	16.6
A Search*	185.2	16.7	114.6
Total	21.9	9.33	14.5

Table 3.4: Results of the efficiency experiment, where each number is the number of sentences processed per second. We compare our proposed parser against neuralccg and our reimplementation of EasySRL.

Efficiency Comparison

We compare the efficiency of the proposed parser to neuralccg and EasySRL_reimpl (Table 3.4).[17] For overall speed (the third row), the proposed parser is faster than neuralccg; however it is slower than EasySRL_reimpl. The proposed supertagger runs slower than those of neuralccg and EasySRL_reimpl. However, for A* search, the proposed parser processes over seven times more sentences than neuralccg. The delay in supertagging can be attributed to several factors, in particular the differences in network architectures including the number of bidirectional LSTM layers (4 vs. 2) and the use of bilinear transformation rather than linear transformation. In addition, there are many implementation differences in our parsers (C++ A* parser with a neural network model implemented with Chainer (Tokui et al., 2015)) and neuralccg (Java parser with C++ TensorFlow (Abadi et al., 2016) supertagger and recursive neural model in C++ DyNet (Neubig et al., 2017))[18].

3.6.4 Japanese Parsing Result

We show the results of the Japanese parsing experiment in Table 3.5, where the bunsetsu attachment accuracy of a Japanese dependency parser CaboCha (Kudo and Matsumoto, 2002) is included for reference. We train and evaluate the CaboCha parser on the same sets of sentences as CCGBank train and test sets, extracted from the Kyoto

[17]This experiment was conducted on a laptop with a 2-core 4-thread 2.0 GHz CPU.

[18]There seems to be room to optimize the proposed parser's efficiency. We found that our EasySRL_reimpl is slower than the original implementation of the supertag-factored model (Lewis et al., 2016), which neuralccg uses internally.

Method	Category	Bunsetsu Dep.
CaboCha (Kudo and Matsumoto, 2002)	-	91.7
Transition-based (Jigg; Noji and Miyao (2016))	93.0	87.5
Supertag model	93.7	81.5
LEWISRULE (Ours)	93.8	90.8
HEADFINAL (Ours)	**94.1**	**91.5**

Table 3.5: Results on the Japanese CCGbank.

University Text Corpus (Kawahara et al., 2002). It achieves a slightly better score, presumably because it is directly optimized for this metric. The simple application of the supertag-factored model (Lewis et al., 2016) is not effective for Japanese, showing the lowest attachment score of 81.5%. We observe a performance boost with the proposed method, particularly with HEADFINAL dependencies, which outperforms the baseline shift-reduce parser by 1.1 points on category assignments and 4.0 points on bunsetsu dependencies.

The reduced performance of the simple application of the supertag-factored model can be attributed to the fact that Japanese sentence structure is still highly ambiguous given supertags. This seems primarily due to the fact that the language has freer word ordering. Many parsing errors are observed around constructions where adverbial modifiers are involved (Figure 3.5). They are ambiguous in that which of succeeding verbs it modifies cannot be determined only from the modifier's supertag. The result suggests the importance of modeling dependencies in some languages, e.g., Japanese.

Comparing A* Parser and Shift-Reduce Parser

To the best of our knowledge, this is the first work that develops an A* CCG parsing method for the Japanese language. Jigg implements a transition-based parser adopting shift-reduce algorithm. One of the critical issues with a transition-based method for CCG is that, the sequence of predicted actions must be strictly consistent across time steps; there can occur a situation wherein the transition system gets stuck in the middle of parsing and thus it cannot output a tree, because there are no combinatory rules (correspondingly no transition actions) applicable to the root categories of the two top stack elements. To overcome this situation, the Jigg parser implements a heuristic rule that forcibly combines them into one. In fact, the Jigg parser encountered such

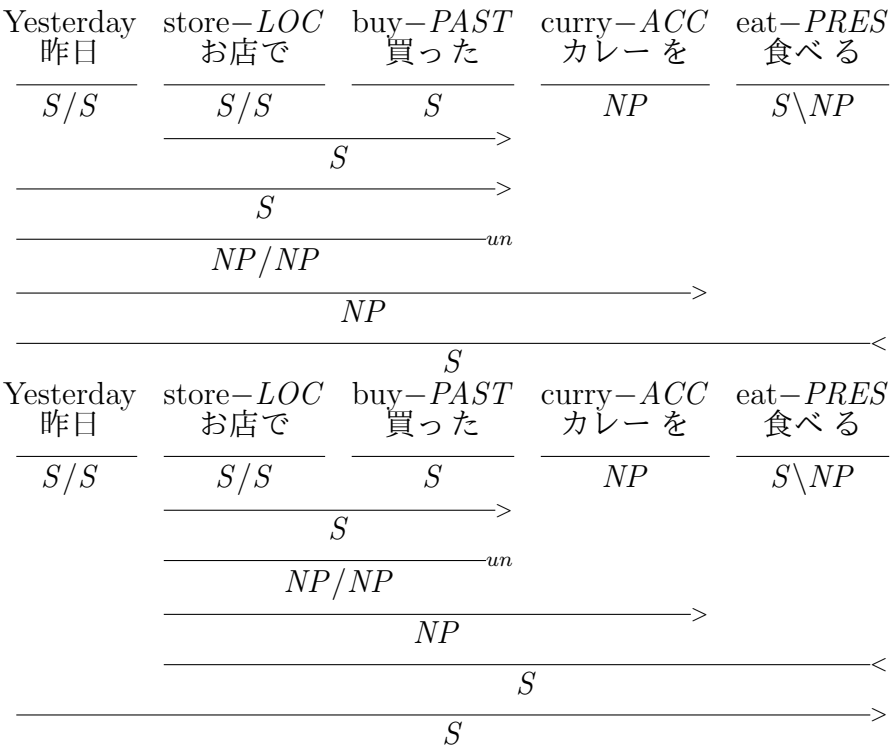


Figure 3.5: An ambiguous Japanese sentence given fixed supertags. The tree above is the more probable (The translation is *"I eat the curry I bought at a store yesterday"*). In the right below, it is unlikely that "昨日" (yesterday) modifies the last verb in the present tense, when it is immediately followed by "買った" (buy-*PAST*).

a situation in 7.2% (642 out of 8,948) of the test sentences. The proposed chart-based A* parser is guaranteed to output a well-formed tree. It should be emphasized that in the conducted experiments throughout this chapter (regardless of English or Japanese), the proposed method always outputs a tree.

Transition-based parsers are well-known for their efficiency, because their computational complexity only increases linearly in terms of the length of the input sentence. Since we have also emphasized the efficiency of the proposed method, which results from the combination of a locally-factored model and A* parsing, we compare the proposed method and the `Jigg` parser in terms of processing speed using the Japanese CCGBank test set.[19] We found that the proposed method processes 14.2 sentences per second while `Jigg` 16.0 sentences. Although, in terms of processing speed, the

[19]This experiment was conducted on a laptop with a 2-core 4-thread 2.5 GHz CPU.

transition-based parser outperforms the proposed parser, the gap is rather tolerable given the merits noted above.

3.7 RTE Experiments

In addition to the parsing experiments, we conduct experiments to evaluate the proposed method in terms of RTE task. RTE is an elemental technology for semantic analysis of multiple sentences, where, given premise(s) and a hypothesis, a system predicts whether the former entails (*entailment*), contradicts (*contradiction*), or is neutral to (*unknown*), the latter. The experiments are performed by combining the proposed CCG parser with `ccg2lambda` and `LangPro`, logic-based RTE systems that reason over meaning representations converted from CCG trees (Section 2.5). Previously, it has been reported that one of such systems shows significantly improved RTE performance by using gold CCG trees rather than predicted ones (from 74.1% to 87.3% accuracy, on the JSeM dataset), implying the huge impact of the accuracy of CCG parsing on RTE (Mineshima et al., 2016). We evaluate how much the parsing performance achieved in the previous section transfers to one of the other application tasks.

3.7.1 Experimental Settings

English

In the English experiment, we test the performance of `ccg2lambda` and `LangPro` combined with the proposed CCG parser on the SICK dataset (Marelli et al., 2014). This dataset was originally developed to test approaches of compositional distributional semantics and includes a variety of syntactic, lexical, and semantic phenomena. The dataset contains 4,500 problems (a pair of premise and hypothesis) for training, 500 for trial and 4,927 for testing, with a ratio of *entailment* / *contradiction* / *unknown* problems of .29 / .15 / .56 in all splits. Table 3.6 shows some problems from the dataset, that contain complex syntactic phenomena such as PP-attachment, passive construction and coordination. It is evident that accurate CCG parsing is important to solve this dataset.

Here, we report results for `ccg2lambda` with the default settings (with SPSA abduction; Martínez-Gómez et al. (2017)) and results for two versions of `LangPro`, one described in (Abzianidze, 2015) (henceforth we refer to it as `LangPro15`) and the

ID	Premise and Hypothesis	Label
1761	P: *A flute is being played in a lovely way by a girl.* H: *One woman is playing a flute*	entailment
880	P: *The girl in the blue and white uniform is cheering.* H: *Some cheers are being performed by the girl in the blue and white uniform.*	entailment
3013	P: *The man on stage is singing into the microphone.* H: *There is no man in a suit standing at a microphone and singing .*	unknown

Table 3.6: Example RTE problems from the SICK dataset. To solve this dataset, capturing syntactic phenomena such as PP-attachment, passive construction and coordination accurately is as important as logical reasoning and external lexical knowledge.

other in (Abzianidze, 2017) (`LangPro17`). We report accuracy / precision / recall / F1 values[20] of the two RTE systems that use the best performing HEADFIRST parser without NF constraints trained on tri-training (hereafter `depccg`) as a syntactic parsing component instead of the baseline CCG parsers, i.e., `C&C` (Clark and Curran, 2007) and `EasyCCG` (Lewis and Steedman, 2014a).

It is well known that the CCG parsing accuracy is one of the bottlenecks of logic-based systems. To mitigate the error propagation, `ccg2lambda` has an option to use multiple CCG parsers and aggregates their results (*multi-parsing* setting). In this work, we also report RTE scores obtained by combining `depccg` with `C&C` and `EasyCCG`. We regard two results as contradicted with each other if one is *entailment* and the other is *contradiction*. In such cases the system outputs *unknown*; otherwise, if at least one parser results in *entailment* or *contradiction*, that is adopted as the system output.

Japanese

In the Japanese experiment, we evaluate the performance of the Japanese version of `ccg2lambda`, combined with the best performing HEADFINAL (`depccg`). We use the JSeM RTE dataset (Kawazoe et al., 2017) for evaluation. Following Mineshima et al. (2016), we report scores obtained for a subset of 523 problems, which includes Japanese translations of the FraCaS dataset (Cooper et al., 1994). These datasets focus on types of logical inferences that do not require world knowledge. The problems used

[20] In the RTE literature, these metrics are defined as follows: recall is the number of correctly predicted entailments divided by the total number of premise-hypothesis pairs given to a system; precision is the number of correctly predicted entailments divided by the number of predictions made by the system.

Section	Premise and Hypothesis	Label
Attitude	P: 花子は太郎が次郎の結婚式に来ることを期待していた。 "*Hanako expected that Taro went Jiro's wedding ceremony.*" H: 太郎は次郎の結婚式に来た。 "*Taro went Jiro's wedding ceremony.*"	unknown
Plural	P: 会議に出席していた人々全員が新しい議長に投票した。 "*All the people who were at the meeting voted for a new chairman.*" H: 会議に出席していたすべての人が新しい議長に投票した。 "*Everyone at the meeting voted for a new chairman.*"	entailment

Table 3.7: Example RTE problems from the JSeM dataset.

in the experiment are labeled with one of the tags from generalized quantifier, plural, adjective, verb, and attitude, and treat linguistic phenomena related to it. Table 3.7 shows the example problems. We use the same baseline parser `Jigg` (Noji and Miyao, 2016) used in the CCGBank experiment. We also report *multi-parsing* results obtained by combining `depccg` and `Jigg` (Section 3.7.1).

3.7.2 English RTE Results

Table 3.8 shows the experimental results on the SICK dataset. As can be seen, when combined with `LangPro15` and `LangPro17`, the proposed parser (`depccg`) contributes to consistent improvement in the accuracy and recall (with a drop in precision), which yields the improved overall F1 scores (0.87 % and 0.96 % up for both systems compared to `C&C`). When combined with `ccg2lambda`, the proposed parser shows the same tendency as `LangPro`, where better accuracy and recall yield the improved F1 score. This is also evident in the multi-parsing setting where the proposed `depccg` is used on top of `C&C` and `EasyCCG`.

Interestingly, the proposed parser always contributes to improved recall under all settings. When the proposed parser is used in combination with `C&C` and `EasyCCG`, greater than 1% recall improvement is observed, which indicates that the proposed parser complements existing parsers. Given the definition of recall in RTE, it can be speculated that the proposed parser can parse a broader range of sentences (in terms of constructions) robustly. In error analysis, we find that this is actually the case. In the SICK dataset, there are many examples of the "*there is*" construction, such as "There is no dog barking" (Figure 3.6), which contains a noun phrase followed by a

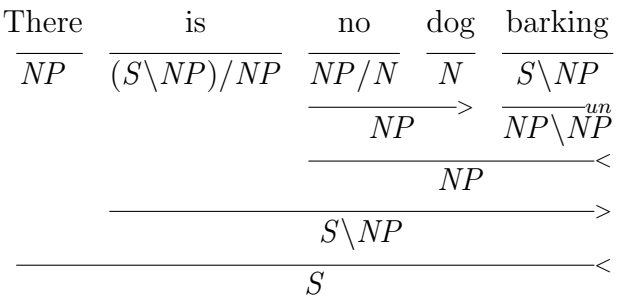


Figure 3.6: In SICK, there are many examples of the "*there is* $[_{NP}$ *NP PRP*]" construction (*PRP*: present participle adjective). Existing parsers incorrectly analyze the *NP* constituent as a noun preceded by modifiers (*N/N N/N N*).

present participle adjective (with categories *NP/N*, *N*, *S\NP*). We have observed cases where `C&C` and `EasyCCG` incorrectly parse the phrase as "*N/N N/N N*", where the last adjective is analyzed as a noun modified by the preceding words. In contrast, the proposed parser correctly predicts the structure shown in Figure 3.6.

3.7.3 Japanese RTE Results

In the Japanese experiment, the proposed parser has resulted in poor RTE performance compared to the baseline Jigg parser (Table 3.9). We hypothesize that this is due to the fact that the previous work created the semantic templates for this language by analyzing Jigg's parse outputs (and thus designed them to absorb the errors of a specific parser). We speculate that this resulted in a kind of "overfitting" in the templates. In future work, we will conduct a detailed investigation of the issue. The RTE performance has significantly improved when the parsers are combined in multi-parsing setting. Among 538 test problems, `ccg2lambda` combined with `depccg` has solved 30 problems for which the one with `jigg` has failed. Conversely, the latter is successful in 44 problems for which the former is not, which further indicates that the proposed parser works complementarily to existing parsers, which is in accordance with the English RTE experiments.

One critical issues with our model is that it does not consider the probability of applying a unary rule directly. In the Japanese CCGBank, the *ADNint* and *ADNext* unary rules are employed to account for two classes of sentential noun modification (Teramura, 1969): "inner relationship", a relative clause equivalent in Japanese that involves

Method	Accuracy	Precision	Recall	F1
LangPro15 (Abzianidze, 2015)				
C&C	79.93	97.99	54.73	70.23
EasyCCG	79.05	**98.00**	52.67	68.51
depccg	**80.37**	97.94	**55.81**	**71.10**
LangPro17 (Abzianidze, 2017)				
C&C	81.04	**97.62**	57.55	72.41
EasyCCG	81.04	97.47	57.69	72.48
depccg	**81.53**	97.51	**58.81**	**73.37**
ccg2lambda (Martínez-Gómez et al., 2017)				
C&C	**81.95**	96.81	59.88	73.99
EasyCCG	81.59	**97.73**	58.48	73.17
depccg	**81.95**	97.19	**59.98**	**74.18**
C&C + EasyCCG	83.13	**97.04**	63.64	77.14
C&C + EasyCCG + depccg	**83.82**	96.57	**64.66**	**77.46**

Table 3.8: RTE results on the test section of SICK

argument extraction from within the subordinate clause (Figure 3.7a), and "outer relationship", another broader class of sentential modifier that includes complements (Figure 3.7b). As such, these unary rules result in largely different logical formulas in semantic parsing. In the case of ccg2lambda, the former is converted to a conjunctive modifier, while the latter to a higher-order *Content* predicate, as shown in Figure 3.7. Among the 44 problems for which only Jigg is successful, we have found examples where the proposed method predicts *ADNint* where *ADNext* is appropriate (and vice versa). Currently, the adequacy of applying either of these rules is modeled indirectly through the probability of the structure of the modifier clause.

In the current evaluation protocol of Japanese CCG parsing (Section 3.6.2), misprediction of either of these unary rules is captured via the **Category** metric (cf. Figure 3.7). We approximate the accuracy of the use of *ADNint*/*ADNext* by computing that metric only on those verbs (determined based on their gold part-of-speech tag) that appear under these unary rules in the Japanese CCGBank test set (5,171 test sentences contain at least one such verb occurrence, resulting in 12,227 verbs considered in total). Note that this is considered an "approximation" because not all of the verbs may be involved with the constructions. The result is 83.5%, which is considerably low

Method	Accuracy	Precision	Recall	F1
jigg	74.76	92.62	65.12	76.47
depccg	69.40	88.74	59.07	70.92
jigg + depccg	79.34	90.52	74.35	81.64

Table 3.9: RTE results using ccg2lambda on JSeM

given that the metric on all words in the dataset is 94.1% (Table 3.5). Actually this is one of the causes of the large performance drop of the system combined with depccg alone. Its failure in analyzing the sentence "会議に出席していたすべての人が新しい議長に投票した。"(*Everyone at the meeting voted for a new chairman.*), which is contained in four test problems, has resulted in the failure of theorem proving on all of the problems. Extending the proposed method to model unary rules explicitly is an important future direction relative to realizing robust semantic parsing using CCG.

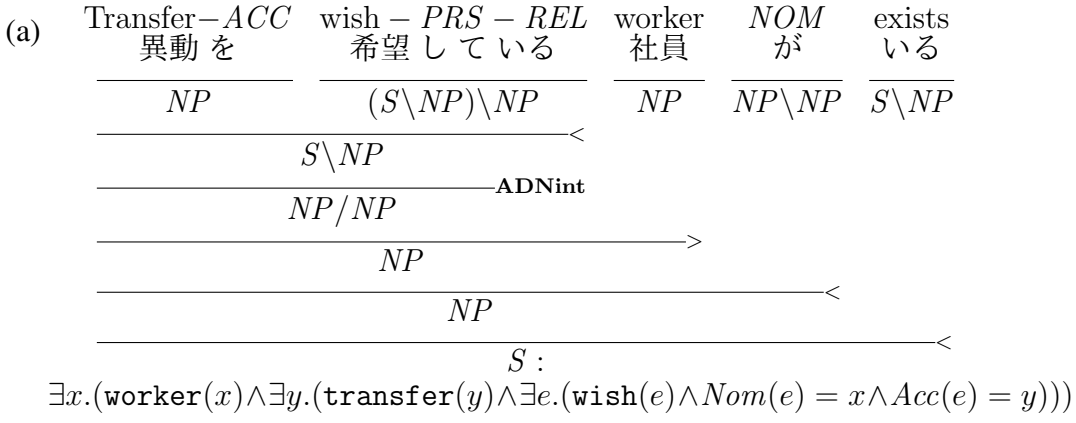
3.8 Related Work

Previous studies have utilized dependencies in lexicalized grammar parsing.

For Head-driven Phrase Structure Grammar (HPSG; Pollard and Sag (1994)), several studies used the predicted dependency structure to improve the accuracy of HPSG parsing. In (Sagae et al., 2007), the form of the output tree is constrained to match the dependencies. As in our method, for each rule (schema) application, they define which child becomes the head and impose a soft constraint that these dependencies agree with the output of a dependency parser. The proposed method is different in that we do not use the one-best dependency structure alone. Rather, we search for a CCG tree that is optimal in terms of dependencies and CCG supertags. Zhang et al. (2010) analyzes that capturing two syntactic properties, i.e., the subject and complements, is important for HPSG supertagging. Based on this observation, they developed a two-stage method that first parses the input sentence into a dependency structure and then uses dependency-based discrete features for an averaged perceptron-based supertagger, which results in improved tagging accuracy.

In CCG parsing literature, some studies have focused on the optimization of *dependency models* (Xu et al., 2014; Clark and Curran, 2007), which are a class of models defined over semantic dependencies obtained from a CCG tree (similar to a predi-

(a)



(b)

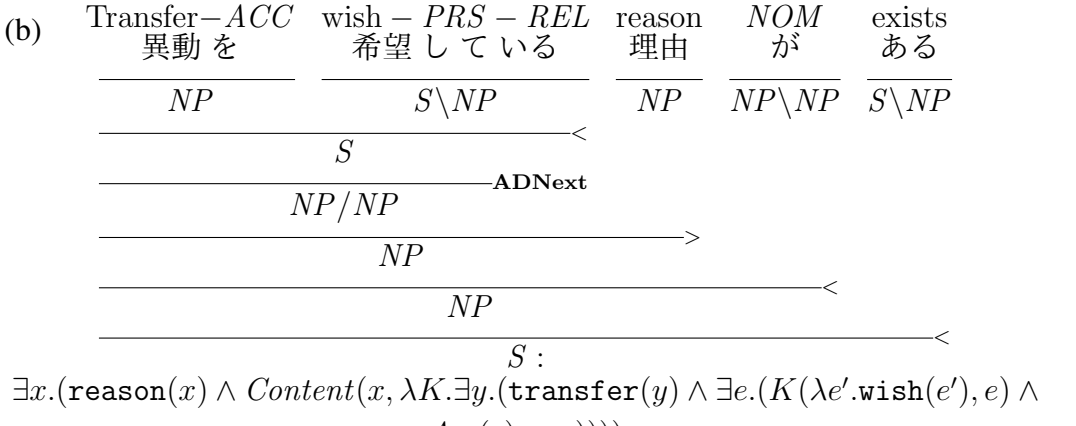


Figure 3.7: Examples of the use of *ADNint* (inner relationship) and *ADNext* (outer relationship) unary rules. The English translation is (a) *"There exists a worker who wants to transfer his post"* and (b) *"There is a reason why he wants to transfer his post"*.

cate argument structure). Such optimization is typically achieved by summing over all derivations that realize the same semantic dependencies. This approach is reasonable given that the objective matches the evaluation metric. Rather than modeling only dependencies, the proposed method finds a CCG derivation with a higher dependency score. CCG-based semantic dependencies are also used as features in joint CCG parsing and semantic role labeling (Lewis et al., 2015); however such features are ineffective to recover long-range dependencies over LSTMs in CCGbank parsing (Lewis et al., 2016).

3.9 Summary

We have proposed an A* CCG parsing method, in which the probability of a CCG tree is decomposed into local factors of the CCG categories and its dependency structure. By modeling the dependency structure explicitly, we do not require any deterministic heuristics to resolve attachment ambiguities and can keep the model locally factored such that all probabilities can be precomputed prior to running the search. The proposed parser finds the optimal parses efficiently and achieves the state-of-the-art performance in both English and Japanese parsing. We also evaluated the proposed parser in terms of logic-based RTE systems. The experimental results demonstrate the proposed method's robustness against a wide range of constructions while also highlighting the future direction of improvement in terms of the use of unary rules.

Chapter 4

Domain Adaptation for CCG Parsing

4.1 Introduction

The recent advancement of CCG parsing, combined with formal semantics, has enabled high-performing natural language inference systems (as introduced in Section 2.5). Our interest in this chapter is to transfer the success to a range of applications, such as building inference systems on scientific papers and speech conversation.

To achieve the goal, it is urgent to enhance the CCG parsing accuracy on new domains, i.e., solving a notorious problem of *domain adaptation* of a statistical parser, which has long been addressed in the literature. Especially in CCG parsing, prior work (Rimell and Clark, 2008; Lewis et al., 2016) has taken advantage of highly informative categories, which determine the most part of sentence structure once correctly assigned to words. It is demonstrated that the annotation of only pre-terminal categories is sufficient to adapt a CCG parser to new domains. However, the solution is limited to a specific parser's architecture, making non-trivial the application of the method to the current state-of-the-art parsers (Section 2.3), which require full parse annotation. Additionally, some ambiguities remain unresolved with mere supertags, especially in languages other than English, as discussed in the previous chapter (Section 3.6.2), to which the method is not portable.

Distributional embeddings are proven to be powerful tools for solving the issue of domain adaption, with their unlimited applications in NLP, not to mention syntactic parsing (Lewis and Steedman, 2014b; Mitchell and Steedman, 2015; Peters et al., 2018). Among others, Joshi et al. (2018) reports huge performance boosts in constituency parsing using contextualized word embeddings (Peters et al., 2018), which is orthogonal to our work, and the combination shows huge gains. Including (Joshi et al.,

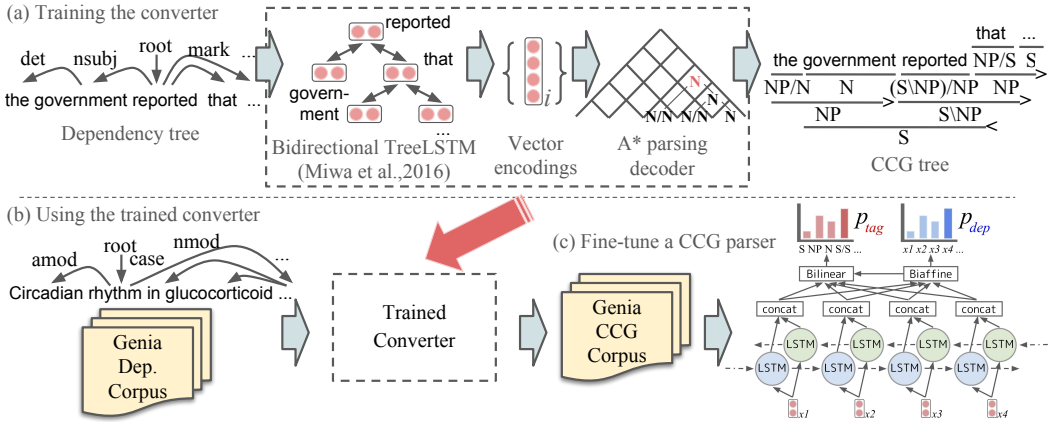


Figure 4.1: Overview of the proposed method. (a) A neural network-based model is trained to convert a dependency tree to a CCG one using aligned annotations on WSJ part of the Penn Treebank and the English CCGbank. (b) The trained converter is applied to an existing dependency corpus (e.g., the Genia corpus) to generate a CCGbank, (c) which is then used to fine-tune the parameters of an off-the-shelf CCG parser.

2018), there are studies to learn from partially annotated trees (Mirroshandel and Nasr, 2011; Li et al., 2016; Joshi et al., 2018), again, most of which exploit specific parser architecture.

In this work, we propose a conceptually simpler approach to the issue, which is agnostic on any parser architecture, namely, *automatic generation of CCGbanks* (i.e., CCG treebanks)[1] for new domains, by exploiting cheaper resources of dependency trees. Specifically, we train a deep conversion model to map a dependency tree to a CCG tree, on aligned annotations of the Penn Treebank (Marcus et al., 1993) and the English CCGbank (Hockenmaier and Steedman, 2007) (Figure 4.1a). When we need a CCG parser tailored for a new domain, the trained converter is applied to a dependency corpus in that domain to obtain a new CCGbank (4.1b), which is then used to fine-tune an off-the-shelf CCG parser (4.1c). The assumption that we have a dependency corpus in that target domain is not demanding given the abundance of existing dependency resources along with its developed annotation procedure, e.g., Universal Dependencies (UD) project (Nivre et al., 2016), and the cheaper cost to train an annotator.

[1]In this chapter, we call a treebank based on CCG grammar *a CCGbank*, and refer to the specific one constructed in (Hockenmaier and Steedman, 2007) as *the English CCGbank*.

One of the biggest bottlenecks of syntactic parsing is handling of countless *unknown words*. It is also true that there exist such unfamiliar input data types to our converter, e.g., disfluencies in speech and symbols in math problems. We address these issues by *constrained decoding* (Section 4.4), enabled by incorporating a parsing technique into our converter. Nevertheless, syntactic structures exhibit less variance across textual domains than words do; our proposed converter suffers less from such unseen events, and expectedly produces high-quality CCGbanks.

The work closest to ours is (Jiang et al., 2018), where a conversion model is trained to map dependency treebanks of different annotation principles, which is used to increase the amount of labeled data in the target-side treebank. Our work extends theirs and solves a more challenging task; the mapping to learn is to more complex CCG trees, and it is applied to datasets coming from plainly different natures (i.e., domains). Some prior studies design conversion algorithms to induce CCGbanks for languages other than English from dependency treebanks (Bos et al., 2009; Ambati et al., 2013). Though the methods may be applied to our problem, they usually cannot cover the entire dataset, consequently discarding sentences with characteristic features. On top of that, unavoidable information gaps between the two syntactic formalisms may at most be addressed probabilistically.

To verify the generalizability of our approach, on top of existing benchmarks on (1) **biomedical texts** and (2) **question sentences** (Rimell and Clark, 2008), we conduct parsing experiments on (3) **speech conversation texts**, which exhibit other challenges such as handling informal expressions and lengthy sentences. We create a CCG version of the Switchboard corpus (Godfrey et al., 1992), consisting of full train/dev/test sets of automatically generated trees and manually annotated 100 sentences for a detailed evaluation. Additionally, we manually construct experimental data for parsing (4) **math problems** (Seo et al., 2015), for which the importance of domain adaptation is previously demonstrated by Joshi et al. (2018). We observe huge additive gains in the performance of the `depccg` parser, developed in Chapter 3, by combining contextualized word embeddings (Peters et al., 2018) and our domain adaptation method: in terms of unlabeled F1 scores, 90.68% to 95.63% on speech conversation, and 88.49% to 95.83% on math problems, respectively.

4.2 Problem Statement

Currently, there are mainly two resources available for CCG parsing: the English CCGbank (Hockenmaier and Steedman, 2007) for news texts, and the Groningen Meaning Bank (Bos et al., 2017) for wider domains, including Aesop's fables. However, when one wants a CCG parser tuned for a specific domain, he or she faces the issue of its high annotation cost:

- The annotation requires linguistic expertise, being able to keep track of semantic composition performed during a derivation.
- An annotated tree must strictly conform to the grammar, e.g., inconsistencies such as combining N and $S\backslash NP$ result in ill-formed trees and hence must be disallowed.

The alternative approach may be first to parse sentences in the target domain into CCG trees, and then manually fix the resulting trees. This may reduce the above costs, but not fully eliminates them, given that the parsed trees can contain lots of errors (this is why domain adaptation is needed).

We relax these assumptions by using *dependency tree*, which is a simpler representation of the syntactic structure, i.e., it lacks information of long-range dependencies and conjunct spans of a coordination structure. However, due to its simplicity and flexibility, it is easier to train an annotator for the grammar, and there exist plenty of accessible dependency-based resources. It is also easier to manually check errors in automatically parsed trees in the target domain. We exploit these merits in this work.

4.3 Dependency-to-CCG Converter

We propose a domain adaptation method based on the automatic generation of a CCGbank out of a dependency treebank in the target domain. This is achieved by our dependency-to-CCG converter, a neural network model consisting of a dependency tree encoder and a CCG tree decoder.

In the encoder, higher-order interactions among dependency edges are modeled with a bidirectional TreeLSTM (Miwa and Bansal, 2016), which is important to facilitate mapping from a dependency tree to a more complex CCG tree. Due to the strict nature

of CCG grammar, we model the output space of CCG trees explicitly;[2] our decoder is inspired by the idea of A* CCG parsing described in Chapter 3, where the most probable valid tree is found using A* parsing. In the following, we describe the details of the proposed converter.

Firstly, we define a probabilistic model of the dependency-to-CCG conversion process. Our finding in the previous chapter is that, the structure of a CCG tree $\boldsymbol{y}$ for sentence $\boldsymbol{x} = (x_1, ..., x_N)$ is almost uniquely determined if a sequence of the pre-terminal CCG categories (supertags) $\boldsymbol{c} = (c_1, ..., c_N)$ and a dependency structure $\boldsymbol{d} = (d_1, ..., d_N)$, where $d_i \in \{0, ..., N\}$ is an index of dependency parent of x_i (0 represents a root node), are provided. Note that the dependency structure $\boldsymbol{d}$ is generally different from an input dependency tree.[3] Let the input dependency tree of sentence $\boldsymbol{x}$ be $\boldsymbol{z} = (\boldsymbol{p}, \boldsymbol{d}', \boldsymbol{\ell})$, where p_i is a part-of-speech tag of x_i, d'_i an index of its dependency parent, and ℓ_i is the label of the corresponding dependency edge, then the conversion process is expressed as follows:[4]

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) = \prod_{i=1}^{N} p_{tag}(c_i|\boldsymbol{x}, \boldsymbol{z}) \prod_{i=1}^{N} p_{dep}(d_i|\boldsymbol{x}, \boldsymbol{z}).$$

Based on this formulation, we model c_i and d_i conditioned on a dependency tree $\boldsymbol{z}$, and search for $\boldsymbol{y}$ that maximizes $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z})$ using A* parsing.

Encoder A bidirectional TreeLSTM consists of two distinct TreeLSTMs (Tai et al., 2015). A *bottom-up* TreeLSTM recursively computes a hidden vector $\boldsymbol{h}_i^{\uparrow}$ for each x_i, from vector representation $\boldsymbol{e}_i$ of the word and hidden vectors of its dependency children $\{\boldsymbol{h}_j^{\uparrow} | d'_j = i\}$. A *top-down* TreeLSTM, in turn, computes $\boldsymbol{h}_i^{\downarrow}$ using $\boldsymbol{e}_i$ and a hidden vector of the dependency parent $\boldsymbol{h}_{d'_i}^{\downarrow}$. In total, a bidirectional TreeLSTM returns concatenations of hidden vectors for all words: $\boldsymbol{h}_i = \boldsymbol{h}_i^{\uparrow} \oplus \boldsymbol{h}_i^{\downarrow}$.

We encode a dependency tree as follows, where $\boldsymbol{e}_v$ denotes the vector representation of variable v, and Ω and $\Xi_{\boldsymbol{d}'}$ are shorthand notations of the series of operations of

[2] The strictness and the large number of categories make it still hard to leave everything to neural networks to learn. We trained constituency-based RSP parser (Joshi et al., 2018) on the English CCGbank by disguising the trees as constituency ones, whose performance could not be evaluated since most of the output trees violated the grammar.

[3] In this work, input dependency tree is based on Universal Dependencies (Nivre et al., 2016), while dependency structure $\boldsymbol{d}$ of a CCG tree is HEADFIRST dependency tree introduced in the previous chapter (Section 3.4). See Section 4.5.1 for the details of the actual implementation.

[4] As the previous Chapter 3, the independence of each c_i and d_i is assumed here.

sequential and tree bidirectional LSTMs, respectively:

$$\boldsymbol{e}_1, ..., \boldsymbol{e}_N = \Omega(\boldsymbol{e}_{p_1} \oplus \boldsymbol{e}_{x_1}, ..., \boldsymbol{e}_{p_N} \oplus \boldsymbol{e}_{x_N}),$$
$$\boldsymbol{h}_1, ..., \boldsymbol{h}_N = \Xi_{\boldsymbol{d}'}(\boldsymbol{e}_1 \oplus \boldsymbol{e}_{\ell_1}, ..., \boldsymbol{e}_N \oplus \boldsymbol{e}_{\ell_N}).$$

Decoder The decoder part adopts the same architecture as one in the `depccg` parser (Section 3.3.1), where $p_{dep|tag}$ probabilities are computed on top of $\{\boldsymbol{h}_i\}_{i=0}^N$, using a *biaffine* layer (Dozat and Manning, 2017) and a bilinear layer, respectively, which are then used in A* parsing to find the most probable CCG tree.

For the completeness, a biaffine layer is used to compute unigram head probabilities p_{dep} as follows:

$$\boldsymbol{r}_i = \psi^{dep}_{child}(\boldsymbol{h}_i), \quad \boldsymbol{r}_j = \psi^{dep}_{head}(\boldsymbol{h}_j),$$
$$s_{i,j} = \boldsymbol{r}_i^\top W \boldsymbol{r}_j + \boldsymbol{w}^\top \boldsymbol{r}_j,$$
$$p_{dep}(d_i = j|\boldsymbol{x}, \boldsymbol{z}) \propto \exp(s_{i,j}),$$

where ψ denotes a multi-layer perceptron. The probabilities p_{tag} are computed by a bilinear transformation of vector encodings x_i and $x_{\hat{d}_i}$, where $\hat{d}_i$ is the most probable dependency head of x_i with respect to p_{dep}: $\hat{d}_i = \arg\max_j p_{dep}(d_i = j|\boldsymbol{x}, \boldsymbol{z})$.

$$\boldsymbol{q}_i = \psi^{tag}_{child}(\boldsymbol{h}_i), \quad \boldsymbol{q}_{\hat{d}_i} = \psi^{tag}_{head}(\boldsymbol{h}_{\hat{d}_i}),$$
$$s_{i,c} = \boldsymbol{q}_i^\top W_c \boldsymbol{q}_{\hat{d}_i} + \boldsymbol{v}_c^\top \boldsymbol{q}_i + \boldsymbol{u}_c^\top \boldsymbol{q}_{\hat{d}_i} + b_c,$$
$$p_{tag}(c_i = c|\boldsymbol{x}, \boldsymbol{z}) \propto \exp(s_{i,c}).$$

A* Parsing Since the probability $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z})$ of a CCG tree $\boldsymbol{y}$ has the exactly same structure as the probabilistic model for parsing proposed in the previous chapter (Section 3.3.1), we can use A* parsing to obtain the most probable tree. Here, we perform A* parsing as described in the section, and adopt an admissible heuristic by taking the sum of the max $p_{tag|dep}$ probabilities outside a subtree. The advantage of employing an A* parsing-based decoder is not limited to the optimality guarantee of the decoded tree; it enables constrained decoding, which is described next.

4.4 Constrained Decoding

While our method is a fully automated treebank generation method, there are often cases where we want to control the form of output trees by using external language

resources. For example, when generating a CCGbank for biomedical domain, it will be convenient if a disease dictionary is utilized to ensure that a complex disease name in a text is always assigned the category NP. In our decoder based on A* parsing, it is possible to perform such a controlled generation of a CCG tree by imposing *constraints* on the space of trees.

A constraint is a triplet (c, i, j) representing a constituent of category c spanning over words $x_i, ..., x_j$. The constrained decoding is achieved by refusing to add a subtree (denoted as (c', k, l), likewise, with its category and span) to the priority queue when it meets one of the following conditions:

- The spans overlap: $i < k \leq j < l$ or $k < i \leq l < j$.
- The spans are identical ($i = k$ and $j = l$), while the categories are different ($c \neq c'$) and no category c'' exists such that $c' \Rightarrow c''$ is a valid unary rule.

The last condition on unary rule is necessary to prevent structures such as $(NP\ (N\ \text{dog}))$ from being accidentally discarded, when using a constraint to make a noun phrase to be NP. A set of multiple constraints are imposed by checking the above conditions for each of the constraints when adding a new item to the priority queue. When one wants to constrain a pre-terminal category to be c, that is achieved by manipulating p_{tag}: $p_{tag}(c|\boldsymbol{x}, \boldsymbol{z}) = 1$ and for all categories $c' \neq c$, $p_{tag}(c'|\boldsymbol{x}, \boldsymbol{z}) = 0$.

4.5 Experiments

4.5.1 Experimental Settings

We evaluate our method in terms of performance gain obtained by fine-tuning the `depccg` parser (Chapter 3), on a variety of CCGbanks obtained by converting existing dependency resources using the method.

As described in the chapter, in the plain `depccg`, the word representation $\boldsymbol{e}_{x_i}$ is a concatenation of GloVe[5] vectors and vector representations of affixes. The parser is trained on both the English CCGbank (Hockenmaier and Steedman, 2007) and the tri-training dataset (Section 3.5). In this work, on top of that, we include as a baseline a setting where the affix vectors are replaced by contextualized word representation

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

(ELMo (Peters et al., 2018); $\boldsymbol{e}_{x_i} = \boldsymbol{x}_{x_i}^{GloVe} \oplus \boldsymbol{x}_{x_i}^{ELMo}$),[6] which we find marks the current best scores in the English CCGbank parsing (Table 4.1).

The evaluation is based on the standard evaluation metric, where the number of correctly predicted predicate argument relations is calculated (Section 2.4), where *labeled* metrics take into account the category through which the dependency is constructed, while *unlabeled* ones do not.

Implementation Details The input word representations to the converter are the concatenation of GloVe and ELMo representations. Each of $\boldsymbol{e}_{p_i}$ and $\boldsymbol{e}_{\ell_i}$ is randomly initialized 50-dimensional vectors, and the two-layer sequential LSTMs Ω outputs 300 dimensional vectors, as well as bidirectional TreeLSTM $\Xi_{\boldsymbol{d}'}$, whose outputs are then fed into 1-layer 100-dimensional MLPs with ELU non-linearity (Clevert et al., 2016). The training is done by minimizing the sum of negative log likelihood of $p_{tag|dep}$ using the Adam optimizer (with $\beta_1 = \beta_2 = 0.9$), on a dataset detailed below.

Data Processing In this work, the input tree to the converter follows Universal Dependencies (UD) v1 (Nivre et al., 2016). Constituency-based treebanks are converted using the Stanford Converter[7] to obtain UD trees. The output dependency structure $\boldsymbol{d}$ follows HEADFIRST dependency tree (Section 3.4), where a dependency arc is always from left to right. These choices are made by the following consideration: (1) since available UD-based resources are abundant and increasing, it is meaningful to evaluate the effectiveness of our method in terms of them considering future applications, and (2) modeling HEADFIRST dependencies in a CCG tree is found the most effective in the previous chapter. The conversion model is trained to map UD trees in the Wall Street Journal (WSJ) portion 2-21 of the Penn Treebank (Marcus et al., 1993) to its corresponding CCG trees in the English CCGbank (Hockenmaier and Steedman, 2007).

Fine-tuning the CCG Parser In each of the following domain adaptation experiments, newly obtained CCGbanks are used to fine-tune the parameters of the baseline parser described above, by re-training it on the mixture of labeled examples from the new target-domain CCGbank, the English CCGbank, and the tri-training dataset.

[6] We used the "original" ELMo model, with 1,024-dimensional word vector outputs (https://allennlp.org/elmo).

[7] https://nlp.stanford.edu/software/stanford-dependencies.shtml. We used the version 3.9.1.

Method	UF1	LF1
depccg	94.0	88.8
+ ELMo	94.98	90.51
Converter	96.48	92.68

Table 4.1: The performance of baseline CCG parsers and the proposed converter on WSJ23, where UF1 and LF1 represents unlabeled and labeled F1, respectively.

Relation	Parser	Converter	#
(a) *PPs attaching to NP / VP*			
$(NP\backslash\underline{\mathbf{NP}})/NP$	90.62	97.46	2,561
$(S\backslash NP)\backslash(\mathbf{S}\backslash\underline{\mathbf{NP}}))/NP$	81.15	88.63	1,074
(b) *Subject / object relative clauses*			
$(NP\backslash\underline{\mathbf{NP}})/(S_{dcl}\backslash NP)$	93.44	98.71	307
$(NP\backslash\underline{\mathbf{NP}})/(S_{dcl}/NP)$	90.48	93.02	20

Table 4.2: Per-relation F1 scores of the proposed converter and depccg + ELMo (Parser). "#" column shows the number of occurrence of the phenomenon.

4.5.2 Evaluating Converter's Performance

First, we examine whether the trained converter can produce high-quality CCG trees, by applying it to dependency trees in the test portion (WSJ23) of Penn Treebank and then calculating the standard evaluation metrics between the resulting trees and the corresponding gold trees (Table 4.1). This can be regarded as evaluating the upper bound of the conversion quality, since the evaluated data comes from the same domain as the converter's training data. Our converter shows much higher scores compared to the current best-performing depccg combined with ELMo (1.5% and 2.17% up in unlabeled/labeled F1 scores), suggesting that, using the proposed converter, we can obtain CCGbanks of high quality.

Inspecting the details, the improvement is observed across the board (Table 4.2); the converter precisely handles PP-attachment (4.2a), notoriously hard parsing problem, by utilizing input's pobj dependency edges, as well as relative clauses (4.2b), one of well-known sources of long-range dependencies, for which the converter has to learn from the non-local combinations of edges, their labels and part-of-speech tags

Method	P	R	F1
C&C	77.8	71.4	74.5
EasySRL	81.8	82.6	82.2
depccg	83.11	82.63	82.87
+ ELMo	85.87	85.34	85.61
+ GENIA1000	85.45	84.49	84.97
+ Proposed	**86.90**	**86.14**	**86.52**

Table 4.3: Results on the biomedical domain dataset (Section 4.5.3). P and R represent precision and recall, respectively. The scores of C&C and EasySRL fine-tuned on the GENIA1000 is included for comparison (excerpted from (Lewis et al., 2016)).

Method	P	R	F1
C&C	-	-	86.8
EasySRL	88.2	87.9	88.0
depccg	90.42	**90.15**	**90.29**
+ ELMo	**90.55**	89.86	90.21
+ Proposed	90.27	89.97	90.12

Table 4.4: Results on question sentences (Section 4.5.3). All of baseline C&C, EasySRL and depccg parsers are retrained on Questions data.

surrounding the phenomenon.

4.5.3 Biomedical Domain and Questions

Previous work (Rimell and Clark, 2008) provides CCG parsing benchmark datasets in biomedical texts and question sentences, each representing two contrasting challenges for a newswire-trained parser, i.e., a large amount of out-of-vocabulary words (biomedical texts), and rare or even unseen grammatical constructions (questions).

Since the work also provides small training datasets for each domain, we utilize them as well: GENIA1000 with 1,000 sentences and Questions with 1,328 sentences, both annotated with pre-terminal CCG categories. Since pre-terminal categories are not sufficient to train depccg, we automatically annotate HEADFIRST dependencies

using RBGParser (Lei et al., 2014), trained to produce this type of trees (We use the same parser trained in the Section 3.5).

Following the previous work, the evaluation is based on the Stanford grammatical relations (GR; (Marneffe et al., 2006)), a deep syntactic representation that can be recovered from a CCG tree.[8]

Biomedical Domain By converting the Genia corpus (Tateisi et al., 2005), we obtain a new CCGbank of 4,432 sentences from biomedical papers annotated with CCG trees. During the process, we have successfully assigned the category NP to all the occurrences of complex biomedical terms by imposing constraints (Section 4.4) that NP spans in the original corpus be assigned the category NP in the resulting CCG trees as well.

Table 4.3 shows the results of the parsing experiment, where the scores of previous work (C&C and EasySRL; Section 2.3) are included for reference. The plain depccg already achieves higher scores than these methods, and boosts when combined with ELMo (improvement of 2.73 points in terms of F1). Fine-tuning the parser on GENIA1000 results in a mixed result, with slightly lower scores. This is presumably because the automatically annotated HEADFIRST dependencies are not accurate. Finally, by fine-tuning on the Genia CCGbank, we observe another improvement, resulting in the highest 86.52 F1 score.

Questions In this experiment, we obtain a CCG version of the QuestionBank (Judge et al., 2006), consisting of 3,622 question sentences, excluding ones contained in the evaluation data.

Table 4.4 compares the performance of depccg fine-tuned on the QuestionBank, along with other baselines. Contrary to our expectation, the plain depccg retrained on Questions data performs the best, with neither ELMo nor the proposed method taking any effect. We hypothesize that, since the evaluation set contains sentences with similar constructions, the contributions of the latter two methods are less observable on top of Questions data. Inspection of the output trees reveals that this is actually the case; the majority of differences among parser's configurations are irrelevant to question constructions, suggesting that the models capture well the syntax of question in the data.[9]

[8] We used their public script (https://www.cl.cam.ac.uk/~sc609/candc-1.00.html).

[9] Due to many-to-many nature of mapping to GRs, the evaluation set contains relations not recov-

4.5.4 Speech Conversation

Setup We apply the proposed method to a new domain, transcription texts of speech conversation, with new applications of CCG parsing in mind. We create the CCG version of the Switchboard corpus (Godfrey et al., 1992), by which, as far as we are aware of, we conduct the first CCG parsing experiments on speech conversation.[10] We obtain a new CCGbank of 59,029/3,799/7,681 sentences for each of the train/test /development set, where the data split follows prior work on dependency parsing on this dataset (Honnibal and Johnson, 2014).

In the conversion, we have to handle one of the characteristics of speech transcription texts, *disfluencies*. In real application, it is ideal to remove disfluencies such as interjection and repairs (e.g., *I want a flight to Boston um to Denver*), prior to performing CCG-based semantic composition. Since this corpus contains a layer of annotation that labels their occurrences, we perform constrained decoding to mark the gold disfluencies in a tree with a dummy category X, which can combine with any category from both sides (i.e., for all category C, $C\ X \Rightarrow C$ and $X\ C \Rightarrow C$ are allowed). In this work, we perform parsing experiments on texts that are clean of disfluencies, by removing X-marked words from sentences (i.e., a pipeline system setting with an oracle disfluency detection preprocessor).[11]

Another issue in conducting experiments on this dataset is evaluation. Since there exists no evaluation protocol for CCG parsing on speech texts, we evaluate the quality of output trees by two procedures; in the first experiment, we parse the entire test set, and convert them to constituency trees using a method by Kummerfeld et al. (2012).[12] We report labeled bracket F1 scores between the resulting trees and the gold trees in the true Switchboard corpus, using the EVALB script.[13] However, the reported scores suffer from the compound effect of failures in CCG parsing as well as ones occurred in the conversion to the constituency trees. To evaluate the parsing performance in detail, the first author manually annotated a subset of randomly sampled 100 sentences

erable from the gold supertags using the provided script; for example, we find that from the annotated supertags of sentence *How many battles did she win ?*, the (`amod battle many`) relation is obtained instead of the gold `det` relation. This implies one of the difficulties to obtain further improvement on this set.

[10] Since the annotated part-of-speech tags are noisy, we automatically reannotate them using the `core_web_sm` model of spaCy (`https://spacy.io/`), version 2.0.16.

[11] We regard developing joint disfluency detection and syntactic parsing method based on CCG as future work.

[12] `https://github.com/jkkummerfeld/berkeley-ccg2pst`

[13] `https://nlp.cs.nyu.edu/evalb/`

a.	*we should cause it does help*
b.	*the only problem i see with term limitations is that i think that the bureaucracy in our government as is with most governments is just so complex that there is a learning curve and that you ca n't just send someone off to washington and expect his first day to be an effective congress precision*

Table 4.5: Example sentences from the manually annotated subset of Switchboard test set.

Error type	#
PP-attachment	3
Adverbs attaching wrong place	11
Predicate-argument	5
Imperative	2
Informal functional words	2
Others	11

Table 4.6: Error types observed in the manually annotated Switchboard subset data.

from the test set. Sentences with less than four words are not contained, to exclude short phrases such as nodding. Using this test set, we report the standard CCG parsing metrics. Sentences from this domain exhibit other challenging aspects (Table 4.5), such as less formal expressions (e.g., use of *cause* instead of *because*) (4.5a), and lengthy sentences with many embedded phrases (4.5b).[14]

Results On the whole test set, `depccg` shows consistent improvements when combined with ELMo and the proposed method, in the constituency-based metrics (**Whole** columns in Table 4.7). Though the entire scores are relatively lower, the result suggests that the proposed method is effective to this domain on the whole. By directly

[14]Following Honnibal and Johnson (2014), sentences in this data are fully lower-cased and contain no punctuation.

Method	Whole			Subset	
	P	R	F1	UF1	LF1
depccg	74.73	73.91	74.32	90.68	82.46
+ ELMo	75.76	76.62	76.19	93.23	86.46
+ Proposed	**78.03**	**77.06**	**77.54**	**95.63**	**92.65**

Table 4.7: Results on speech conversation texts (Section 4.5.4), on the whole test set and the manually annotated subset.

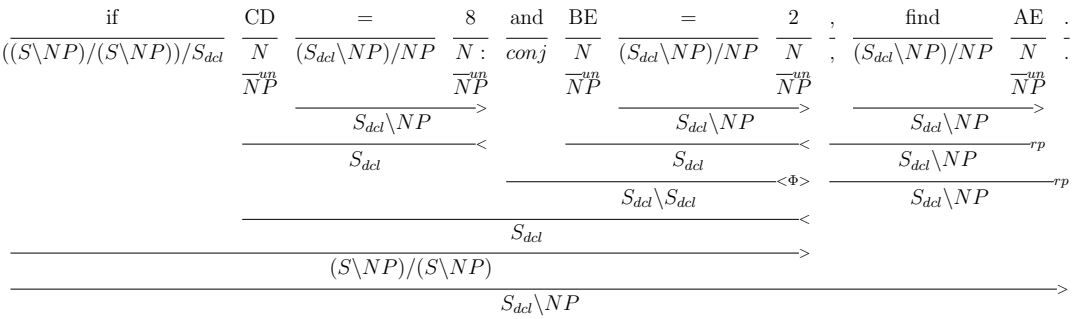


Figure 4.2: Parse output by the re-trained parser for sentence *if CD = 8 and BE = 2, find AE.* from math problems.

evaluating the parser's performance in terms of predicate argument relations (**Subset** columns), we observe that it actually recovers the most of the dependencies, with the fine-tuned depccg achieving as high as 95.63% unlabeled F1 score.

We further investigate error cases of the fine-tuned depccg in the subset dataset (Table 4.6). The tendency of error types is in accordance with other domains, with frequent errors in PP-attachment and predicate-argument structure, and seemingly more cases of attachment errors of adverbial phrases (11 cases), which occur in lengthy sentences such as in Table 4.5b. Other types of error are failures to recognize that the sentence is in imperative form (2 cases), and ones in handling informal functional words such as *cause* (Table 4.5a). We conclude that the performance on this domain is as high as it is usable in application. Since the remaining errors are general ones, they will be solved by improving general parsing techniques.

Method	UF1	LF1
depccg	88.49	66.15
+ ELMo	89.32	70.74
+ Proposed	**95.83**	**80.53**

Table 4.8: Results on math problems (Section 4.5.5).

4.5.5 Math Problems

Setup Finally, we conduct another experiment on parsing math problems. Following previous work of constituency parsing on math problem (Joshi et al., 2018), we use the same train/test sets by Seo et al. (2015), consisting of 63/62 sentences respectively, and see if a CCG parser can be adapted with the small training samples. Again, the first author annotated both train/test sets, dependency trees on the train set, and CCG trees on the test set, respectively. In the annotation, we follow the manuals of the English CCGbank and the UD. We regard as an important future work extending the annotation to include fine-grained feature values in categories, e.g., marking a distinction between integers and real numbers (Matsuzaki et al., 2017). Figure 4.2 shows an example CCG tree from this domain, successfully parsed by fine-tuned depccg.

Results Table 4.8 shows the F1 scores of depccg in the respective settings. Remarkably, we observe huge additive performance improvement. While, in terms of labeled F1, ELMo contributes about 4 points on top of the plain depccg, adding the new training set (converted from dependency trees) improves more than 10 points.[15] Examining the resulting trees, we observe that the huge gain is primarily involved with expressions unique to math. Figure 4.2 is one of such cases, which the plain depccg falsely analyzes as one huge NP phrase. However, after fine-tuning, it successfully produces the correct "If S_1 and S_2, S_3" structure, recognizing that the equal sign is a predicate.

[15]Note that, while in the experiment on this dataset in the previous constituency parsing work (Joshi et al., 2018), they evaluate on partially annotated (unlabeled) trees, we perform the "full" CCG parsing evaluation, employing the standard evaluation metrics. Given that, the improvement is even more significant.

4.6 Summary

In this work, we have proposed a domain adaptation method for CCG parsing, based on the automatic generation of new CCG treebanks from dependency resources. We have conducted experiments to verify the effectiveness of the proposed method on diverse domains: on top of existing benchmarks on biomedical texts and question sentences, we newly conduct parsing experiments on speech conversation and math problems. Remarkably, when applied to our domain adaptation method, the improvements in the latter two domains are significant, with the achievement of more than 5 points in the unlabeled metric.

Finally, the technique developed in this chapter is actually not limited to domain adaptation. The huge gains in various domains suggest effective applications of the method in other settings; for example, the method may be used to increase the labeled samples in the news-wire domain, which may in turn makes it possible to construct an even performant parser on CCGBank parsing benchmarks (Section 3.6). Evaluating this aspect of our method is an important direction in future work.

Chapter 5

Axiom Injection for Logic-based Inference System

5.1 Introduction

RTE is a challenging NLP task where the objective is to judge whether a hypothesis H logically follows from premise(s) P. Advances in RTE have positive implications in other areas such as information retrieval, question answering and reading comprehension. Various approaches have been proposed to the RTE problem in the literature. Some methods are based on deep neural networks (Rocktäschel et al., 2016; Chen et al., 2018; Nie and Bansal, 2017), where a classifier is trained to predict the relation using H and P encoded in a high-dimensional space. Other methods are purely symbolic (Bos et al., 2004; Mineshima et al., 2015; Abzianidze, 2015), where logical formulas that represent H and P are constructed and used in a formal proof system. In this chapter, we adopt a strategy based on logic, encouraged by the high-performance that these systems achieve in linguistically complex datasets (Mineshima et al., 2015; Abzianidze, 2015), which contain a variety of semantic phenomena that are still challenging for the current neural models (Wang et al., 2018).

Contrary to the end-to-end machine learning approaches, a logic-based system must explicitly maintain lexical knowledge necessary for inference. A critical challenge here is to deal with such knowledge in an efficient and scalable way. A promising approach in past work is on-demand axiom injection (*abduction mechanism*; Martínez-Gómez et al. (2017)), which allows one to construct knowledge between words in P and H as lexical axioms, and feed them to a logic prover when necessary. Combined with `ccg2lambda` (Section 2.5), their method demonstrates that injecting lexical knowl-

edge from WordNet (Miller, 1995) and VerbOcean (Chklovski and Pantel, 2004) significantly improves the performance.

Although their method provides a basis for handling external knowledge with a logic-based RTE system, there still remains a practical issue in terms of scalability and efficiency. Their abduction mechanism generates relevant axioms on-the-fly for a present P and H pair, but this means we have to maintain a large knowledge base inside the system. This is costly in terms of memory, and it also leads to slower search due to a huge search space during inference. WordNet already contains relations among more than 150,000 words, and in practice, we want to increase the coverage of external knowledge more by adding different kinds of database such as Freebase. To achieve such a scalable RTE system, we need a more efficient way to preserve database knowledge.

In this chapter, we present an approach to axiom injection, which, by not holding databases explicitly, allows handling of massive amount of knowledge without losing efficiency. Our work is built on Knowledge Base Completion (KBC), which recently has seen a remarkable advancement in the machine learning community. Although KBC models and logic-based approaches to RTE have been studied separately so far, we show that they can be combined to improve the overall performance of RTE systems. Specifically, we replace the search of necessary knowledge on the database with a judgment of whether the triplet (s, r, o) is a fact or not in an n-dimensional vector space that encodes entities s and o and relation r. For each triplet, this computation is efficient and can be done in $O(n)$ complexity. To this end we construct a new dataset from WordNet for training a KBC model that is suitable for RTE. We then show that this approach allows adding new knowledge from VerbOcean without losing efficiency.

Throughout this chapter, we will focus on inferences that require lexical knowledge such as synonym and antonym and its interaction with the logical and linguistic structure of a sentence, distinguishing them from common sense reasoning (e.g., *John jumped into the lake* entails *John is wet*) and inferences based on world knowledge (e.g., *Chris lives in Hawaii* entails *Chris lives in USA*). For evaluation, we use the SICK (Sentences Involving Compositional Knowledge) dataset (Marelli et al., 2014), which focuses on lexical inferences combined with linguistic phenomena.[1]

Another advantage of our approach is that we can complement the missing lexical

[1]Large-scale datasets for training neural natural language inference models such as SNLI (Williams et al., 2018) and MultiNLI (Williams et al., 2018) are not constrained to focus on lexical and linguistic aspects of inferences, which can produce confounding factors in analysis, hence are not suitable for our purposes.

knowledge in existing knowledge bases as latent knowledge. The previous method is limited in that it can only extract relations that are directly connected or reachable by devising some relation path (e.g. transitive closure for hypernym relation); however, there are also lexical relations that are not explicitly available and hence latent in the networks. To carefully evaluate this aspect of our method, we manually create a small new RTE dataset, where each example requires complex lexical reasoning, and find that our system is able to find and utilize such latent knowledge that cannot be reached by the existing approach.

Our final system achieves a competitive RTE performance with Martínez-Gómez et al. (2017)'s one, while keeping the processing speed of the baseline method that does not use any external resources. The last key technique for this efficiency is a new `abduction` tactic, a plugin for a theorem prover Coq (The Coq Development Team, 2017). One bottleneck of Martínez-Gómez et al. (2017)'s approach is that in their system Coq must be rerun each time new axioms are added. To remedy this overhead we develop `abduction` tactic that enables searching knowledge bases and executing a KBC scoring function during running Coq.

Our contributions are summarized as follow:

- We propose to combine KBC with a logic-based RTE system for efficient and scalable reasoning.
- We develop an efficient abduction plugin for Coq, which we make publicly available (Section 1.4).
- We show that our techniques achieve a competitive score to the existing abduction technique while maintaining the efficiency of the baseline with no knowledge bases.
- We construct a set of lexically challenging RTE problems and conduct extensive experiments to evaluate the latent knowledge our KBC model has learned. We demonstrate many examples of those knowledge that are not available for the previous method.

5.2 Related work

5.2.1 Logic-based RTE systems

Earlier work on logic-based approaches to RTE exploited off-the-shelf first-order reasoning tools (theorem provers and model-builders) for the inference component (Bos and Markert, 2005). Such a logic-based system tends to have high precision and low recall for RTE tasks, suffering from the lack of an efficient method to integrate external knowledge in the inference system.

Meanwhile, researchers in Natural Logic (van Benthem, 2008) have observed that the iteration depth of logical operators such as negations and quantifiers in natural languages is limited and, accordingly, have developed a variety of proof systems such as monotonicity calculus (Icard and Moss, 2014) adapted for natural languages that use small parts of first-order logic.

The idea of natural logic has recently led to a renewed interest in symbolic approaches to modeling natural language inference in the context of NLP (MacCartney and Manning, 2008). In particular, theorem provers designed for natural languages have been recently developed, where a proof system such as analytic tableau (Abzianidze, 2015) and natural deduction (Mineshima et al., 2015) is used in combination with wide-coverage parsers. These systems allow a controlled use of higher-order logic, following the tradition of formal semantics (Montague, 1974), and thereby have achieved efficient reasoning for logically complex RTE problems such as those in the FraCaS test suite (Cooper et al., 1994). However, it has remained unclear how one can add robust external knowledge to such logic-based RTE systems without loss of efficiency of reasoning. The aim of this chapter is to address this issue.

We use Coq, an interactive proof assistant based on the Calculus of Inductive Constructions (CiC), in order to implement an RTE system with our method of axiom insertion. Although Coq is known as an interactive proof assistant, it has a powerful proof automation facility where one can introduce user-defined proof strategies called *tactics*. The work closest to our approach in this respect is a system based on Modern Type Theory (Chatzikyriakidis and Luo, 2014), which uses Coq as an automated theorem prover for natural language inferences. It was shown that the system achieved high accuracy on the FraCaS test suite (Bernardy and Chatzikyriakidis, 2017). However, the system relies on a hand-written grammar, suffering from scalability issues. Additionally, this work did not evaluate the efficiency of theorem proving for RTE, nor address the issue of how to extend their RTE system with robust external knowledge.

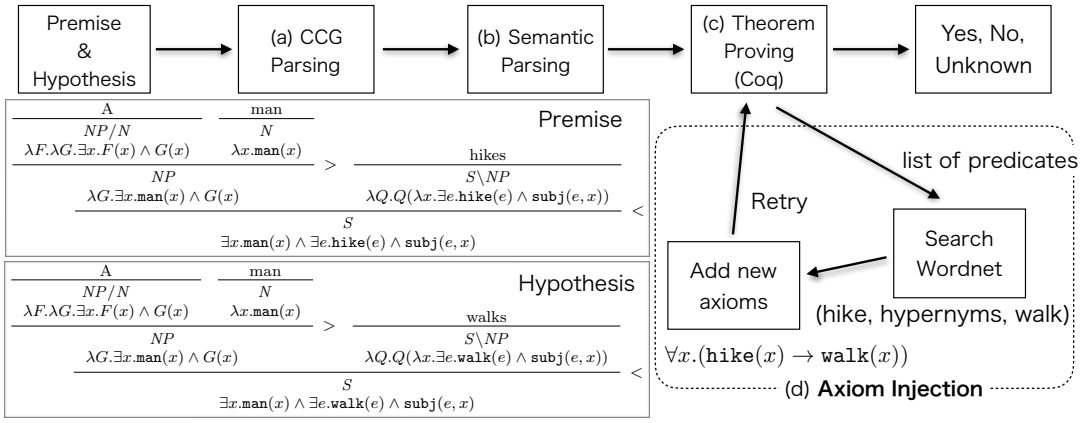


Figure 5.1: A pipeline of `ccg2lambda`. It firstly applies CCG parser to premise (P) and hypothesis (H) sentences (a), and then convert them to logical formulas (b). It tries to prove if entailment (contradiction) can be established by applying Coq to the theorem $P \rightarrow H$ ($P \rightarrow \neg H$) (c). If the proving fails, it tries axiom injection (d).

Generally speaking, it seems fair to say that the issue of efficiency of logic-based reasoning systems with a large database has been under-appreciated in the literature on RTE. To fill this gap, we investigate how our logic-based approach to RTE, combined with machine learning-based Knowledge Base Completion, can contribute to robust and efficient natural language reasoning.

5.2.2 Knowledge Base Completion (KBC)

Several KBC models have been proposed in the literature (Bordes et al., 2013; Trouillon et al., 2016; Dettmers et al., 2017). Among them we use ComplEx (Trouillon et al., 2016), which models triplet (s, r, o) for entities $s, o \in \mathcal{E}$ and relation $r \in \mathcal{R}$ in an n-dimensional complex vector space[2]:

$$\psi_r(s, o) = \sigma(Re(\langle \boldsymbol{e}_s, \boldsymbol{e}_r, \overline{\boldsymbol{e}_o} \rangle)), \tag{5.1}$$

where $\boldsymbol{e}_s, \boldsymbol{e}_r, \boldsymbol{e}_o \in \mathbb{C}^n$, $\langle \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \rangle = \sum_i x_i y_i z_i$, and σ is the sigmoid function. Since Eq. 5.1 consists of one dot-product among three vectors, its computational complexity is $O(n)$. The training is done by minimizing the logistic loss:

[2] $\mathbb{C}$ denotes the set of complex numbers. For $x \in \mathbb{C}$, $Re(x)$ denotes its real part and $\overline{x}$ its complex conjugate.

$$\sum_{((s,r,o),t)\in\mathcal{D}} t \log \psi_r(s,o) + (1-t)\log(1-\psi_r(s,o)), \tag{5.2}$$

where $\mathcal{D}$ is the training data. We have $t = 1$ if the triplet is a fact and $t = 0$ if not (negative example). While negative examples are usually collected by negative sampling, 1-N scoring has been proposed to accelerate training (Dettmers et al., 2017). In 1-N scoring, unlike other KBC models that take an entity pair and a relation as a triplet (s, r, o) and score it (1-1 scoring), one takes one (s, r) pair and scores it against all entities $o \in \mathcal{E}$ simultaneously. It is reported that this brings over 300 times faster computation of the loss for their convolution-based model. This method is applicable to other models including ComplEx and scales to large knowledge bases. We employ this technique in our experiments.

5.3 System overview

We build our system on `ccg2lambda`. Though we have briefly introduced the system Chapter 2.5, here, we describe the system in detail for the purpose of this chapter. Figure 2.4 (repeated as Figure 5.1) shows the pipeline of the system.

Note that although we use a particular RTE system to test our hypotheses, other logic-based RTE systems can also benefit from our KBC-based method. In so far as a lexical relation is modeled as a triplet, it could be adapted to their inference modules; for instance, if r is a hypernym relation, a triplet (s, r, o) is mapped to $s \sqsubseteq o$, where $\sqsubseteq$ is a containment relation in Natural Logic (MacCartney and Manning, 2008) or a subtyping relation in Modern Type Theory (Chatzikyriakidis and Luo, 2014), rather than to $\forall x.(s(x) \to o(x))$ as in the standard logic we use in this chapter.

5.3.1 CCG and semantic parsing

The system processes premise(s) (P) and a hypothesis (H) using CCG parsers. CCG is a lexicalized grammar that provides syntactic structures transparent to semantic representations (Figure 5.1a). In CCG, each lexical item is assigned a pair (C, M) of the syntactic category C and a meaning representation M encoded as a λ-term; for instance, "man" is assigned $(N, \lambda x.\mathtt{man}(x))$. The parses (called *derivation trees*) are

converted into logical formulas by composing λ-terms assigned to each terminal word in accordance with combinatory rules (Figure 5.1b).

For the assignment of λ-terms, we use a template-based procedure, where closed-class words (logical or functional expressions) are mapped to their specific meaning representations and other words to schematic meaning representations based on CCG categories. In this work, we adopt a semantic template based on Neo-Davidsonian Event Semantics (Parsons, 1990), where a sentence is mapped to a formula involving quantification over events and a verb is analyzed as a 1-place predicate over events using auxiliary predicates for semantic roles such as `subj` (see the formulas in Figure 5.1b). One of the main attractions of this approach is that it facilitates the simple representation of lexical relations for nouns and verbs, since both can be uniformly analyzed as 1-place predicates (see the axioms in Table 5.1).

5.3.2 Theorem proving

The system uses automated theorem proving in Coq (Figure 5.1c) to judge whether entailment ($P \rightarrow H$) or contradiction ($P \rightarrow \neg H$) holds between the premise and the hypothesis. It implements a specialized prover for higher-order features in natural language, which is combined with Coq's build-in efficient first-order inference mechanism. Coq has a language called Ltac for user-defined automated tactics (Delahaye, 2000). The additional axioms and tactics specialized for natural language constructions are written in Ltac. We run Coq in a fully automated way, by feeding to its interactive mode a set of predefined tactics combined with user-defined proof-search tactics.

5.3.3 Axiom insertion (abduction)

Previous work (Martínez-Gómez et al., 2017) extends `ccg2lambda` with an axiom injection mechanism using databases such as WordNet (Miller, 1995). When a proof of $T \rightarrow H$ or $T \rightarrow \neg H$ fails, it searches these databases for lexical relations that can be used to complete the theorem at issue. It then restarts a proof search after declaring the lexical relations as axioms (Figure 5.1d). This mechanism of on-demand insertion of axioms is called *abduction*.

A problem here is that this abduction mechanism slows down the overall processing speed. This is mainly due to the following reasons: (1) searches for some sort of

Relation r	Generated Axiom	Example
synonym, hypernym, derivationally-related	$\forall x.s(x) \to o(x)$	$(\texttt{make}, \texttt{synonym}, \texttt{build}) \rightsquigarrow \forall e.\texttt{make}(e) \to \texttt{build}(e)$
antonym	$\forall x.s(x) \to \neg o(x)$	$(\texttt{parent}, r, \texttt{child}) \rightsquigarrow \forall x.\texttt{parent}(x) \to \neg\texttt{child}(x)$
hyponym	$\forall x.o(x) \to s(x)$	$(\texttt{talk}, r, \texttt{advise}) \rightsquigarrow \forall e.\texttt{advise}(e) \to \texttt{talk}(e)$

Table 5.1: Triplet (s, r, o) and axioms generated in terms of r. The type of an argument is determined in the semantic parsing part of `ccg2lambda`. While we use unary predicates (Neo-Davidsonian semantics), it can be generalized to any arity.

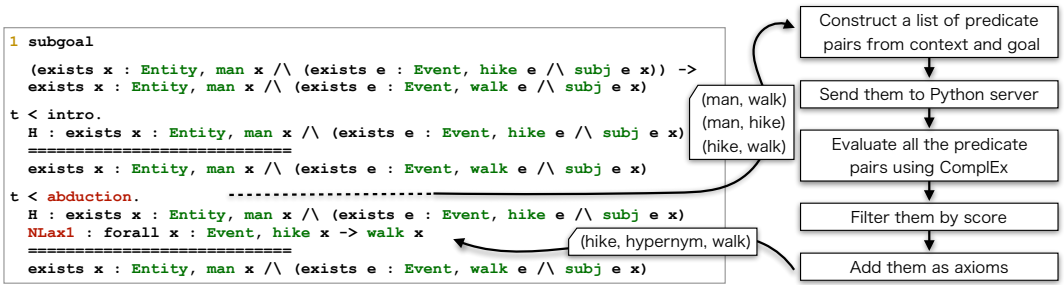


Figure 5.2: Running example of `abduction` tactic in a Coq session proving "*A man hikes*" $\to$ "*A man walks*". When the tactic is executed, it interacts with a ComplEx model on a different process and injects high scoring triplets as axioms.

relations incur multi-step inference over triplets (e.g. transitive closure of hypernym relation); (2) the theorem proving must be done all over again to run an external program that searches the knowledge bases. In this work, to solve (1), we propose an efficient $O(n)$ abduction mechanism based on KBC (Section 5.4.1). For (2), we develop `abduction` tactic, that enables adding new lexical axioms without quitting a Coq process (Section 5.4.2).

5.4 Proposed method

Our abduction mechanism adds new axioms whenever the prover stops due to the lack of lexical knowledge. Specifically, the system collects pairs of predicates from a current proof state and evaluates the pairs for every relation $r \in \mathcal{R} = \{\texttt{synonym}, \texttt{hypernym}, \texttt{antonym}, \texttt{hyponym}, \texttt{derivationally-related}\}$ with ComplEx (Eq. 5.1). It then declares as axioms logical formulas converted from the triplets

whose scores are above a predefined threshold θ. See Table 5.1 for the conversion rules. We describe the construction of a training data for ComplEx in Section 5.4.1 and `abduction` tactic that performs the axiom injection in Section 5.4.2.

5.4.1 Data creation

Although there already exist benchmark datasets for WordNet completion (e.g., WN18 (Bordes et al., 2013)), we construct our own dataset. We find two problems on the existing datasets considering its application to RTE tasks. One is a gap between the entities and relations appearing in those benchmark datasets and those needed to solve RTE datasets. For example the knowledge on disease names are not necessary for the present dataset.

Another, more critical issue is that in WordNet many relations including hypernym and hyponym are defined among *synsets*, i.e., sets of synonymous lemmas, and the existing datasets also define a graph on them. This is problematic in practice for `ccg2lambda`, which normalizes a word's surface form into its lemma when obtaining a logical formula. A possible option to mitigate this discrepancy is to normalize each word into synset by adding word-sense disambiguation (WSD) step in `ccg2lambda`. Another option is to construct a new dataset in which each relation is defined among lemmas. We choose the latter for two reasons: (1) existing WSD systems do not perform well and cause error propagation; and (2) a dataset defined among lemmas eases the data augmentation using other resources. For example, VerbOcean defines relations between lemmas so it is straightforward to augment the training data with it, as we show later.

We use different strategies to extract relations for each relation $r \in \mathcal{R}$ from WordNet as follows.[3]

synonym Since synonym relation is not defined in WordNet, we find them from other relations. We regard two synsets s_1 and s_2 as synonym if they are in `also_sees`, `verb_groups`, or `similar_tos` relation, or if they share some lemma $l \in s_1 \cap s_2$. Then, we take a Cartesian product of s_1 and s_2, that is, $(l_1, \texttt{synonym}, l_2)$ for all $l_1 \in s_1$ and $l_2 \in s_2$, and add them to the dataset.

hypernym and hyponym These relations are defined among synsets in WordNet. As in `synonym`, for `hypernym` we take a Cartesian product of s_1 and s_2, when

[3]For simplicity, we use the set theoretical notation: $l \in s$ denotes lemma l is an element of synset s.

they are in hypernym relation. We also collect the triplets obtained from its transitive closure, since `hypernym` is a transitive relation.[4] We process hyponym relations in the same way.

antonym and derivationally-related Since antonym and derivationally-related relations are defined among lemmas in WordNet, we simply collect them.

Since the constructed dataset contains many entities that will not be used in the existing RTE datasets, we strip off triplets (s, r, o) if s or o is not found in our predefined lemma list, consisting of all lemmas of words appearing in the training part of SICK (Marelli et al., 2014) and SNLI (Williams et al., 2018), as well as the pre-trained GloVe word vectors.[5] The resulting dataset contains 1,656,021 triplets and the number of the entities is 41,577. We spare random 10,000 triplets for development and use the rest for training.

VerbOcean is a semi-automatically constructed repository of semantic relations among verbs. Since it defines a triplet among lemmas, we simply map the relations in the dataset to $\mathcal{R}$ after filtering triplets using the lemma list above. Concretely, in the experiments we use `similar` relations, which consists of 17,694 triplets.

5.4.2 Axiom injection with `abduction` tactic

As mentioned earlier, the abduction method in the previous work needs to rerun Coq when adding new axioms. Now we introduce our `abduction` tactic that enables adding new axioms on the fly without quitting the program.

Figure 5.2 shows a running example of `abduction` tactic. When executed, this tactic collects from the context (the set of hypotheses) and the goal (the theorem to be proven) all the pairs of predicates and sends them to a Python server running on other process. The Python program evaluates the pairs for all relations in $\mathcal{R}$ with ComplEx and then sends back the triplets whose score is above a threshold θ. In Coq, these are converted to logical formulas according to Table 5.1 and finally added to the context as axioms.

In Coq, as one can compose tactics into another tactic, it is possible to construct a tactic that performs higher-order reasoning for natural language that additionally

[4] e.g., $(\texttt{puppy}, \texttt{hypernym}, \texttt{animal})$ follows from $(\texttt{puppy}, \texttt{hypernym}, \texttt{dog})$ and $(\texttt{dog}, \texttt{hypernym}, \texttt{animal})$.

[5] https://nlp.stanford.edu/projects/glove/. We use the one with 6 billion tokens.

injects lexical axioms when needed. Note that search-based axiom injection also can be performed with this tactic, by replacing the KBC-based scoring function with the search on databases.

We should note that one can combine our `abduction` tactic with other semantic theories (e.g. (Chatzikyriakidis and Luo, 2014); (Bernardy and Chatzikyriakidis, 2017)) and put the system in application. Coq has been used mainly for system verification and formalization of mathematics and there has been no tactic that is solely aimed at natural language reasoning. However, Coq provides an expressive formal language that combines higher-order logic and richly-typed functional programming language, and thus offers a general platform for various natural language semantics. We believe that with our work, it will be easier to develop an NLP systems based on advanced linguistic theories. This architecture also opens up new opportunities to integrate theorem proving and sophisticated machine-learning techniques. For example, we could implement in a tactic more complex tasks such as premise selection with deep models (Alemi et al., 2016).

5.5 Experiments

5.5.1 SICK dataset

We evaluate the proposed method on SICK dataset (Marelli et al., 2014), which we have used in Chapter 3 for evaluating our parser developed in the chapter (Section 3.7.1). Table 3.6 shows example problems from this dataset. Note that `ccg2lambda` itself is an unsupervised system and does not use any training data. We use the train part for the lemma list in data construction (Section 5.4.1) only and use the trial part to determine a threshold θ and to evaluate the processing speed.

5.5.2 New LexSICK lexical inference dataset

While SICK dataset provides a good testbed for evaluating logical inference involving linguistic phenomena such as negation and quantification, we found that the dataset is not ideal for evaluating complex lexical inference, specifically latent knowledge learned by a KBC model.

To evaluate the capability of our knowledge-empowered logic-based method, we construct our own dataset, which is small but challenging because of its combination

Id: (a) Label: CONTRADICTION
P: *A white and tan dog is running through the tall and green grass*
H: *A white and tan dog is ambling through a field*
Id: (b) Label: ENTAILMENT
P: *Someone is dropping the meat into a pan*
H: *The meat is being thrown into a pan*
Id: (c) Label: ENTAILMENT
P: *The man is singing and playing the guitar*
H: *The guitar is being performed by a man*
Id: (d) Label: CONTRADICTION
P: *A man and a woman are walking together through the wood*
H: *A man and a woman are staying together*
Id: (e) Label: ENTAILMENT
P: *A man is emptying a container made of plastic completely*
H: *A man is clearing a container made of plastic completely*

Table 5.2: LexSICK RTE problems require a mix of logical reasoning and external lexical knowledge.

of non-trivial lexical gaps and linguistic phenomena. Table 5.2 shows example problems, where lexical inferences are combined with linguistic phenomena such as quantification (Example b), verb coordination (Example c), and passive-active alternation (Example b, c). The process of the dataset construction is as follow: a human expert picks a sentence (premise) from SICK dataset, changes a word to its synonym/antonym according to thesauruses,[6] and then changes its sentence structure as well to construct a hypothesis.

The dataset (we refer to it as LexSICK) contains 58 problems, 29 of which is labeled *entailment*, 29 *contradiction*, and no *unknown* label. The problems in the dataset is listed in Appendix C.

5.5.3 Experimental settings

Settings for ComplEx Unless otherwise stated, we set the dimension of embeddings $n = 50$ and train it on the triplets obtained from WordNet (excluding VerbOcean

[6]We avoided WordNet and used thesaurus.com (http://www.thesaurus.com/) and Merriam-Webster (http://www.merriam-webster.com/).

Method	MRR	Hits		
		@1	@3	@10
ComplEx	77.68	71.07	81.76	90.08
ConvE	67.41	57.11	75.02	85.76

Table 5.3: The performance of KBC models trained and evaluated on WordNet triplets in Section 5.4.1. We report filtered scores.[7]

triplets) in Section 5.4.1 by minimizing the logistic loss (Eq. 5.2) using Adam optimizer. We use 1-N scoring (Section 5.2.2), since our dataset is fairly large compared to standard benchmark datasets. For the other hyperparameters, we use the same setting as in (Trouillon et al., 2016), except for the batch size of 128. In Table 5.3, we show Mean Reciprocal Rank (MRR) and Hits@N ($N = 1, 3, 10$) of ComplEx model (and the state-of-the-art ConvE (Dettmers et al., 2017) with the default hyperparameters for comparison) on development part of the dataset in Section 5.4.1.[7] The ComplEx model scores 77.68 in MRR, which is comparably lower than scores reported for the widely used WN18 benchmark data (above 93). Notably, ComplEx performs better than ConvE in terms of all metrics in this experiment. We adopt ComplEx in this work, since it achieves better results with much lower computational load.

Settings for `ccg2lambda` We decide the threshold $\theta = 0.4$ for filtering triplets based on the accuracy on the SICK trial set. As baselines, we replicate the system of Martínez-Gómez et al. (2017) with the default settings of `ccg2lambda`[8]. As we have done in Section 3.7.1, we use *multi-parsing* setting of the system, where it uses four CCG parsers and aggregates the results: `C&C`, `EasyCCG`, `EasySRL` and `depccg`.

We report accuracy / precision / recall / F1 on the test part and processing speed (macro average of five runs) in the trial set.[9]

[7] We report the scores in filtered setting. That is, compute the rank of o for gold (s, r, o) against all $e \in \mathcal{E}$ such that (s, r, e) is not in either of training or development data.

[8] We use a version of `ccg2lambda` committed to the master branch on October 2, 2017.

[9] We preprocess each RTE problem and do not include in the reported times those involved with CCG/semantic parsing. We set the time limit of proving to 100 seconds. These experiments are conducted on a machine with 18 core 2.30 GHz Intel Xeon CPU $\times$ 2.

System	Method			Dataset		Accuracy	Precision	Recall	F1	Speed
	search	KBC	tactic	WN	VO					
Mineshima et al. (2015)						77.30	98.93	48.07	64.68	3.79
Martínez-Gómez et al. (2017)	✓			✓		83.55	97.20	63.63	76.90	9.15
	✓			✓	✓	83.68	96.88	64.15	77.16	9.42
Ours	✓		✓	✓	✓	83.64	97.15	64.01	77.16	7.07
		✓	✓	✓		83.55	96.28	64.38	77.14	4.03
		✓	✓	✓	✓	83.45	95.75	64.47	77.04	3.84

Table 5.4: Results on the SICK test set. The results of the baseline systems are above the dashed line. The **Method** columns represent the use of search-based abduction, KBC-based abduction and `abduction` tactic, while the **Dataset** columns datasets used in abduction: WordNet (WN) and VerbOcean (VO). **Speed** shows macro average of processing time (sec.) of an RTE problem.

Method	#Correct
ResEncoder (Nie and Bansal, 2017)	18 / 58
search-based abduction	20 / 58
KBC-based abduction	21 / 58

Table 5.5: Experimental results on LexSICK. Both search- and KBC-based abductions use WordNet and VerbOcean.

5.5.4 Results on SICK set

Table 5.4 shows the experimental results on SICK. The first row is a baseline result without any abduction mechanism, followed by ones using WordNet and VerbOcean additively with the search-based axiom injection. By introducing our `abduction` tactic that is combined with the search-based axiom injection (4th row), we have successfully reduced the computation time (-2.35 sec. compared to 3rd row). By replacing the search-based abduction with the ComplEx model, the averaged time to process an RTE problem is again significantly reduced (5th row). The time gets close to the baseline without any database (only +0.24 sec.), with much improvement in terms of RTE performance, achieving the exactly same accuracy with Martínez-Gómez et al. (2017)'s WordNet-based abduction.

Finally, we re-train a ComplEx model with `similar` relations from VerbOcean (final row). When combining the datasets, we find that setting the dimension of embeddings larger to $n = 100$ leads to better performance. This may help the KBC model

accommodate the relatively noisy nature of VerbOcean triplets. The VerbOcean triplets have contributed to the improved recall by providing more knowledge not covered by WordNet, while it has resulted in drops in the other metrics. Inspecting the details, it has actually solved more examples than when using only WordNet triplets; however, noisy relations in the data, such as $(\texttt{black}, \texttt{similar}, \texttt{white})$, falsely lead to proving *entailment* / *contradiction* for problems whose true label is *unknown*. The higher recall has contributed to the overall processing speed, since it has led more problems to be proven before the timeout (-0.25 sec.).

We conduct detailed error analysis on 500 SICK trial problems. The RTE system combined with our KBC-based abduction (trained only on WordNet) has correctly solved one more problem than a Martínez-Gómez et al. (2017)'s baseline system (which additionally uses VerbOcean), resulting in the accuracy of 84.8%. In this subset, we found the following error cases: 71 cases related to lexical knowledge, 4 failures in CCG parsing, and 1 timeout in theorem proving. This shows that CCG parsing is quite reliable. Around five cases among lexical ones are due to false positively asserted lexical axioms, while the most of the others are due to the lack of knowledge. In the following, we exclusively focus on issues related to lexical inference.

5.5.5 Evaluating latent knowledge

Table 5.5 shows experimental results on LexSICK dataset. For comparison, we add a result of ResEncoder (Nie and Bansal, 2017), one of the state-of-the-art neural network-based RTE models. When trained on the training part of SICK, it scores 82.00% accuracy on the SICK test part, while performs badly on LexSICK, indicating this dataset is more challenging. The accuracies of two logic-based methods are also not high, suggesting the difficulty of this problem. The KBC-based method shows the same tendency as in the results in the previous section; it solves more examples than the search-based method, along with false positive predictions.

We observe interesting examples that show the effectiveness of using latent lexical knowledge. One is Example (d) in Table 5.2, for which a latent relation $(\texttt{walk}, \texttt{antonym}, \texttt{stay})$ is predicted by the KBC model, while it is not available for the search-based method. Another case is Example (e) in Table 5.2, where our method obtains the axiom $\forall x.\texttt{clear}(x) \rightarrow \texttt{empty}(x)$ by scoring the triplet $(\texttt{empty}, \texttt{synonym}, \texttt{clear})$ as high as $(\texttt{empty}, \texttt{hyponym}, \texttt{clear})$, leading to the correct prediction. The search-based method derives only $\forall x.\texttt{empty}(x) \rightarrow \texttt{clear}(x)$, which is

not relevant in this case.

Similarly, by examining the results on SICK dataset, we found more examples that the KBC-based method has successfully solved by utilizing latent lexical knowledge. For example, in SICK-6874 we have a premise *A couple of white dogs are running and jumping along a beach* and a hypothesis *Two dogs are playing on the beach*. The KBC model successfully proves the inference by producing multiple axioms: $\forall x.\texttt{along}(x) \rightarrow \texttt{on}(x)$, $\forall x.\texttt{couple}(x) \rightarrow \texttt{two}(x)$ and $\forall x.\texttt{run}(x) \rightarrow \texttt{play}(x)$. One characteristic of the KBC-based method is that it can utilize lexical relations between general words such as frequent verbs and prepositions. Though this may cause more false positive predictions, our experiments showed that under the control of the abduction method, it contributed to improving recall while keeping high precision.

5.6 Summary

In this work, we have proposed an automatic axiom injection mechanism for natural language reasoning based on Knowledge Base Completion. In the experiments, we show that our method has significantly improved the processing speed of an RTE problem, while it also achieves the competitive accuracy, by utilizing the obtained latent knowledge.

In future work, we are interested in extending this framework to generate phrasal axioms (e.g., $\forall x.\texttt{have}(x) \wedge \texttt{fun}(x) \rightarrow \texttt{enjoy}(x)$). Generating this sort of axioms accurately is the key for logic-based systems to achieve high performance. In order to do that, we will need a framework to learn to compute compositionally a vector from those of *have* and *fun* and to predict relations such as `synonym` between the vector and that of *enjoy*.

Chapter 6

Conclusion

In this thesis, we have addressed problems in developing natural language inference systems based on CCG and logic. The main contributions are summarized as follows: the development of (1) an efficient and accurate CCG parsing method, (2) domain adaptation method of CCG parsing, and (3) knowledge insertion mechanism for logic-based natural language inference system.

In Chapter 3, we have proposed a new probabilistic model of a CCG tree and an efficient parsing algorithm based on the A* algorithm. The key of the proposed method is that it models the probability of a CCG tree by local factors of supertags and dependency edges, which are predicted independently with a strong unigram model defined over bidirectional LSTMs. In the experiments, we have demonstrated the proposed method is effective in various settings: in CCGbank parsing in the English and Japanese languages, and in RTE experiments for the two languages.

In Chapter 4, we have focused on the domain adaptation issue of a CCG parser. We have proposed a new domain adaptation method for CCG parsing, where we exploit cheaper and abundant dependency-based treebanks and convert them to the corresponding CCGbanks, using a dependency-to-CCG converter developed by extending the parsing technique in the previous chapter. We have conducted extensive parsing experiments; on top of existing benchmark datasets on (1) biomedical texts and (2) question sentences, we have evaluated the method on (3) speech conversation and (4) math problems. Using the proposed technique, we have demonstrated the parser developed in Chapter 3 shows further improvement, from 90.7% to 96.6% on speech conversation, and from 88.5% to 96.8% on math problems.

In Chapter 5, we have solved an issue of logic-based natural language inference systems: the tension between adding more knowledge data to the system for the ca-

pability of wider coverage of reasoning types, and the system's efficiency in solving a problem. We have shown the processing time of an RTE system can be significantly reduced by replacing its search-based knowledge insertion mechanism by that based on Knowledge Base Completion. We have also integrated the mechanism within a Coq plugin. We have shown empirically that, in this framework, adding new knowledge data contributes to better RTE performance while not harming the processing speed.

Throughout this thesis, we have demonstrated logic-based systems can perform on par with the latest deep learning-based models on some RTE benchmarks (e.g., SICK), and they even show superiority on the linguistically challenging FraCaS and JSeM datasets, owing to their strengths in handling linguistic constructions. However, we are aware of some weaknesses of these systems. For example, some RTE datasets such as SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018) are currently difficult for these systems due to several reasons. To solve these datasets, among others, a system must be equipped with knowledge on relations among multiple words (phrases), in addition to those between single words; for a system to be able to judge that *A black race car starts up in front of a crowd of people* does not entail *A man is driving down a lonely road* (a problem from SNLI), the system must know that pairs of phrases *a crowd of people* and *lonely*, and *car starts up* and *driving* are related to each other. As pointed out in the conclusion of Chapter 5, one of solutions to handle these relations is to extend the RTE systems with phrasal axioms. On top of that, there seem to be two promising directions to address this issue.

There are some projects on constructing so-called *Entailment Graph*, whose nodes are text segments (words or any longer phrases) and directed edge between nodes represents entailment relation (Hosseini et al., 2018, 2019). The constructed graph will be used, for example, in a Question Answering system, where various ways of asking a question are normalized using the knowledge graph (e.g., *Does Verizon own Yahoo?* vs. Does Verizon buy Yahoo? vs. *Is Yahoo owned by Verizon?*). The graph will naturally be useful in solving RTE tasks as well; it can be converted to a set of logical formulas, to be used in theorem proving.

The other approach is the *Vector-based Semantics* (Kartsaklis and Sadrzadeh, 2016; Wijnholds and Sadrzadeh, 2019), in which they replace logical meaning representations with vector embeddings. Their vector representations for larger phrases are calculated according to the syntactic structure using a tensor-based operation. The research group have been conducting challenging experiments, such as the evaluation of whether their method can handle phenomena such as the extraction of relative clauses,

by constructing the RELPRON dataset (Rimell et al., 2016). Though the result in the paper is unfortunately not as expected, where the simple vector addition model beats others, we believe that this direction is also intriguing and promising.

Finally, it must be an interesting venue to pursue to transfer the success in this thesis to languages other than English and Japanese. The idea of converting dependency-based resources to CCGbanks may be used to construct a CCG alternative of Universal Dependencies (Nivre et al., 2016). Concretely, we can use the techniques of recently developed unsupervised constituency/dependency parsing (He et al., 2018; Kim et al., 2019), and instead of simply applying them to CCG, we can use the techniques in a way that a CCG tree is generated conditionally on the corresponding UD tree. Evaluating our parser on languages in Parallel Meaning Bank (Abzianidze et al., 2017) must be also an interesting future direction.

Appendix A

Details of English CCG Grammar

In the following we summarize the details of English and Japanese CCG grammars implemented in our parser. Please refer to Table 2.1 for the details of the basic combinatory rules.
The set of binary combinatory rules:

- forward application $>$
- backward application $<$
- forward composition $>\mathbf{B}^1$
- backward crossed composition $<\mathbf{B}^1_\times$
- generalized forward composition $>\mathbf{B}^n$, where $n = 2$
- generalized backward crossed composition $<\mathbf{B}^n_\times$, where $n = 2$

The other binary rules are summarized in Table A.1. Here, $PUNCT \in \{comma, conj, semicolon\}$. Notationally, $C_{\mathtt{x|y}}$ is an atomic category having either of `x` or `y` as its feature. This rule generalizes to the case with more than one vertical bar. Rules marked with * are found in the implementation of `EasySRL`. In this work, we also employ these rules. The set of unary rules is summarized in Table A.2.

Our parser is configured to output only those derivations whose root category is from the following list:

$S_{\mathtt{dcl}}$, $S_{\mathtt{wq}}$, $S_{\mathtt{q}}$, $S_{\mathtt{qem}}$, NP

conjunction	$\frac{conj \quad X}{X\backslash X}\,\Phi$
remove punctuation right	$\frac{PUNCT \quad X}{X}\,\langle rp \rangle$
remove punctuation left	$\frac{X \quad PUNCT}{X}\,\langle lp \rangle$
comma_verb_phrase_to_adverb*	$\frac{comma \quad S_{\texttt{ng\|pass}}\backslash NP}{(S\backslash NP)\backslash(S\backslash NP)}\,\langle * \rangle$
parenthetical_direct_speech*	$\frac{comma \quad S_{\texttt{dcl}}/S_{\texttt{dcl}}}{(S\backslash NP)/(S\backslash NP)}\,\langle * \rangle$

Table A.1: Additional binary rules in our English parser. X is a variable matching any category, and $PUNCT \in \{comma, conj, semicolon\}$.

Bare noun phrases	$\frac{N}{NP}^{<un>}$
Type raising	$\frac{NP}{S/(S\backslash NP)}^{<un>}$ $\frac{NP}{(S\backslash NP)/((S\backslash NP)/NP)}^{<un>}$ $\frac{PP}{(S\backslash NP)/((S\backslash NP)/PP)}^{<un>}$
Reduced relative clause	$\frac{S_{\texttt{pss\|ng\|adj\|to}}\backslash NP}{NP\backslash NP}^{<un>}$ $\frac{S_{\texttt{to}}\backslash NP}{N\backslash N}^{<un>}$ $\frac{S_{\texttt{dcl}}/NP}{NP\backslash NP}^{<un>}$
VP sentence modifiers	$\frac{S_{\texttt{pss\|ng\|to}}\backslash NP}{S/S}^{<un>}$

Table A.2: The set of unary rules in the English parser.

Appendix B

Details of Japanese CCG Grammar

The set of binary combinatory rules:

- forward application $>$
- backward application $<$
- forward composition $>\mathbf{B}^1$
- backward composition $<\mathbf{B}^1$
- generalized backward composition $<\mathbf{B}^n$ for $n \in \{2, 3, 4\}$
- cross forward composition $>\mathbf{B}^n$ for $n \in \{2, 3\}$
- SSEQ

SSEQ combines two constituents each licensed as a sentence; for all category X in the set of categories that can appear at a root of a CCG tree (defined below), $X \;\; X \Rightarrow X$. The set of unary rules are summarized in Table B.1. In the table, $S_{\texttt{mod=X,form=Y,fin=Z}}$ is abbreviated as $S_{\texttt{X,Y,Z}}$ and $NP_{\texttt{case=X,mod=Y,fin=Z}}$ as $NP_{\texttt{X,Y,Z}}$ for simplicity.

Our parser outputs only derivations whose root category is from the following list:

$NP_{\texttt{nc,nm,f}}$, $NP_{\texttt{nc,nm,t}}$, $S_{\texttt{nm,attr,t}}$, $S_{\texttt{nm,base,f}}$, $S_{\texttt{nm,base,t}}$, $S_{\texttt{nm,cont,f}}$, $S_{\texttt{nm,cont,t}}$, $S_{\texttt{nm,da,f}}$, $S_{\texttt{nm,da,t}}$, $S_{\texttt{nm,hyp,t}}$, $S_{\texttt{nm,imp,f}}$, $S_{\texttt{nm,imp,t}}$, $S_{\texttt{nm,r,t}}$, $S_{\texttt{nm,s,t}}$, $S_{\texttt{nm,stem,f}}$, $S_{\texttt{nm,stem,t}}$

Adnominal Modifier (Inner relationship)	$\dfrac{S_{\texttt{adn,attr\|base,f}} \backslash NP_{\texttt{ga\|o,nm,f}}}{NP_{\texttt{nc,X1,X2}} / NP_{\texttt{nc,X1,X2}}}\,\langle un \rangle$
Adnominal Modifier (Outer relationship)	$\dfrac{S_{\texttt{adn,attr\|base\|cont\|hyp\|imp\|stem,f}}}{NP_{\texttt{nc,X1,X2}} / NP_{\texttt{nc,X1,X2}}}\,\langle un \rangle$
Adverbial Modifier	$\dfrac{\dfrac{S_{\texttt{adv,cont\|hyp\|stem,f}}}{S_{\texttt{X1,X2,X3}} / S_{\texttt{X1,X2,X3}}}\,\langle un \rangle \quad S_{\texttt{adv,cont,f}} \backslash NP_{\texttt{ga,nm,f}}}{(S_{\texttt{X1,X2,X3}} \backslash NP_{\texttt{ga,nm,f}}) / (S_{\texttt{X1,X2,X3}} \backslash NP_{\texttt{ga,nm,f}})}\,\langle un \rangle$
Other	$\dfrac{NP_{\texttt{nc,adv,f}}}{S_{\texttt{X1,X2,X3}} / S_{\texttt{X1,X2,X3}}}\,\langle un \rangle$

Table B.1: The set of unary rules in the Japanese parser. $C_{\texttt{x\|...\|y}}$ is an atomic category having a feature listed in x|...|y.

Appendix C

LexSICK dataset

Premise and Hypothesis	Label
P: *A girl in white is dancing* H: *A girl in white is stepping*	entailment
P: *A white and tan dog is running through the tall and green grass* H: *A white and tan dog is strolling through the tall and green grass*	contradiction
P: *A white and tan dog is running through the tall and green grass* H: *A white and tan dog is ambling through a field*	contradiction
P: *A man and a woman are hiking through a wooded area* H: *A man and a woman are staying*	contradiction
P: *A man and a woman are walking together through the woods* H: *A man and a woman are staying together*	contradiction
P: *The woman is measuring the other woman* H: *A woman is being scaled by another woman*	entailment
P: *A man is emptying a container made of plastic completely* H: *A man is clearing a container made of plastic completely*	entailment
P: *A man is emptying a container made of plastic completely* H: *A man is filling a container made of plastic*	contradiction
P: *Some cameras are being burned by a person with a blow torch* H: *Some cameras are being melted by a person with a blow torch*	entailment
P: *Some cameras are being burned by a person with a blow torch* H: *Some cameras are being heated by a person with a blow torch*	entailment
P: *The lady is slicing a tomato* H: *Someone is chopping a tomato*	entailment
P: *Someone is strumming the guitar* H: *Someone is picking the guitar.*	entailment
P: *A bull dog is being brushed by the monkey* H: *A bull dog is being cleaned by the monkey.*	entailment
P: *A bull dog is being brushed by the monkey* H: *The monkey is washing a bull dog*	entailment
P: *A bull dog is being brushed by the monkey* H: *The monkey is sweeping a bull dog.*	entailment
P: *A guinea pig, which is small, is gnawing and eating a piece of carrot on the floor*	entailment

H: *A small guinea pig is gnawing and champing a piece of carrot on the floor*	
P: *A guinea pig, which is small, is gnawing and eating a piece of carrot on the floor* H: *A guinea pig, which is small, is nibbling and eating a piece of carrot on the floor*	entailment
P: *The man is lifting barbells* H: *The man is dropping barbells*	contradiction
P: *The man is lifting barbells* H: *The man is lowering barbells*	contradiction
P: *A large green ball is hitting a potato* H: *A large green colored ball is missing a potato*	contradiction
P: *A kitten is getting bored* H: *A kitten is animated*	contradiction
P: *A kitten is getting bored* H: *A kitten is energized*	contradiction
P: *A woman is placing two eggs into a pot of water* H: *A woman is setting two eggs into a pot of water*	entailment
P: *A boy is waving at some young runners from the ocean* H: *A boy is gesturing at some young runners from the ocean*	entailment
P: *A boy is nodding at some young runners from the ocean* H: *A boy is gesturing at some young runners from the ocean*	entailment
P: *A woman is cleaning a shrimp* H: *The woman is contaminating a shrimp*	contradiction
P: *A woman is cleaning a shrimp* H: *Someone is dirtying a shrimp*	contradiction
P: *A panda bear is eating some bamboo* H: *Some bamboo is being consumed by a panda bear*	entailment
P: *A panda bear is eating some bamboo* H: *Some bamboo is being devoured by a panda bear*	entailment
P: *A man is unfolding a tortilla* H: *A man is unrolling a tortilla*	entailment
P: *A woman is freeing a fish* H: *A woman is releasing a fish*	entailment
P: *A person is jotting something down with a pencil* H: *A person is scribbling something with a pencil*	entailment
P: *A person is scribbling something with a pencil* H: *A person is writing*	entailment
P: *The man is skillfully playing the guitar* H: *The man is awkwardly playing a guitar*	contradiction
P: *Some fish are swimming quickly.* H: *Some fish are swimming slowly*	contradiction
P: *Someone is dropping the meat into a pan* H: *The meat is being thrown into a pan*	entailment
P: *The person is slicing onions* H: *Onions are being cut by the person*	entailment
P: *The person is slicing onions* H: *Onions are being split by the person*	entailment
P: *A person is scrubbing a zucchini* H: *The person is brushing a zucchini*	entailment
P: *A person is scrubbing a zucchini*	entailment

H: *The person is dirtying a zucchini*	
P: *A woman is slicing a pepper which is green* H: *A woman is chopping a green pepper*	entailment
P: *The man is singing and playing the guitar* H: *The guitar is being performed by a man*	entailment
P: *The man is singing and performing with a guitar* H: *The guitar is being played by a man*	entailment
P: *A man is recklessly climbing a rope* H: *A man is ascending a rope*	entailment
P: *A man is recklessly climbing a rope* H: *A man is descending a rope*	contradiction
P: *A man is recklessly climbing a rope* H: *A man is descending a rope*	contradiction
P: *A yellow dog is running on white snow on a very sunny day* H: *A yellow dog is running on a rainy day*	contradiction
P: *A man is recklessly climbing a rope* H: *A man is rashly climbing a rope*	entailment
P: *A man is recklessly climbing a rope* H: *A man is carelessly climbing a rope*	contradiction
P: *A man is recklessly climbing a rope* H: *A man is cautiously climbing a rope*	contradiction
P: *The doctors are healing a man* H: *The doctor is damaging the man*	contradiction
P: *The doctors are healing a man* H: *The doctor is hurting the man*	contradiction
P: *A large flute is being played by a man* H: *A man is playing a tiny flute*	contradiction
P: *A yellow dog is running on white snow on a very sunny day* H: *A yellow dog is running on a rainy day*	contradiction
P: *A yellow dog is running on white snow on a very sunny day* H: *A yellow dog is running on a cloudy day*	contradiction
P: *A large brown dog and a small grey dog are standing on a rocky surface.* H: *A big brown dog and a small grey dog are moving*	contradiction
P: *The group of people is standing together and looking at the camera* H: *A group of people is standing together and ignoring the camera*	contradiction
P: *The group of people is sitting in a dim room* H: *The group of people is sitting in a room which is shiny*	contradiction

References

Martín Abadi et al. 2016. TensorFlow: A System for Large-scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Savannah, GA, USA. USENIX Association.

Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2492–2502, Lisbon, Portugal. Association for Computational Linguistics.

Lasha Abzianidze. 2016. *A Natural proof System for Natural Language*. Ph.D. thesis, Tilburg University.

Lasha Abzianidze. 2017. LangPro: Natural Language Theorem Prover. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 115–120, Copenhagen, Denmark. Association for Computational Linguistics.

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 242–247, Valencia, Spain. Association for Computational Linguistics.

Kazimierz Ajdukiewicz. 1997. Die Syntaktische Konnexität. *Studia Philosophica*, 1:1–27.

Alexander A. Alemi, François Chollet, Niklas Een, Geoffrey Irving, Christian Szegedy, and Josef Urban. 2016. DeepMath - Deep Sequence Models for Premise Selection. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2243–2251, Barcelona, Spain. Curran Associates Inc.

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2013. Using CCG categories to improve Hindi dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 604–609. Association for Computational Linguistics.

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

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

Srinivas Bangalore and Aravind K Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2):237–265.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219.

Daisuke Bekki. 2010. *A Formal Theory of Japanese Grammar: The Conjugation System, Syntactic Structures, and Semantic Composition*. Kuroshio. (In Japanese).

Johan van Benthem. 2008. A brief history of natural logic. In *Logic, Navya-Nyāya & Applications: Homage to Bimal Krishna Matilal*, pages 21–42. College Publications.

Jean-Philippe Bernardy and Stergios Chatzikyriakidis. 2017. A Type-Theoretical system for the FraCaS test suite: Grammatical Framework meets Coq. In *IWCS 2017 - 12th International Conference on Computational Semantics*.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic Tagging with Deep Residual Networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, pages 3531–3541, Osaka, Japan. The COLING 2016 Organizing Committee.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 2787–2795, Lake Tahoe, Nevada. Curran Associates Inc.

Johan Bos, Valerio Basile, Kilian Evang, Noortje J. Venhuizen, and Johannes Bjerva. 2017. The Groningen Meaning Bank. In *Handbook of Linguistic Annotation*, pages 463–496. Springer Netherlands.

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

Johan Bos, Bosco Cristina, and Mazzei Alessandro. 2009. Converting a Dependency Treebank to a Categorial Grammar Treebank for Italian. In *In Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories*, pages 27–38.

Johan Bos and Katja Markert. 2005. Recognising Textual Entailment with Logical Inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Stergios Chatzikyriakidis and Zhaohui Luo. 2014. Natural Language Inference in Coq. *Journal of Logic, Language and Information*, 23(4):441–480.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2014. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2635–2639, Singapore.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural Natural Language Inference Models Enhanced with External Knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2406–2417, Melbourne, Australia. Association for Computational Linguistics.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 33–40, Barcelona, Spain. Association for Computational Linguistics.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics, Volume 33, Number 4, December 2007*, 33(4):493–552.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building Deep Dependency Structures with a Wide-coverage CCG Parser. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 327–334, Philadelphia, Pennsylvania. Association for Computational Linguistics.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *Proceedings of the 2016 International Conference on Learning Representations*, San Juan, Puerto Rico.

Robin Cooper, Richard Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspers, Hans Kamp, Manfred Pinkal, Massimo Poesio, Stephen Pulman, et al. 1994. FraCaS–a framework for computational semantics. *Deliverable*, D6.

David Delahaye. 2000. A Tactic Language for the System Coq. In *Logic for Programming and Automated Reasoning*, pages 85–95. Springer.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 2017 International Conference on Learning Representations*, Toulon, France.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and A. Noah Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jason Eisner. 1996a. Efficient Normal-Form Parsing for Combinatory Categorial Grammar. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86, Santa Cruz, California, USA. Association for Computational Linguistics.

Jason M. Eisner. 1996b. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of COLING 1996, The 16th International Conference on Computational Linguistics*, pages 340–345, Copenhagen, Denmark. The COLING 1996 Organizing Committee.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. SWITCHBOARD: Telephone Speech Corpus for Research and Development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 517–520, San Francisco, California. IEEE Computer Society.

Izumi Haruta, Koji Mineshima, and Daisuke Bekki. 2019. A CCG-based Compositional Semantics and Inference System for Comparatives. In *Proceedings of the 33rd Pacific Asia Conference on Language, Information and Computation*, Hakodate. Association for Computational Linguistics.

Shinkichi Hashimoto. 1934. *Essentials of Japanese Grammar (Kokugoho Yousetsu)*. Iwanami.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Unsupervised Learning of Syntactic Structure with Invertible Neural Projections. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1292–1302, Brussels, Belgium. Association for Computational Linguistics.

Julia Hockenmaier and Yonatan Bisk. 2010. Normal-form parsing for Combinatory Categorial Grammars with generalized composition and type-raising. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 465–473, Beijing, China. Coling 2010 Organizing Committee.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Matthew Honnibal and Mark Johnson. 2014. Joint Incremental Disfluency Detection and Dependency Parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142.

Mohammad Javad Hosseini, Nathanael Chambers, Siva Reddy, Xavier R. Holt, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2018. Learning Typed En-

tailment Graphs with Global Soft Constraints. *Transactions of the Association for Computational Linguistics*, 6:703–717.

Mohammad Javad Hosseini, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2019. Duality of Link Prediction and Entailment Graph Induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4736–4746, Florence, Italy. Association for Computational Linguistics.

Thomas Icard and Lawrence Moss. 2014. Recent progress in monotonicity. *Linguistic Issues in Language Technology (LiLT)*, 9:1–29.

Xinzhou Jiang, Zhenghua Li, Bo Zhang, Min Zhang, Sheng Li, and Luo Si. 2018. Supervised Treebank Conversion: Data and Approaches. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2706–2716, Melbourne, Australia. Association for Computational Linguistics.

Vidur Joshi, Matthew Peters, and Mark Hopkins. 2018. Extending a Parser to Distant Domains Using a Few Dozen Partially Annotated Examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1190–1199, Melbourne, Australia. Association for Computational Linguistics.

John Judge, Aoife Cahill, and Josef van Genabith. 2006. QuestionBank: Creating a Corpus of Parse-Annotated Questions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 497–504, Sydney, Australia. Association for Computational Linguistics.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. kluwer.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2016. Distributional Inclusion Hypothesis for Tensor-based Composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2849–2860, Osaka, Japan. The COLING 2016 Organizing Committee.

Daisuke Kawahara, Sadao Kurohashi, and Kôiti Hasida. 2002. Construction of a Japanese Relevance-tagged Corpus. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, Las Palmas, Canary Islands - Spain. European Language Resources Association.

Ai Kawazoe, Ribeka Tanaka, Koji Mineshima, and Daisuke Bekki. 2017. An Inference Problem Set for Evaluating Semantic Theories and Semantic Processing Systems for Japanese. In *New Frontiers in Artificial Intelligence: JSAI-isAI 2015 Workshops, LENLS, JURISIN, AAA, HAT-MASH, TSDAA, ASD-HR, and SKL*, pages 58–65, Cham. Springer International Publishing.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Dan Klein and Christopher D. Manning. 2003. A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 40–47.

Taku Kudo and Yuji Matsumoto. 2002. Japanese Dependency Analysis using Cascaded Chunking. In *Proceedings of the 6th Conference on Natural Language Learning, CoNLL 2002*, pages 63–69, Taipei, Taiwan. Association for Computational Linguistics.

Jonathan K. Kummerfeld, Dan Klein, and James R. Curran. 2012. Robust Conversion of CCG Derivations to Phrase Structure Trees. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 105–109, Jeju Island, Korea. Association for Computational Linguistics.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Global Neural CCG Parsing with Optimality Guarantees. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2366–2376, Austin, Texas. Association for Computational Linguistics.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-Rank Tensors for Scoring Dependency Structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1381–1391, Baltimore, Maryland. Association for Computational Linguistics.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG Parsing and Semantic Role Labelling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1444–1454, Lisbon, Portugal. Association for Computational Linguistics.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, San Diego, California. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014a. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 990–1000, Doha, Qatar. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014b. Improved CCG Parsing with Semi-supervised Supertagging. *Transactions of the Association for Computational Linguistics*, 2:327–338.

Zhenghua Li, Min Zhang, Yue Zhang, Zhanyi Liu, Wenliang Chen, Hua Wu, and Haifeng Wang. 2016. Active Learning for Dependency Parsing with Partial Annotation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 344–354, Berlin, Germany. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2008. Modeling Semantic Containment and Exclusion in Natural Language Inference. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 521–528, Manchester, UK. Coling 2008 Organizing Committee.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the Workshop on Human Language Technology*, pages 114–119, Plainsboro, NJ. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):314–330.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 216–223, Reykjavik, Iceland. European Language Resources Association.

M. Marneffe, B. Maccartney, and C. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454, Genoa, Italy. European Language Resources Association.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, Genoa, Italy. European Language Resources Association (ELRA).

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2017. On-demand Injection of Lexical Knowledge for Recognising Textual Entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 710–720, Valencia, Spain. Association for Computational Linguistics.

Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H. Arai. 2017. Semantic Parsing of Pre-university Math Problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 2131–2141, Vancouver, Canada. Association for Computational Linguistics.

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

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061, Lisbon, Portugal. Association for Computational Linguistics.

Koji Mineshima, Ribeka Tanaka, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2016. Building compositional semantics and higher-order inference

system for a wide-coverage Japanese CCG parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2242, Austin, Texas. Association for Computational Linguistics.

Seyed Abolghasem Mirroshandel and Alexis Nasr. 2011. Active Learning for Dependency Parsing Using Partially Annotated Sentences. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 140–149, Dublin, Ireland. Association for Computational Linguistics.

Jeff Mitchell and Mark Steedman. 2015. Parser Adaptation to the Biomedical Domain without Re-Training. In *Proceedings of the Sixth International Workshop on Health Text Mining and Information Analysis*, pages 79–89, Lisbon, Portugal. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Richard Montague. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven.

Reinhard Muskens. 2010. An Analytic Tableau System for Natural Logic. In *Logic, Language and Meaning*, pages 104–113, Berlin, Heidelberg. Springer Berlin Heidelberg.

Graham Neubig et al. 2017. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980*.

Yixin Nie and Mohit Bansal. 2017. Shortcut-Stacked Sentence Encoders for Multi-Domain Inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 41–45, Copenhagen, Denmark. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference*

on Language Resources and Evaluation, pages 1659–1666, Portorož, Slovenia. European Language Resources Association.

Hiroshi Noji and Yusuke Miyao. 2016. Jigg: A Framework for an Easy Natural Language Processing Pipeline. In *Proceedings of ACL-2016 System Demonstrations*, pages 103–108, Berlin, Germany. Association for Computational Linguistics.

Terence Parsons. 1990. *Events in The Semantics of English: a Study in Subatomic Semantics*. The MIT Press.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Dag Prawitz. 1965. *Natural Deduction – A Proof Theoretical Study*. Almqvist & Wiksell.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale Semantic Parsing without Question-Answer Pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark. Association for Computational Linguistics.

Laura Rimell and Stephen Clark. 2008. Adapting a Lexicalized-Grammar Parser to Contrasting Domains. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 475–484, Honolulu, Hawaii. Association for Computational Linguistics.

Laura Rimell, Jean Maillard, Tamara Polajnar, and Stephen Clark. 2016. RELPRON: A Relative Clause Evaluation Data Set for Compositional Distributional Semantics. *Computational Linguistics*, 42(4):661–701.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. In *Proceedings of the 2016 International Conference on Learning Representations*, San Juan, Puerto Rico.

Kenji Sagae, Yusuke Miyao, and Jun'ichi Tsujii. 2007. HPSG Parsing with Shallow Dependency Constraints. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 624–631, Prague, Czech Republic. Association for Computational Linguistics.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning Character-level Representations for Part-of-speech Tagging. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, pages 1818–1826, Beijing, China. JMLR.org.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1476, Lisbon, Portugal. Association for Computational Linguistics.

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

Mark Steedman. 2012. *Taking Scope: The Natural Semantics of Quantifiers*. The MIT Press.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005. Syntax Annotation for the GENIA Corpus. In *Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts*, pages 220–225. Association for Computational Linguistics.

Hideo Teramura. 1969. The Syntax of Noun Modification in Japanese. *The Journal-Newsletter of the Association of Teachers of Japanese*, 6(1):64–74.

The Coq Development Team. 2017. The Coq Proof Assistant: Reference Manual: Version 8.7.1. INRIA.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a Next-Generation Open Source Framework for Deep Learning. In *Proceedings of Workshop on Machine Learning Systems in The Twenty-ninth Annual Conference on Neural Information Processing Systems*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pages 2071–2080, New York, USA. JMLR.org.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 1999. Japanese Dependency Structure Analysis Based on Maximum Entropy Models. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 196–203, Bergen, Norway. Association for Computational Linguistics.

Sumire Uematsu, Takuya Matsuzaki, Hiroki Hanaoka, Yusuke Miyao, and Hideki Mima. 2015. Integrating Multiple Dependency Corpora for Inducing Wide-

Coverage Japanese CCG Resources. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 14(1):1:1–1:24.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging With LSTMs. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 232–237, San Diego, California. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured Training for Neural Network Transition-Based Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 323–333, Beijing, China. Association for Computational Linguistics.

Gijs Wijnholds and Mehrnoosh Sadrzadeh. 2019. Evaluating Composition Models for Verb Phrase Elliptical Sentence Embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 261–271, Minneapolis, Minnesota. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Wenduan Xu, Stephen Clark, and Yue Zhang. 2014. Shift-Reduce CCG Parsing with a Dependency Model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 218–227, Baltimore, Maryland. Association for Computational Linguistics.

Yao-zhong Zhang, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010. A Simple Approach for HPSG Supertagging Using Dependency Information. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 645–648, Los Angeles, California. Association for Computational Linguistics.

List of Publications

Journal

1. Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto, *A* CCG Parsing with a Supertag and Dependency Factored Model*, 自然言語処理, Vol.26 No.1, pp. 83–119, March 2019.

International Conferences (Refereed)

1. Masashi Yoshikawa, Hiroshi Noji, Koji Mineshima and Daisuke Bekki, *Automatic Generation of High Quality CCGbanks for Parser Domain Adaptation*, In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), pp. 129–139, Florence, Italy, July 2019.

2. Masashi Yoshikawa, Koji Mineshima, Hiroshi Noji and Daisuke Bekki, *Combining Axiom Injection and Knowledge Base Completion for Efficient Natural Language Inference*, In Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI-2019), pp. 7410–7417, Honolulu, Hawaii, USA, January 2019.

3. Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto, *A* CCG Parsing with a Supertag and Dependency Factored Model*, In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-2017), pp. 277–287, Vancouver, Canada, July 2017.

Other Publications (Refereed)

1. Riko Suzuki, Hitomi Yanaka, Masashi Yoshikawa, Koji Mineshima and Daisuke Bekki, *Building a Logical Inference System for Visual-Textual Entailment*, In

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop (ACL-SRW 2019), pp. 386–392, Florence, Italy, July 2019.

2. Riko Suzuki, Hitomi Yanaka, Masashi Yoshikawa, Koji Mineshima and Daisuke Bekki, *Towards building a logical inference system for image retrieval*, In Proceedings of the Second Workshop on Shortcomings in Vision and Language (SiVL) NAACL-HLT2019, 2 pages, Minneapolis, USA, June 2019.

3. Kana Manome, Masashi Yoshikawa, Hitomi Yanaka, Pascual Martínez-Gómez, Koji Mineshima and Daisuke Bekki, *Neural sentence generation from formal semantics*, In Proceedings of the 11th International Conference on Natural Language Generation (INLG-2018), pp.408–414, Tilburg, Netherlands, November 2018.

4. Masashi Yoshikawa, Koji Mineshima, Hiroshi Noji and Daisuke Bekki, *Consistent CCG Parsing over Multiple Sentences for Improved Logical Reasoning*, In Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2018), pp.407–412, New Orleans, USA, June 2018.

5. Masashi Yoshikawa, Hiroyuki Shindo, and Yuji Matsumoto, *Joint Transition-based Dependency Parsing and Disfluency Detection for Automatic Speech Recognition Texts*, In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2016), pp.1036–1041, Texas, USA, November 2016.

Domestic Conferences (Non-refereed)

1. 鈴木莉子, 吉川将司, 谷中瞳, 峯島宏次, 戸次大介, 「テキスト情報と画像情報を組み合わせた論理推論システムの構築」, 人工知能学会第 33 回全国大会 (JSAI2019), 4 pages, June 2019.

2. 吉川将司, 能地宏, 峯島宏次, 戸次大介, 「係り受け木を用いたツリーバンク自動生成による CCG 解析分野適応」, 言語処理学会第 25 回年次大会 (NLP2019), pp. 197–200, March 2019.

3. 吉川将司, 峯島宏次, 能地宏, 戸次大介, 「知識ベース補完を用いた高階論理推論のための自動公理生成」, 言語処理学会第 24 回年次大会 (NLP2018), pp. 113–116, March 2018.

4. 馬目華奈, 谷中瞳, 吉川将司, 峯島宏次, 戸次大介, 「RNN 系列変換モデルを用いた高階論理式からの文生成」, 言語処理学会第 24 回年次大会 (NLP2018), pp. 380–383, March 2018.

5. 吉川 将司, 能地 宏, 松本 裕治, 「係り受け構造とスーパータグの同時予測による A* CCG 解析」, 情報処理学会 第 231 回自然言語処理研究会, 9 pages, May 2017.

6. 吉川将司, 能地宏, 松本裕治, 「係り受け構造との同時予測による A* CCG 解析」, 言語処理学会第 23 回年次大会 (NLP2017), pp. 274–277, March 2017.

7. 吉川将司, 進藤裕之, 松本裕治, 「話し言葉の依存構造解析における音声認識誤りの影響と評価」, 言語処理学会第 22 回年次大会 (NLP2016), pp. 477–480, March 2016.

Awards

1. 優秀賞, 言語処理学会第 23 回年次大会 (NLP2017), 2017（主著論文）

2. 若手奨励賞, 言語処理学会第 25 回年次大会 (NLP2019), 2019.（主著論文）

3. 優秀賞, 人工知能学会第 33 会全国大会 (JSAI2019) 口頭発表部門, 2019（共著論文）