Neural Approaches for Syntactic and Semantic Analysis

Shuhei Kurita

February 2019

Abstract

Natural language processing is an essential technology for systems that understand human languages and communicate with us. In natural language processing, computational systems grasp the structures and meanings of the natural language texts with syntactic and semantic analyses. These analyses are often called the fundamental analyses on the texts and related to the natural language understanding problems. The performances of the fundamental analysis directly affect the following models and applications. Therefore, there is great demand for the high accuracy analyses of textual inputs from both the academic and social requirements today. However, syntactic and semantic analyses of texts are quite challenging problems because of the complexity and ambiguity of texts. First, the structures of texts are complex and therefore several analyses are applied at the same time. Analyses models often take inputs from other models, which often cause the input biases and errors. Second, the meaning of the natural language texts are sometimes ambiguous and the external knowledge is required to grasp the meaning of them. Therefore, models for syntactic and semantic analyses often require the external knowledge that is hardly extracted from the limited annotated corpora. Third, the existing models of the fundamental analyses often rely on the arbitrary structures of the models that are designed for specific domains of datasets. The model developers often employ detailed feature engineering at the sacrifice of the generality and freedom of the model representations. This sometimes causes the overfitting of the models for limited domains.

Neural networks become common utilities in the last decade in many studies with machine learning. This is because neural networks have strong abilities of representation learning thanks to a large number of their internal parameters. In natural language processing, however, simple supervised neural network models suffer from the difficulties of the fundamental analysis of natural language texts. They suffer from the input

biases and errors from the preceding models. Tasks of the fundamental analyses often require the external knowledge, which neural network models that are trained from limited annotated corpus cannot deal with. To solve these problems, we propose new neural network-based approaches with three different approaches of the joint models, generative models and reinforcement learning. Especially, we focus on the following three problems of the syntactic and semantic analyses of natural language texts in this thesis.

The first problem is the joint syntactic analysis of Chinese sentences. Syntactic analysis of Chinese consists of word-segmentation, POS tagging and dependency parsing. Since POS tags and word dependencies can be determined after word boundaries are determined, models for these tasks are applied one by one and thus have a hierarchy. In a naive approach, models are combined and used as a pipeline. However, this inevitably causes the error-propagation from the preceding models. We explore neural network-based joint models for these tasks to prevent the error-propagation problem. Joint models are variations of the multi-task models in which tasks have hierarchical combinations. We also introduced the distributional representations of character strings in addition to the existing word and character representations. Distributional representations of character strings are strong ways to represent the meanings of incomplete words and therefore suitable for joint models. In our experiments, our models surpass existing models in Chinese word segmentation and POS tagging, and achieve preferable accuracies in dependency parsing. To avoid detailed feature-engineering, we also explore biLSTM models with fewer features. This model is competitive with existing models in terms of accuracy.

Second, we focus on the tasks between syntactic and semantic analysis: Japanese case analysis and zero anaphora resolution. These tasks are based on Japanese predicate-argument structure (PAS) analysis. In these two tasks, zero anaphora resolution is known as a difficult task because this task largely relies on the external knowledge. Such external knowledge cannot be covered with a limited existing annotated corpus. We propose a novel Japanese PAS analysis model based on adversarial training with an unannotated corpus. This idea comes from the recently proposed generative models of the generative adversarial network (GAN). Our proposed model enables to extract the external knowledge with generative approaches. In our experiments, our model outperforms existing state-of-the-art models in Japanese PAS analysis.

Finally, we conducted the semantic analysis of sentences in forms of the semantic

dependency parsing (SDP). In Semantic Dependency Parsing (SDP), semantic relations of words are expressed in directed acyclic graphs. We propose a new parsing algorithm for extracting semantic dependency graphs with both supervised learning and reinforcement learning. Using reinforcement learning, the model can choose the ways of creating the resulting graphs during training. In our experiments, our model achieves a new state-of-the-art performance on the semantic dependency parsing. In addition, we observe that our model trained with reinforcement learning takes an easy-first strategy.

In Chapter 1, we briefly introduce the fundamental analysis tasks of natural language processing. In Chapter 2, we introduce the joint syntactic analysis model with neural networks. In Chapter 3, we explain the case analysis and zero anaphora resolution model. This work is a combination of syntactic and semantic analysis. In Chapter 4, we introduce the semantic dependency parsing model with reinforcement learning. In Chapter 5, we present the conclusion and future work of our studies. We also present the future views of neural network and semantic studies.

Acknowledgments

I would like to express sincere appreciation for Professor Sadao Kurohashi and Professor Daisuke Kawahara for their supervision of this entire thesis and our whole work. I joined Kurohashi-Kawahara laboratory as a PhD student after I finished my master course work in the physics department, Kyoto University. Professor Sadao Kurohashi and Professor Daisuke Kawahara greatly supported me to work with natural language processing. I sincerely appreciate them.

I would like to express great appreciation to Professor Anders Søgaard at the University of Copenhagen for the supervision of our joint work for semantic dependency parsing with reinforcement learning. I have visited Copenhagen, Denmark for three months. I really appreciate people who have supported my visit.

I would like to express appreciation to Professor Masataka Goto for the JST ACT-I project and Professor Yutaka Matsuo for the supervision of my ACT-I project with Professor Sadao Kurohashi. I would like to express sincere appreciation to JST and all people who work with my ACT-I project. Given the ACT-I grant support, I was able to visit Professor Anders Søgaard for the semantic dependency parsing work. I would like to express appreciation to the JST CREST project of Professor Sadao Kurohashi, which I have worked as a research member throughout my PhD course.

I would like to thanks to Professor Tomohide Shibata and Professor Yugo Murawaki for the supervision and discussion of the broad NLP, neural networks and linguistic studies. Professor Tomohide Shibata gave me helpful comments on the Japanese PAS analysis studies. Professor Yugo Murawaki gave me great advice on machine learning and NLP.

I feel great appreciation to Fabien Cromieres, Toshiaki Nakazawa, Chenhui Chu and Raj Dabre. They worked for machine translation. Fabien Cromieres developed Kyoto neural network-based machine translation (aka KNMT), which stimulated me well in the

first year of my PhD. Chenhui gave me precious advice for Chinese syntactic parsing. Raj is my favorite friend and I'm grateful for the discussion of the neural end-to-end models such as Google's transformer.

I would like to appreciate to Hajime Morita, Yuta Hayashibe, Tetsuaki Nakamura, Teruaki Oka, and Ribeka Tanaka. They also give me precious advice on NLP and Lingistics. Hajime Morita developed the recurrent neural network-based Japanese morphological analysis model, which stimulated me well.

I would like to gratefully thank you to Yuko Ashihara and Naho Yoshitoshi. They worked as secretaries of our laboratory. I stand in awe of their great support.

I feel sincere appreciation to Gong-Ye Jin, Shen Mo, and John Richardson. They were always good senior PhD students. They succeed in their own ways after they got their PhDs. During my PhD course, their success always stimulated me.

During my PhD student course, I have met many talented and skilled young engineers and researchers. I sincerely appreciate to Tomohiro Sakaguchi, Tolmachev Arseny, Yudai Kishimoto, Xun Wang, Huang Yin-Jou, Tariq Alkhaldi, and Wataru Sakata. They work for their PhDs and are always good colleagues, rivals and friends. We all love NLP, neural networks and future technologies. I met many talented master and bachelor students during my student-hood. I hope all of them succeed in their fields.

I would like to express deep and sincere appreciation to Professor Masataka Dantsuji at Kyoto University, Professor Shigeru Shinomoto at Kyoto University, Professor Ryota Kobayashi in NII and Professor Katsunori Kitano at Ritsumeikan University. They helped me before and during I joined this laboratory and I always feel appreciations for them. I also express sincere appreciation to two senior PhD students of Yasuhiro Mochizuki and Masahiro Ueda in my old major. They are always serious at researching. I felt great appreciation to other old colleagues, some of which achieve great studies and some of which have already gone missing. I hope they are still alive and succeed in their own fields.

I would like to thanks to my family and friends for their continuous supports and general discussion during and before I achieve this work.

The research of natural language processing and artificial intelligence have evolved drastically in these years. I succeeded in my PhD but this is due to not only my skills and efforts but also the great changes of these social movements towards machine intel-

ligence, neural networks, and natural language processing. In this sense, I sometimes found myself even “lucky”. And so from old faithful researchers as good as forgotten in the twice “AI” winters, came young students as numerous as the stars in the sky and as countless as the sand on the seashore to study machine intelligence. I studied the C language by myself before I entered middle school just because I wanted to create “AIs” with silly thoughts. Now they came out and went into the society, and the whole young researchers let neural networks rush down the steep bank into the minima and the society follow in a muddle. I’m not sure where these social rushes toward “AI” will reach for in the future. One thing sure is that our experiences, efforts, skills and the great community of NLP would be beneficial in any circumstance. I would like to appreciate the great chance in Kurohashi-Kawahara laboratory.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural language processing (NLP) is a crucial technology that allow us to grasp the structures and meanings of natural language text and handle it in computational systems. NLP plays a crucial role in systems that communicate with humans (aka *human-computer interactions*, HCI) because languages are natural communication tools for us. In NLP, syntactic and semantic analyses for natural languages texts are essential technologies for systems that resolve structures of texts and grasp the meanings for the following applications. They are also closely correlated to *natural language understanding* (NLU) problems. Syntactic and semantic analyses of natural language texts are tough and quite challenging problems. Linguistic structures of texts often become complicated to be extracted but they are so important that they have been frequently applied to application systems. Their meaning is sometimes ambiguous and some knowledge of the external world is required to grasp the meaning of texts, which is difficult to be extracted from annotated corpora.

Recently, neural networks have been widely applied to many NLP models including syntactic and semantic analyses. However, the syntactic and semantic analyses of natural language texts are still difficult because of the complexity of texts we mentioned above. Simple models of supervised neural networks face serious difficulties of natural language texts in these tasks of NLP. In this thesis, we propose the new neural approaches for the syntactic and semantic analyses of texts to resolve the complexity of the textual structures and meanings.

1.1 Introduction to NLP

The purpose of natural language processing (NLP) is to understand the meanings of natural language texts and provide some methods for computational systems to handle texts in some application systems. The subjects of the NLP studies are so broad: from grammatical analyses of texts to natural language understanding and application systems such as machine translation systems and chatbots.

In the early era of the NLP research history, many studies employed rule-based approaches. They relied on the hand-written rules that mainly come from linguistic viewpoints and dictionaries for processing texts. Since linguistic phenomena are so complex, many hand-written rules must be continuously updated with efforts. After that, many studies have adapted the data-driven and learning-based approaches. They tend to use some linguistic datasets or *corpora* with *machine learning* models. For the last decade, neural networks become the common tool of the machine learning models in NLP. We will introduce them in Sec. 1.3.

Recently there is a great demand for high-performance NLP models from academic and business requirements. One reason for this is that more and more text data become available thanks to many Internet services such as web blogs and social networking services. The applications of the NLP technology would expand in our society in the next several decades.

1.1.1 Fundamental Analyses and Application Studies of NLP

In NLP studies, syntactic and semantic analyses of texts are often called fundamental analyses. The fundamental analyses includes the grammatical analyses of sentence such as word segmentation, POS tagging, chunking and syntactic parsing. Grammatical analyses hold a prominent position in the fundamental analyses and have a long history in NLP studies. Semantic analysis such as semantic parsing is another large part of the fundamental analyses and getting more and more important in natural language understanding. We note that these analyses often depends on each other. For example, the word segmentation has critical effects on the following analyses because words are the conventional units of the meaning representations and many later analyses assume that input words are correctly segmented as a matter of course.

In contrast to the fundamental analysis, studies for some applications systems such as machine translation, question-answering, chatbots and searching systems are called application studies. In many applications system, the fundamental analyses plays important roles for analyzing input texts and grasp their meanings. The performance of the application systems greatly depends on those of the fundamental analysis models. This is because the analyses of input texts (e.g. input texts of machine translations or user queries for searching systems) are crucial for the following tasks in the application systems. We also note that some recent application systems sometimes take *end-to-end* framework that the systems take low-level inputs and directly outputs the final result without any other tasks or analyses. However, much of end-to-end application models still rely on other fundamental analyses models. Otherwise the whole application system often make severe errors due to the misunderstanding of the input texts. Therefore the fundamental analyses are necessary for many of application systems.

In this thesis, we focus on the fundamental analyses of natural language texts. As we explained, many NLP models and application systems rely on the syntactic and semantic analyses of texts. The performance of the following models and application systems directly depends on the preceding fundamental analysis models. Thus we cannot overemphasize the importance of the fundamental analyses.

1.1.2 Flows of Fundamental Analyses

We briefly introduce the flow of the whole fundamental analyses. natural language texts have structures: characters, words, phrases, sentences, paragraphs and documents as is shown in Fig 1.1.

Characters	Words	Phrases	Sentences	Paragraphs
c, a, t	cat	the cat	The cat sat on the mat.	The cat sat on the mat. Her eyes are closed, but her tail waves slowly...

Table 1.1: Structures of Languages.

Tasks in the fundamental analyses are based on these textual structures. Some of the fundamental analyses tasks, such as word segmentation and chunking of texts, aim to

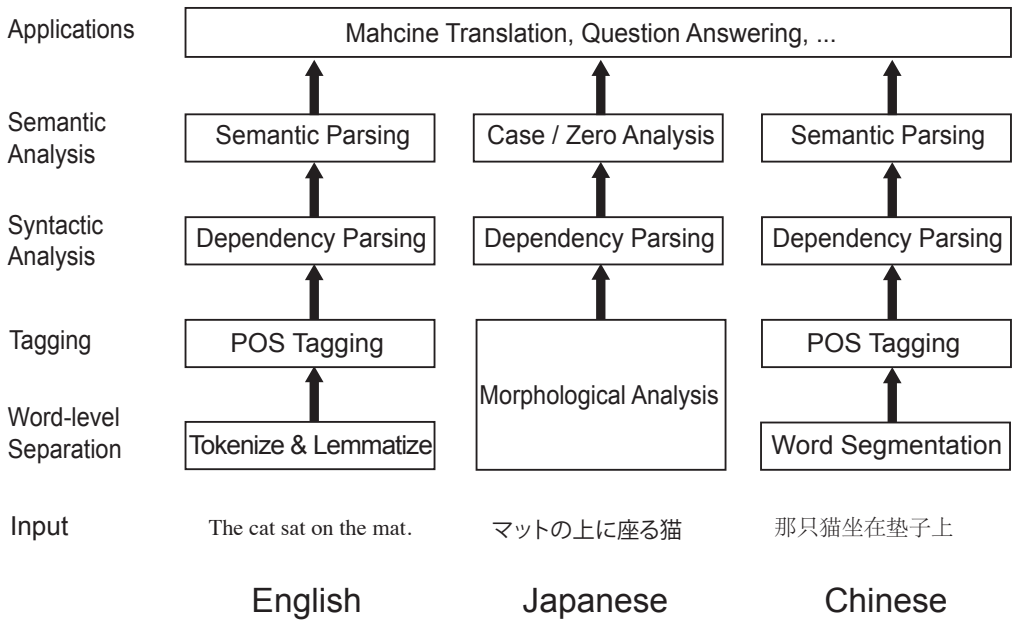

Figure 1.1: Difference of the applied fundamental analyses in English, Japanese and Chinese. In Japanese and Chinese, the word segmentation is the essential problem for many upper tasks and applications.

reveal structures of such texts and tokenize them. Other tasks such as POS tagging and parsing aim to analyze functional and semantic roles of tokens.

For details, the applied fundamental analyses depend on the languages. Fig 1.1.2 shows the difference of the fundamental analyses applied for English, Japanese and Chinese. The major difference lies in the basic analysis. For all listed languages, inputs are character sequences. [1] *Tokanization*, *Lemmatization* and *POS tagging* is used in basic analyses of English texts, while *word segmentation* is needed for Chinese texts. In Japanese, the corresponding task is the *morphological analysis*.

In the fundamental analysis, tasks have a hierarchical structure as is shown in Fig. . We often call this sequence of processing as the *pipeline model* because they are often

[1] We also note that realistic application systems often need additional preprocessing of input texts such as removing HTML tags, converting special characters such as emojis that are not used in analyses into some other symbols and dividing texts into suitable lengths of sentences. Preprocessing of texts often plays significant roles in real NLP applications although we do not discuss this further in this thesis.

combined with the pipeline of the UNIX shell. In Sec 1.2, we review the problems of this naive pipeline scheme.

We show the detailed flows of the fundamental analyses in Japanese, English and Chinese in turn.

Fundamental Analyses in Japanese

Fig. 1.2 show the flow of fundamental analyses in Japanese with an example sentence of "その男は引き返してデスク係に話しかけた。" (The man went back and spoke to the desk clerk.) In Japanese, the input texts of character sequences are firstly given to *morphological analysis*. The morphological analysis is a collection of tasks of word segmentation, *lemmatization*[2] and labeling of word roles for Japanese texts. The accuracy of word segmentation in Japanese is beyond 99%. Examples of the Japanese morphological analysis tools are juman and juman++ [6].

Dependency parsing is one of the most sophisticated analysis in the syntactic analysis of the sentences. Dependency tree express grammatical relations of words in a form of trees. In *Semantic parsing*, various semantic relations of words or phrases are analyzed. In dependency parsing of Japanese, texts have an interesting rule that a word must depend to another word that is placed in right of the original word. The example of dependency parsing tool is KNP [7]. In semantic parsing of Japanese, case analysis and anaphora resolution are often applied. We will discuss them in Sec. 1.1.3.

Fundamental Analyses in English

Fig. 1.3 shows the basic flow of the NLP analyses in English. For English, the input texts are generally easy to be divided into words because they are written with word separation. *Tokenization* is used for dividing some tokens such as "She's saying" into "She", "'s" and "saying". *Lemmatization* is used to obtain the basic form of word, e.g. "she", "be" and "say" in this example. Lemma is sometimes required in the following tasks. *POS tagging* is a task to find syntactic role labels for words. We will explain this in Sec. 1.1.3. POS tags are so crucial that the performance of the most of following NLP

[2]Lemmatization is a task to find base forms of words. It is sometimes applied for languages that have conjugations of words.

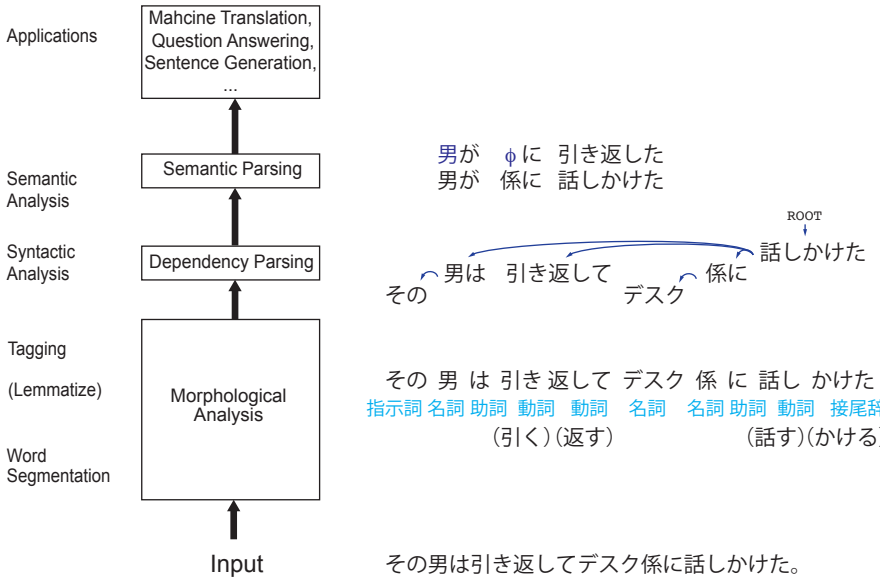

Figure 1.2: Flow of the fundamental analyses in Japanese. The examples of analyses for a Japanese sentence is presented in the right of the figure.

tasks are enhanced with this word role information. For the following analysis, syntactic parsing and semantic parsing are applied as with the Japanese analyses.

Fundamental Analyses in Chinese

Fig. 1.4 shows the analyses in Chinese with an example Chinese sentence of “技术有了新的进展。”(Technologies have made new progress.) In Chinese, word segmentation is known as a quite difficult problem and its accuracy is still low, which cause the low accuracy of dependency parsing results.

1.1.3 Fundamental Analysis: Tasks

We will briefly introduce some of the fundamental analysis tasks that are quite important and frequently used in many NLP systems. We also briefly explain the existing methods for these tasks.

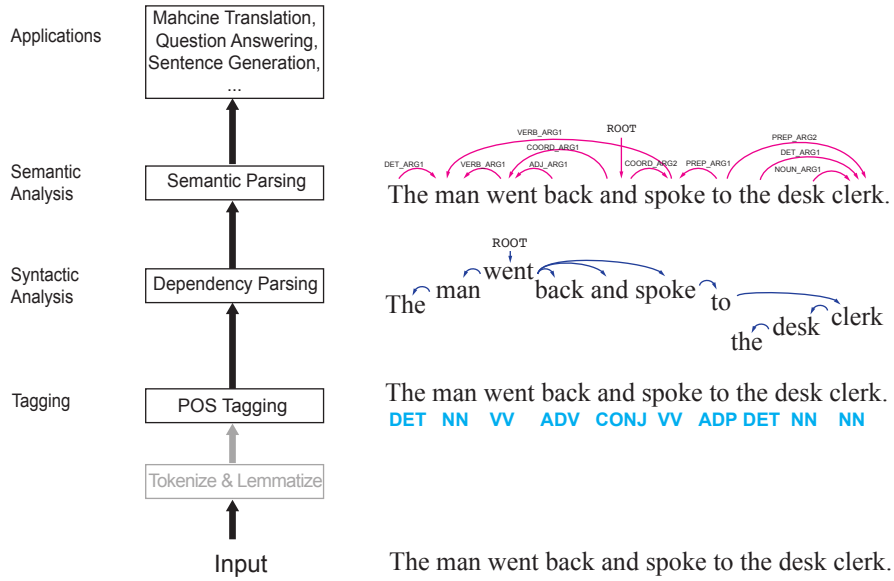

Figure 1.3: Flow of the fundamental analyses in English. An examples of analyses for a sentence "The man went back and spoke to the desk clerk." are represented in the right of the figure.

Word Segmentation

Word segmentation is a task to determine word boundaries for character sequences. From a naive viewpoint, it is said that word segmentation is a character-wise classification task of whether one character in sentences is the beginning of some words or not. Word segmentation is an essential task in some languages such as Chinese and Arabic although the performances of the word segmentation in those languages are not always enough. In Japanese, the corresponding task is the morphological analysis that includes the attachment of syntactic role tags and lemmatization. Many following NLP tasks rely on the resulting words from this task. Therefore the value of the word segmentation tasks cannot be overemphasized.

Sequence labelling for characters with in-and-out-of-word labels are often used for the word segmentation. Such models often rely on the dictionary matching features.

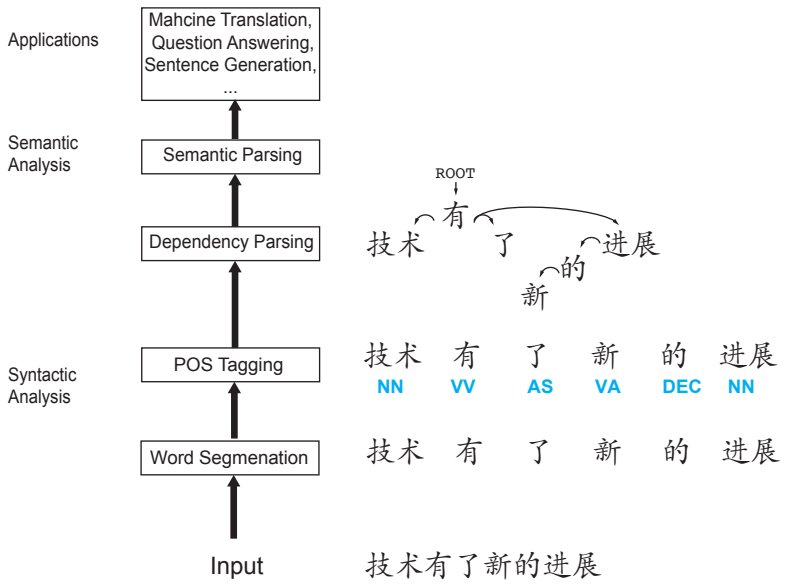

Figure 1.4: Flow of the fundamental analysis in Chinese. In Chinese, some of basic analysis such as word segmentation is quite difficult. This is a one of the major reasons that Chinese NLP is know as difficult.

POS Tagging

POS tagging is a classification task for the syntactic roles of words in sentences. Examples of POS tags are *noun*, *verb*, *determiner*, *adjectives*, *adverbs* and so on. The number of POS tags that are used in one model differs depending on linguistic formalisms. In POS tagging, some words are used only for limited syntactic roles (e.g. English words "emphasis" and "emphasize" can be used for *noun* and *verb* respectively.) and they are easily classified to the corresponding word-role categories. The conjugation of words is also helpful for this task. Otherwise, the roles of words depend on the surrounding contexts. Chinese POS tagging is relatively difficult because of the lack of the conjugation and many words can be used for several syntactic roles depending on their contexts.

Classically, sequence labelling models with inputs of n-gram words and previously-predicted POS tags are often used for this task. Since POS tags are crucial in many NLP applications, many languages have the corresponding tasks with the word-role labelling.

Dependency Parsing

From a grammatical viewpoint, each word in a sentence has one syntactic dependency to another word in the same sentence or a single root entity. [3] Dependency parsing is a task to create syntactic dependency trees that are spanning among all words in the sentences. The dependency trees demonstrate the global and local structures of the sentence.

There are two major ways of dependency parsing: graph-based and transition-based parsing algorithm [8]. Graph-based parsers have a triangle table to store dependency scores from some words to other words. The model firstly assigns all scores in the table and then decode it into a tree with external algorithms such as Kruskal's algorithm for the maximum spanning tree [9]. The example of the graph-based parser is KNP [7] in Japanese. Transition-based parsers, however, process the sentence from the first word to the last word creating and updating partial trees. Transition-based parsers are fast but sometimes suffer from serious error-propagation problems.

Dependency parsing is one of the most important tasks of syntactic analyses and its annotated datasets have been created in many languages. However, the annotation costs for the dependency parsing are relatively heavily in general. Therefore, the available datasets for dependency parsing are less than those of word segmentation and POS tagging.

Case Analysis and Zero Pronoun Resolution in Japanese

Words in sentences have complex relations with other words. Especially, semantic relations of predicates and their argument candidates, such as subject-verb, verb-object and verb-dative relations are important analysis although they are not always presented in the dependency parse trees. The analysis to find out predicate and their argument relations are often referred to the predicate-argument structure (PAS) analysis. PAS analysis lies between the syntactic and semantic analyses.

Case analysis is one form of Japanese PAS analysis. The case analysis is a task to resolve the semantic relation of some predicates and their arguments when they have

[3] The root entry is a conventional symbol that is out of the sentence and used for the root node of the grammatical tree. Each of syntactic dependency parse trees includes only one edge from the root node to some word in the sentence in general.

direct dependencies but their semantic roles are not obvious. Therefore the case analysis largely depends on the preceding syntactic analysis.

Zero pronoun resolution is a task to find out the omitted arguments of the predicates. Japanese texts have many omitted arguments in each predicate such as the writer or speaker expressions. They are omitted because the writer of the sentence considered that they are obvious or redundant in this context. Such omitted arguments are called zero pronouns and some application system such as machine translation often suffer from the zero pronouns in Japanese. In zero pronoun resolution, the argument candidates do not have direct dependencies to the predicate. The model finds out the correct zero pronouns for case roles of predicates from candidates of zero-pronouns from the in and out of sentences and exophora in some cases. Therefore, zero pronoun resolution is independent of the syntactic analyses to some extent and a quite difficult problem.

Semantic Dependency Parsing

Semantic Dependency Parsing (SDP) is a task to create a graph of the word semantic relations. Unlike dependency trees, the SDP forms direct acyclic graphs, not trees and therefore SDP expresses the complex relations of semantic predicates and their arguments. SDP is a relatively new task, introduce in SemEval2014 [10], derived from previous several semantic dependency formalisms including the predicate-argument structure (PAS).

Since graphs of SDP express complex word relations, the parsing algorithms for syntactic dependency structures often cannot deal with them or become inefficient. The graph-based parsers require an external threshold to determine whether the two words have an edge given the dependency scores. Transition-based parsers need complex and arbitrary transitions for SDP.

1.2 Challenges in NLP Tasks

Many of the fundamental analyses in NLP have several difficulties. Such problems characterize the machine learning models used in NLP. Many of problems come from the limited training data for the fundamental analysis. The training data for the fundamental analysis of natural language texts are extracted from the annotated corpora that are cre-

ated by human annotators. Especially for the dependency parsing and semantic analysis in the fundamental analyses, models require detailed annotations and hence the cost for creating annotated corpora become huge. The amount of sentences in current datasets used in the fundamental analyses is typically tens of thousands or less. They are too small to extract the useful knowledge of the external world from these corpora. It is also relatively small if you train the deep learning models with them.

Another reason for the difficulties is that NLP models are often used with other models as so call pipelines. Since most NLP models are frequently used with other models, they take inputs that contains inevitable biases and errors from other models. This is one of the most important characteristics of NLP models. We will explain those differences between NLP models and machine learning models with an example of the POS tagging models and the hand-digit recognition models. Both of them are classification tasks. The models of hand-digit recognition are trained and evaluated with the specific domain of the dataset, such as MNIST [11], which consists of hand-writing digits images. The outputs of such models are uncommonly combined with other machine learning-based models. In POS tagging for words, however, the model inputs often come from another word segmentation model. The POS tags are often used as inputs of the following models. In this sense, the classification problems in NLP studies are not a simple machine learning classification problem.

Parsers of sentences have also unique problems. Transition-based parsers internally use some classifiers that are similar to those used in simple classification tasks. However, the transition-based parsing of sentences is not a simple classification task at all. Rather it is similar to a sequence of the decision making problems that are sometimes too difficult to be trained with the supervised learning because of the unseen states during the test phase. The transition-based parsers frequently face the unseen states during the test phrase because of the errors of the previous decisions or unfamiliar inputs. Graph-based parser have their own problem that the machine learning models of the parser just determine scores for all edges and rely on some external algorithms to decode such scores into trees.

In this thesis, we employ the neural network-based approaches for those difficulties. In experiments, we apply them for several tasks in the fundamental analyses of NLP. However, the much of difficulties are widely shared among NLP tasks and our approaches

can be applied for many tasks. Therefore we assume that our proposed approaches can be applied for many other fundamental analyses systems. We summarize the major problems of the fundamental analyses in NLP below.

1.2.1 Limited Knowledge of The External World

The tasks of NLP often depends on the external knowledge. External knowledge is the information that are not specified in the limited annotated corpus, but used in some tasks implicitly. For example, Japanese verb "焼く" is used for *baking or burning something*. However, in a phrase of "CDにデータを焼く (burning data to a CD)", "焼く" is used for writing out data on a CD. In another phrase of "人に手を焼く (being troubled with some persons)" is an idiom that is used for cumbersome persons or children. In the case of "焼く", its meanings are altered by surrounding words. Such knowledge is not directly specified in the training corpus in general. Therefore the models need to learn these expressions from other knowledge resources.

1.2.2 Error Propagation

Since fundamental analyses in NLP have a hierarchical structure, some inputs of models come from the outputs of the preceding models. Therefore, the minor errors of the preceding models can spread to following models and spoil the entire performance of the system. This is known as the *error-propagation* problem. To avoid the error-propagation problem, the joint model is often used in traditional NLP tasks. The joint model is a kind of the multi-task model that jointly processes the stacked tasks. When the stacked tasks have strong correlations, joint models become strong tools for analyses.

1.2.3 Arbitrary Model Structure

Natural languages have specific internal structures. In existing models, model developers determine what kind of models should be applied for the analyses considering the NLP task structures with empirical ways. Such models are typically easy to be trained because of the pre-implemented strong prior for the task. However, such models often lose the generalities and are extensively tuned for specific domains. We summarize the pros and cons of empirically-tuned models in Table 1.2.

Pros	Easy to train because they have less degrees of freedom. Model size is relatively small because of less trainable parameters.
Cons	Less performance in generative models because the strong prior prevent from the learning from the dataset. Easy to overfit to some specific domains.

Table 1.2: Pros and cons of the empirically-tuned model.

In pre deep learning era, many researchers relies on extensive *feature engineering* of machine-learning models. They tend to use *Support-Vector Machine*, or SVM [12] and *averaged perceptron*, a variant of single layer perceptron that is often used as well as SVM in NLP for classifiers [13]. The feature engineering is a tuning of input features for some specific tasks, which often cause severe preset bias and over-fitting for the some limited domains of datasets. Neural network contributes to reduce the extensive feature engineering because some of feature combinations are represented in the hidden layers. Hence, model developers in modern era do not need to tune input features extensively.

In this thesis, we propose the neural network-based NLP model that can reduce the overused feature-engineering. We also explorer the procedure that the model itself is able to determine the way to analysis input sentences.

1.3 Neural Networks in NLP

Before we explain the proposed models in the next section, we briefly review the history of neural network studies and the recent advancement of deep learning in natural language processing in this section. Neural networks have been widely used in these years in many data-driven science fields including computer-vision, audio recognition, robotics, material science and NLP. However, from the early history of neural network studies, semantic analysis of natural language texts are under great interest.

1.3.1 Neural Networks and Distributional Representations

From a viewpoint of symbolics, anything that are recognizable and thinkable for human beings such as objects, persons, physical states, abstract ideas, images, sounds and words are represented with symbolic methods such as binary or sparse vector representations.

In late 80's, J. E. Hinton and others proposed the novel *distibutional representation* [14]. Their idea was that any entities leaned by models are represented with distributional units and their connections. These network of units and their connections are known as *neural networks*. They visualized the wights of their neural network connections with *Hinton diagram*, which consists of white and black rectangles that corresponds to the strength of excitatory and inhibitory connections of neurons. They showed the distributional representations of persons learned from family tree datasets and some features are shared among persons in their *semantic net*. D. E. Rumelhart also coined the term *back-propagation*. Back-propagation is the gradient decent that is applied to multi-layer neural networks using the chain-rule of gradients [15].

From a philosophical viewpoint, they are often called as *Connectionist* in contrast to symbolic approaches. [4] We also thanks to the very early work of S. Amari for the work of the differentiable neurons [16] derived from neurophysiological and mathematical studies.

Two large streams of the structures of neural network had been proposed: recurrent neural network (RNN) and convolutional neural network (CNN). RNNs exist from the early studies of neural networks [15, 17]. In RNNs, the computation graph has loops that allows the outputs of some layers can become the inputs of the same layers in later time steps. The RNNs can store information in these loops. The back-propagation for RNNs can be applied thought the time sequence. In this sense, neural network with simple loops is very deep in terms of the time sequence. Simple RNNs had a problem of forgetting old inputs because old inputs are easily overwritten by new inputs. Long-short term memory (LSTM) [18] was proposed by adding several gates to input, memory and output layers to reject unnecessary input or histories and sustain critical information. Y. Lecun proposed the *LeNet* that exploits the neural convolution layer and sub-sampling [19]. They used their network on the hand-writing character recognition task. We also

[4] The people who believe in bayesian statistics are often called *Bayesian*, which is in contract to those of classical statistics as *Frequentist*.

thanks to *neocognitron* of K. Fukushima as an original form of CNN [20].

1.3.2 Neural Approaches in NLP

Although the early work of neural networks focused on the representations of *semantics*, the early studies of text processing relied on symbolic and rule based approaches mainly because of the processing speed and amount of the available datasets. They mainly relied on linguistic or statistical approaches. However, there were also well-known NLP models driven by neural networks. R. Collobert and J. Weston proposed a CNN-based fundamental analysis models of NLP [1]. They applied a single convolution layer and max-pooling through time. They used a lookup table for transforming word indices to hidden representations of words, or the *embedding* of words. They also applied the multi-task learning of many tasks: semantic role labeling (SRL), part-of-speech (POS) tagging, chunking, named entity recognition, synonymous and language modeling. In their multi-task neural network, one of lookup tables was shared and none of hidden layers were shared. We note that in many recent models make use of pre-trained word embeddings instead of multi-task learning of embeddings as of Collobert and Weston [1]. As we explain in Sec. 1.4.1, modern multi-task neural networks exploits the structured networks for different tasks.

1.3.3 Mikolov's word2vec

In 2013, Mikolov et al. [21] proposed the well-known *word2vec*, which is an unsupervised learning tool to obtain distributional representations of words from unannotated corpora. They proposed the continuous bag-of-words (CBOW) and skip-gram learnings. CBOW or continuous bag-of-words predicts a single word from contexts of neighboring words. CBOW uses the previous and following words to predict a single word. Therefore CBOW is not a pure language model.[5] Skip-gram is *vice versa*: predicting neighboring words from a single word. CBOW seems an easier task than the skip-gram is because CBOW is a model of predicting only one word. They reported that skip-gram was better than CBOW in terms of semantic representation of words, while CBOW was better in

[5] A language model is a model for repeatedly predicting a following word given the preceding words as contexts to generate some sentences.

syntactic representations. They also claimed that their model had the basic understandings of the external world. For example, a *king* is male while a *queen* is female. *Tokyo* is the capital of *Japan* while *Paris* is the capital of *France*. Therefore their learned vectors had the relations of:

$$\begin{aligned} \mathbf{v}(queen) &\simeq \mathbf{v}(king) - \mathbf{v}(man) + \mathbf{v}(woman) \\ \mathbf{v}(Tokyo) &\simeq \mathbf{v}(Paris) - \mathbf{v}(France) + \mathbf{v}(Japan) \end{aligned}$$

, where $\mathbf{v}(\cdot)$ represents the learned vectors of words. This is called the word analogy task. After their work was published, several word embedding models have been proposed such as Glove [22]. Glove implicitly exploits matrix factorization, claiming better accuracies in the word analogy task than word2vec.

1.3.4 RNN, CNN and Attention-based Neural Networks

Word embeddings are sophisticated representations of words. Using word2vec, systems can obtain some pre-trained word representations. However, such representations for words are independent to their contexts when they are used in some sentences. Therefore, another important question occurs: *what are the best representations for words, phrases or sentences when we encode texts?* Representations for words in sentences should be changed depending on their surrounding contexts. Representations for phrases and sentences should contain enough information for the following tasks and therefore they would be larger than those for words. Neural networks to obtain sentence representations are often called the *sentence encoder*. After the encoder calculate representations of words, phrases or sentences, such representations are used in the following parts of the entire neural network.

R. Socher proposed *recursive neural network*, which is a recurrent neural network constructed through dependency graphs of sentences that are extracted from pre-processes [23]. Their neural network is a syntax-based encoder. K. Yoon proposed a CNN-based model for sentiment analysis of sentences such as movie reviews [24]. The problem of their model is that their model ignores the positions of the words in the sentence. In sentiment analysis of short sentences, the positions of words are not always critical features and the resulting representations for sentences are suitable for this task. However, positions of words are often quite important in the fundamental analyses of NLP.

In 2014, Bahdanau et al. [25] proposed the fully neural network-based end-to-end machine translation model. The model consists of the RNN encoder and the decoder of the RNN-based language model as the sequential word prediction. This model is called the neural machine translation (NMT) compared with the previous statistical machine translation (SMT) or the example-based machine translation [26]. NMT models rapidly expel the previous models while they often suffer from the ambiguity of the input sentences.

RNNs and their variations are frequently used for the sentence encoders now. However, RNNs have problems in the training speeds. The combination of CNN and RNN is also attempted to improve the analysis in terms of the speed and accuracy [27]. The LSTMs, GRUs[6] and their stacked and bi-directional variations are frequently applied for RNN-based sentence encoding. Another recently proposed encoder is the attention-based neural network encoder aka Transformer of Google [29].

1.3.5 The Limitation of Supervised Learning

Most of the neural network-based models succeeded in specific tasks with limited datasets using single-task supervised learning. In many studies, researchers propose a new form of neural networks that perform well in some limited dataset domains with task-specific supervised learning. This sometimes suggests that the model overfits to the domains of datasets that the experiments are performed.

Supervised learning often cause the overfit to specific domains. It could also happen that researchers select arbitrary models that works in specific domains. *Multi-task learning of neural network* prevent models from overfitting specific task domains. *Semi-supervised learning* exploit unlabeled datasets that contains the external data structures or knowledge that can not be covered in the limited training datasets in supervised learning. *Reinforcement learning* allows models to search some hidden structures in the external environment through the exploration. We explain the details of these learning methods for NLP problems in the next section.

[6]Gated recurrent unit was firstly proposed in the conference of EMNLP2014 by Cho et al. [28].

1.4 Proposed Models

In the end of the introduction chapter, we explain the background of our proposed models and introduce the models and corresponding tasks we discuss in this thesis and why they are so challenging in NLP.

1.4.1 New Approaches in Neural Networks

We adapt several neural network models that have a high profile in recent deep learning and NLP studies. We briefly explain them.

Joint Neural Network

The idea of multi-task learning is old in NLP as is seen in Ronan Collobert and Jason Weston (2008). However, the modern form of hierarchical multi-task learning neural network was introduced in Søgaard and Goldberg [30]. They proposed a deep neural network with hidden layers for different tasks. They used the hidden representations from a lower layer for POS tagging and those from an upper layer for chunking. This corresponds to the fact that the chunking is an upper task compared with POS tagging in English NLP. Fig. 1.5 briefly presents the differences of these two models.

The multi-task neural network is superior to the single-task neural network in terms of the generalities of the learned representations. However, the information from other tasks become strong regularization to parameter tuning for a single task. If tasks have strong correlations such as joint models of NLP, the multi-task neural network become a great solution.

Generative Models

Generative adversarial networks (GAN) is firstly introduced in Goodfellow et al. [31] for the image generation task. This is a combined neural network of the generator neural network and its adversarial counterpart as the discriminator neural network. The generator outputs images that is similar to real images while the discriminator attempts to discriminate the generated images from real images. This is a mini-max game of two parts of the neural network. We note that the mini-max game based GAN training is sometimes unstable and combined with supervised learning.

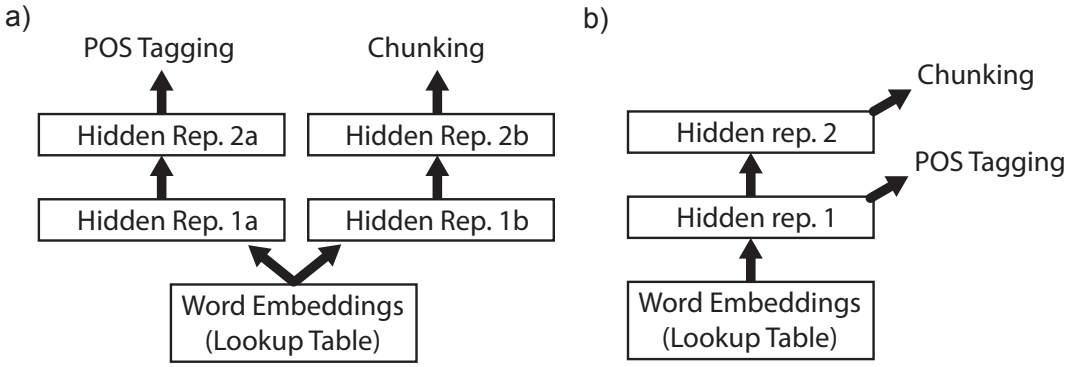

Figure 1.5: The difference of models in (a) Collobert and Weston [1] and (b) Andor et al. [2]. Note that each of models employs the state-of-the-art architecture for their tasks at that time. Collobert and Weston [1] uses CNNs and multi-layer perceptron (MLP), while Andor et al. [2] employs stacked-LSTMs.

Many researchers have attempted to apply GANs for the sentence generation task in NLP, but they still face difficulties. This is because the sentence generation is a task for the temporal sequence generation for words. The generated words are symbolic. Therefore, one the model generate some words, they stop the gradient computation of the neural network computation graph when they are used in the following word predictions. Although there are some possible solutions, for example, Wasserstein GAN [32], the sentence generation through GAN approaches is still difficult.

Another possible application of GANs is that employing GAN-like training on unlabeled corpora to extract the external knowledge from them. We will discuss this approach on this thesis.

Reinforcement Learning

Reinforcement learning is a training that an internal model, or often referred as an *agent*, repeatedly interacts with the surrounding environments via some actions and gets rewards from the environment. The agent attempts to maximize the rewards that the agent is going

to obtain in the near future.

There are several classifications of reinforcement leaning. The model that searches the transition space *off-policy* and *on-policy*. In on-policy learning, the agent explorers states depending on its internal policy while in off-policy learning the model samples states without its policy. An example of off-policy model is q-learning [33] and an example of on-policy learning is SARSA [34]. In deep reinforcement learning, the deep q-learning (DQN) is off-policy learning [35]. Policy gradient [36] is used for on-policy learning, although it is possible to use it in an off-policy learning manner. The *model-based* and *mode-free* reinforcement learning is used whether the decisions of the agent are based on the information of the environments. The model-based approaches use the information of their action, rewards and the observation of the environment, while the model-free approaches do not use the observation of the environment. In this thesis, we focus on the model-based and on-policy reinforcement learning.

In some NLP subtasks, the model processes sentences by repeatedly deciding the internal structures of the sentences. Here, reinforcement learning is the promising solution for these tasks. This is because the model can find out the optimal decisions to solve the tasks by themselves. In this thesis, we apply the reinforcement learning for parsers to allow them to choose the optimal parsing ways through the exploration.

1.4.2 Joint Syntactic Analysis with Neural Network

In the fundamental analysis, some tasks are closely correlated. Especially the word segmentation and POS tagging are closely correlated because the roles of words are limited and some POS tags become obvious when the word boundaries are determined. POS tagging and word dependencies are also closely correlated because words has dependency destinations that are limited depending on their syntactic roles. We discuss this joint syntactic analyses model on Chapter 2.

1.4.3 Neural Case Analysis and Zero-pronoun Resolution with Generative Approaches

The difficulty of the case analysis and zero-pronoun resolutions tasks is that they relies on the knowledge of the frequent combinations of predicates and arguments. This

knowledge is often referred as *selectional preferences* for predicates and their arguments. Selectional preferences is an subset of the external knowledge and are difficult to be extracted from annotated corpora because the selectional preferences are numerous while annotated corpora are limited. We explorer the models to extract selectional preferences from largely available unannotated corpora with the recent proposed generative adversarial networks (GAN) for this. We proposed the validator network instead of the discriminator network of GAN. The validator network validates the results of the extracted predicate-argument combinations from unannotated corpora. We discuss this model on Chapter 3.

1.4.4 Semantic Dependency Parsing with Reinforcement Learning

We study the reinforcement learning-based semantic dependency parsing model. Semantic dependency parsing (SDP) is a newly proposed semantic parsing in which the resulting semantic graphs compose directed acyclic graphs instead of trees. We propose the reinforcement learning based model in which the model itself is able to decide how to parse sentences through explorations. We discuss this model on Chapter 4.

Chapter 2

Neural Network-based Joint Syntactic Analysis

We present neural network-based joint models for Chinese word segmentation, POS tagging and dependency parsing. Our models are the first neural approaches for fully joint Chinese analysis that is known to prevent the error propagation problem of pipeline models. Although word embeddings play a key role in improving the performance of dependency parsing, they cannot be applied directly to the joint task in the previous work. To address this problem, we introduce distributed representations of character strings, in addition to words that are turned out to be effective for the joint task. Experiments show that our models outperform existing systems in Chinese word segmentation and POS tagging, and perform preferable accuracies in dependency parsing. We also explore bi-LSTM models with fewer features, that performs competitively with these existing models.

2.1 Introduction

Dependency parsers have been enhanced by the use of neural networks and embedding vectors [37, 38, 39, 40, 2, 41]. These dependency parsers are word-based. Therefore, when these dependency parsers process sentences in English and other languages that use symbols for word separations, they can be very accurate. However, for languages such as Chinese or Japanese that do not contain word separation symbols, dependency

parsers are used in pipeline processes with word segmentation and POS tagging models, and encounter serious problems because of error propagations from the previous models. Even for these languages, the previous dependency parser also work accurately when the given input sentences are well-segmented by words. However, this is not realistic in real usages. In particular, Chinese word segmentation is notoriously difficult because sentences are written without word dividers and Chinese words are not clearly defined. Hence, the pipeline of word segmentation, POS tagging and dependency parsing always suffers from word segmentation errors. Once words have been wrongly-segmented, word embeddings and traditional one-hot word features, used in dependency parsers, will mistake the precise meanings of the original sentences. As a result, pipeline models achieve dependency scores of around 80% for Chinese.

A traditional solution to this error propagation problem is to use joint models. Many Chinese words play multiple grammatical roles with only one grammatical form. Therefore, determining the word boundaries and the subsequent tagging and dependency parsing are closely correlated. Transition-based joint models for Chinese word segmentation, POS tagging and dependency parsing are proposed by Hatori et al. [3] and Zhang et al. [4]. Hatori et al. [3] state that dependency information improves the performances of word segmentation and POS tagging, and develop the first transition-based joint word segmentation, POS tagging and dependency parsing model. Zhang et al. [4] expand this and find that both the inter-word dependencies and intra-word dependencies are helpful in word segmentation and POS tagging.

Although the models of Hatori et al. [3] and Zhang et al. [4] perform better than pipeline models, they rely on the one-hot representation of characters and words, and do not assume the similarities among characters and words. In addition, not only words and characters but also many incomplete tokens appear in the transition-based joint parsing process. Such incomplete or unknown words (UNK) could become important cues for parsing, but they are not listed in dictionaries or pre-trained word embeddings. Some recent studies show that character-based embeddings are effective in neural parsing [42, 43], but their models could not be directly applied to joint models because they use given word segmentations. To solve these problems, we propose neural network-based joint models for word segmentation, POS tagging and dependency parsing. We use both character and word embeddings for known tokens and apply character string embeddings

for unknown tokens.

Another problem in the models of Hatori et al. [3] and Zhang et al. [4] is that they rely on detailed feature engineering. Recently, bidirectional LSTM (bi-LSTM) based neural network models with very few feature extraction are proposed [44, 45]. In their models, bi-LSTMs are used to represent the tokens including their context, along with few basic features in the middle of the neural network graph. Indeed, such neural networks can observe whole sentences through bi-LSTMs, whereas the feature-based neural networks cannot. This bi-LSTMs is used to obtain the sentence representation that is similar to neural machine translation models [25]. As a result, Kiperwasser and Goldberg [44] achieve competitive scores with the previous state-of-the-art models. We also develop joint models with n-gram character string bi-LSTM.

We develop the parsing model with neural structured learning based on Andor et al. [2] in addition to the greedy models which run extremely fast. In the experiments, we obtain state-of-the-art Chinese word segmentation and POS tagging scores, and the pipeline of the dependency model achieves the better dependency scores than the previous joint models. To the best of our knowledge, this is the first model to use embeddings and neural networks for Chinese full joint parsing.

Our contributions are summarized as follows: (1) we propose the first embedding-based fully joint parsing model, (2) we use character string embeddings for UNK and incomplete tokens. (3) we also explore bi-LSTM models to avoid the detailed feature engineering in previous approaches. (4) in experiments using a Chinese language corpus, we achieve state-of-the-art scores in word segmentation, POS tagging and dependency parsing.

2.2 Model

All full joint parsing models we present in this paper use the transition-based algorithm in Section 2.2.1 and the embeddings of character strings in Section 2.2.2. We present two neural networks: the feature-input models in Section 2.2.3 and the biLSTM models in Section 2.2.4.

2.2.1 Transition-based Algorithm for Joint Segmentation, POS Tagging, and Dependency Parsing

Based on Hatori et al. [3], we use a modified arc-standard algorithm for character transitions. Fig. 2.1 shows the joint parsing for a Chinese sentence "技术有了新的进展。", where the first four characters "技术有了" have been processed and a partial tree is made. The model consists of one buffer and one stack. The buffer contains characters in the input sentence, and the stack contains words shifted from the buffer. The stack words may have their child nodes. The words in the stack are formed by the following transition operations.

- `SH(t)` (*shift*): Shift the first character of the buffer to the top of the stack as a new word.
- `AP` (*append*): Append the first character of the buffer to the end of the top word of the stack.
- `RR` (*reduce-right*): Reduce the right word of the top two words of the stack, and make the right child node of the left word.
- `RL` (*reduce-left*): Reduce the left word of the top two words of the stack, and make the left child node of the right word.

The `RR` and `RL` operations are the same as those of the arc-standard algorithm [46]. `SH` makes a new word whereas `AP` makes the current word longer by adding one character. The POS tags are attached with the `SH(t)` transition.

In this paper, we explore both greedy models and beam decoding models. This parsing algorithm works in both types. We also develop a joint model of word segmentation and POS tagging, along with a dependency parsing model. The joint model of word segmentation and POS tagging does not have `RR` and `RL` transitions.

2.2.2 Embeddings of Character Strings

First, we explain the embeddings used in the neural networks. Later, we explain details of the neural networks in Section 2.2.3 and 2.2.4.

技术有了新的进展。
Technology have made new progress.

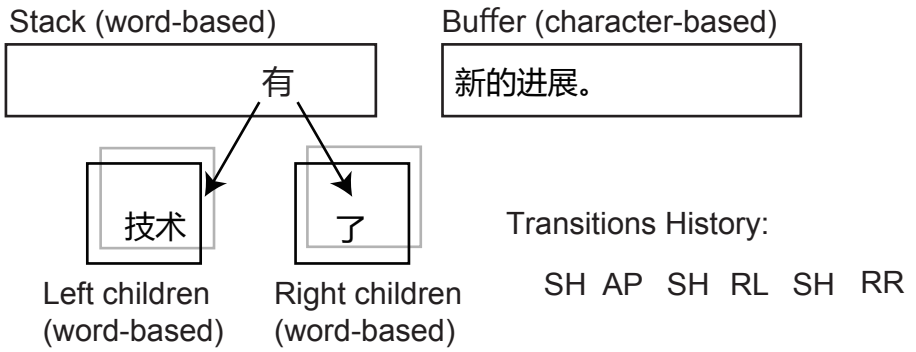

Figure 2.1: Transition-based Chinese joint model for word segmentation, POS tagging and dependency parsing. The buffer contains characters of the input sentences while the stack contains the partial trees.

Both meaningful words and incomplete tokens appear during transition-based joint parsing. Only character sequences that compose some words can be use for word representations as the inputs of models, while other incomplete tokens can not be represented in models regardless to whether the representations used in models are one-hot or distributional representations. If models can handle representations of incomplete tokens, they could become useful features during parsing. For example, “南京东路” (Nanjing East Road, the famous shopping street of Shanghai) is treated as a single Chinese word in the Penn Chinese Treebank (CTB) corpus. There are other named entities of this form in CTB, e.g, “北京西路” (Beijing West Road) and “湘西路” (Hunan West Road). In these cases, “南京” (Nanjing) and “北京” (Beijing) are location words, while “东路” (East Road) and “西路” (West Road) are sub-words. “东路” and “西路” are similar in terms of their character composition and usage, which is not sufficiently considered in the previous work.

“物理学” is often divided into two words “物理” and “学”(learning, subject) in other corpus. In such cases, if word segmentator make a wrong segmentation between

“物理” and “学”, the meaning of the whole sentence might not alter. Therefore, the parser could make a parse tree which is close to the gold tree if they know which words or characters have similar meanings. Moreover, representations of incomplete tokens are helpful for compensating the segmentation ambiguity. Suppose that the parser makes over-segmentation errors and segments “南京东路” to “南京” and “东路”. In this case, “东路” becomes UNK. However, the models could infer that “东路” is also a location, from its character composition and neighboring words. This could give models robustness of segmentation errors. In our models, we prepare the word and character embeddings in the pre-training. We also use the embeddings of character strings for sub-words and UNK which are not in the pre-trained embeddings.

Using character representations are not enough and character string representations are required. To allow models to use the representations of the complete word before its word boundary is determined, the models have to access the tokens across the buffer and the stack. However, such character string features could also produce numerous unknown tokens. In another example, when models have made segmentation errors in the previous transitions, the models have to treat wrongly-segmented tokens for features. Such wrongly-segmented tokens often become UNK. From these reasons, the neural network model we propose here requires not only the word or character embeddings but also the embeddings of arbitrary character strings.

The characters and words are embedded in the same vector space during pre-training. We prepare the same training corpus with the segmented word files and the segmented character files. Both files are concatenated and learned by word2vec [21]. We use the embeddings of 1M frequent words and characters. Words and characters that are in the training set and do not have pre-trained embeddings are given randomly initialized embeddings. The development set and the test set have out-of-vocabulary (OOV) tokens for these embeddings.

The embeddings of the unknown character strings are generated in the neural computation graph when they are required during the model training and parsing. Consider a character string $c_1 c_2 \cdots c_n$ consisting of characters c_i. When this character string is not in the pre-trained embeddings, the model obtains the embeddings $\mathbf{v}(c_1 c_2 \cdots c_n)$ by means of each character embeddings $\sum_{i=1}^{n} \mathbf{v}(c_i)$. The word embeddings and the means of the character embeddings have the same dimension. Embeddings of words, characters

and character strings have the same dimension and are chosen in the neural computation graph. We avoid using the "UNK" vector as far as possible, because this degenerates the information about unknown tokens. However, models use the "UNK" vector if the parser encounters characters that are not in the pre-trained embeddings, though this is quite uncommon.

2.2.3 Feedforward Neural Network with Feature Inputs

Neural Network

We present a neural network with feature inputs model in Figure 2.2. The neural network for greedy training is based on the neural networks of Chen and Manning [37] and Weiss et al. [38]. We add the dynamic generation of the embeddings of character strings for unknown tokens, as described in Section 2.2.2. These embeddings can be swapped to other word embeddings, character embeddings and combinations of them in the neural computation graph. This neural network has two hidden layers with 8,000 dimensions. This is larger than Chen and Manning [37] (200 dimensions) or Weiss et al. [38] (1,024 or 2,048 dimensions). We use the ReLU for the activation function of the hidden layers [47] and the softmax function for the output layer of the greedy neural network. There are three randomly initialized weight matrices between the embedding layers and the softmax function. The loss function $L(\theta)$ for the greedy training is

$$L(\theta) = -\sum_{s,t} \log p_{s,t}^{\text{greedy}} + \frac{\lambda}{2}||\theta||^2,$$

$$p_{s,t}^{\text{greedy}}(\boldsymbol{\beta}) \propto \exp\left(\sum_j w_{tj}\beta_j + b_t\right),$$

where t denotes one transition among the transition set $\mathcal{T}$ ($t \in \mathcal{T}$). s denotes one element of the single mini-batch. $\boldsymbol{\beta}$ denotes the output of the previous layer. $\mathbf{w}$ and $\mathbf{b}$ denote the weight matrix and the bias term. θ contains all parameters. We use the $L2$ penalty term and the Dropout. The backprop is performed including the word and character embeddings. We use Adagrad [48] to optimize learning rate. We also consider Adam [49] and SGD, but find that Adagrad performs better in this model. The other learning parameters are summarized in Table 3.1.

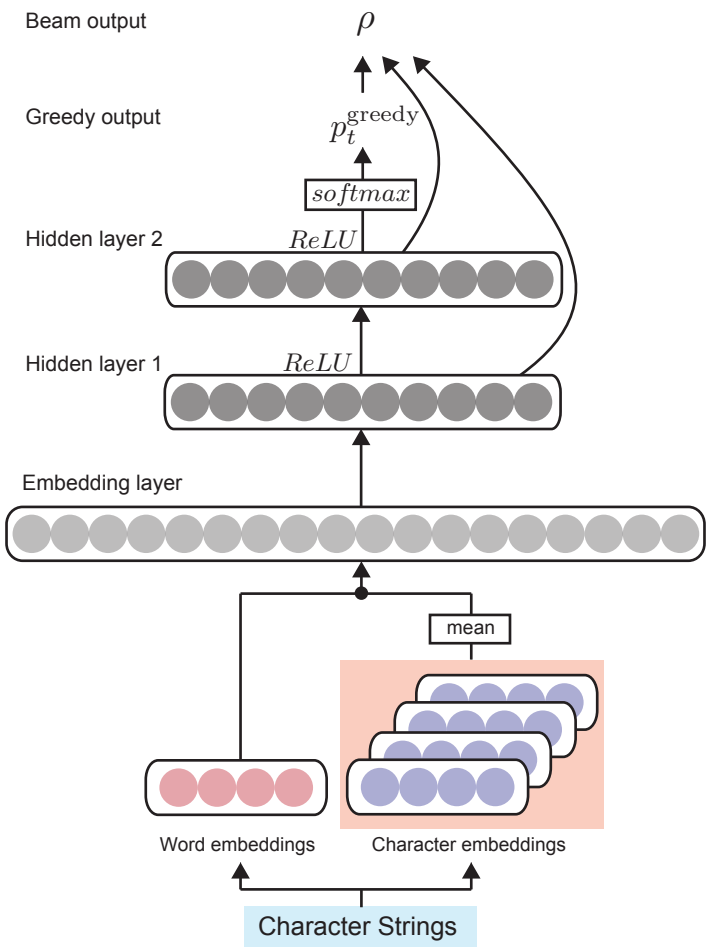

Figure 2.2: The neural network with feature inputs model. The greedy output is obtained at the second top layer, while the beam decoding output is obtained at the top layer. The input character strings are translated into word embeddings if the embeddings of the character strings are available. Otherwise, the embeddings of the character strings are used. We also apply a small size of embeddings for the representations of the length of words.

Type	Value
Size of $\mathbf{h_1}$,$\mathbf{h_2}$	8,000
Initial learning rate	0.01
Initial learning rate of beam decoding	0.001
Embedding vocabulary size	1M
Embedding vector size	200
Small embedding vector size	20
Minibatch size	200

Table 2.1: Parameters for the neural network structure and its training.

In our model implementation, we divide all sentences into training batches. Sentences in the same training batches are simultaneously processed by the neural mini-batches. By doing so, the model can parse all sentences of the training batch in the number of transitions required to parse the longest sentence in the batch. This allows the model to parse more sentences at once, as long as the neural mini-batch can be allocated to the GPU memory. This can be applied to beam decoding.

Features

The features of this neural network are listed in Table 2.2. We use three kinds of features: (1) features obtained from Hatori et al. [3] by removing combinations of features, (2) features obtained from Chen and Manning [37], (3) original features related to character strings. In particular, the original features include sub-words, character strings across the buffer and the stack, and character strings in the buffer. Character strings across the buffer and stack could capture the currently-segmented word. To avoid using character strings that are too long, we restrict the length of character string to a maximum of four characters. This model also uses embeddings with small dimensions as a feature of the parsing states. Unlike Hatori et al. [3], we use sequential characters of sentences for features, and avoid hand-engineered combinations among one-hot features, because such combinations could be automatically generated in the neural hidden layers.

In the later section, we evaluate a joint model for word segmentation and POS tag-

ging. This model does not use the children and children-of-children of stack words as features.

Note that Hatori et al. [3] use complex features including combinations of two or three features of one-hot representations. Such features are learned by averaged perceptron, without hidden layers. In this paper, we do not employ such combinations of the one-hot representation features. All features are given by the embeddings of characters, character strings and words, or the embeddings of parsing states. Therefor, such combinations of several no directly correlated features are represented as distributed representations [14].

Beam Search

Structured learning plays an important role in previous joint parsing models for Chinese.[1] In this paper, we use the structured learning model proposed by Weiss et al. [38] and Andor et al. [2].

In Figure 2.2, the output layer for the beam decoding is at the top of the network. There are a perceptron layer which has inputs from the two hidden layers and the greedy output layer: $[\mathbf{h}_1, \mathbf{h}_2, \mathbf{p}^{\text{greedy}}(\mathbf{y})]$. This layer is learned by the following cost function [2]:

$$L(d^*_{1:j};\theta) = -\sum_{i=1}^{j} \rho(d^*_{1:i-1}, d^*_i;\theta) + \ln \sum_{d'_{1:j}\in\mathcal{B}_{1:j}} \exp \sum_{i=1}^{j} \rho(d'_{1:i-1}, d'_i;\theta),$$

where $d_{1:j}$ denotes the transition path and $d^*_{1:j}$ denotes the gold transition path. $\mathcal{B}_{1:j}$ is the set of transition paths from 1 to j step in beam. ρ is the value of the top layer in Figure 2.2. This training can be applied throughout the network. However, we separately train the last beam layer and the previous greedy network in practice, as in Andor et al. [2]. First, we train the last perceptron layer using the beam cost function freezing the previous greedy-trained layers. After the last layer has been well trained, backprop is performed including the previous layers. We notice that training the embedding layer at this stage could make the results worse, and thus we exclude it. Note that this whole network backprop requires considerable GPU memory. Hence, we exclude particularly

[1] Hatori et al. [3] report that structured learning with a beam size of 64 is optimal.

Type	Features
Stack word and tags	s0w, s1w, s2w
	s0p, s1p, s2p
Stack 1 children and tags	s0l0w, s0r0w, s0l1w, s0r1w
	s0l0p, s0r0p, s0l1p, s0r1p
Stack 2 children	s1l0w, s1r0w, s1l1w, s1r1w
Children of children	s0l0lw, s0r0rw, s1l0lw, s1r0rw
Buffer characters	b0c, b1c, b2c, b3c
Previously shifted words	q0w, q1w
Previously shifted tags	q0p, q1p
Character of q0	q0e
Parts of q0 word	q0f1, q0f2, q0f3
Strings across q0 and buf.	q0f1b1, q0f1b2, q0f1b3
Strings of buffer characters	b0-2, b0-3, b0-4
	b1-3, b1-4, b1-5
	b2-4, b2-5, b2-6
	b3-5, b3-6
	b4-6
Length of q0	lenq0

Table 2.2: Features for the joint model. "q0" denotes the last shifted word and "q1" denotes the word shifted before "q0". In "part of q0 word", f1, f2 and f3 denote sub-words of q0, which are 1, 2 and 3 sequential characters including the last character of q0 respectively. In "strings across q0 and buf.", q0f1bX denotes X sequential characters beginning from q0f1, including the buffer characters. In "strings of buffer characters", bX-Y denotes sequential characters from the X-th to Y-th character of the buffer. The suffix "e" denotes the end character of the word. The dimension of the embedding of "length of q0" is 20.

large batches from the training, because they cannot be on GPU memory. We use multiple beam sizes for training because models can be trained faster with small beam sizes. After the small beam size training, we use larger beam sizes. The test of this fully joint model takes place with a beam size of 16.

Hatori et al. [3] use special alignment steps in beam decoding. The AP transition has size-2 steps, whereas the other transitions have a size-1 step. Using this alignment, the total number of steps for an N-character sentence is guaranteed to be $2N - 1$ (excluding the root arc) for any transition path. This can be interpreted as the AP transition doing two things: appending characters and resolving intra-word dependencies. This alignment stepping assumes that the intra-word dependencies of characters to the right of the characters exist in each Chinese word. [3] Fig. 2.3 shows the example of intra-

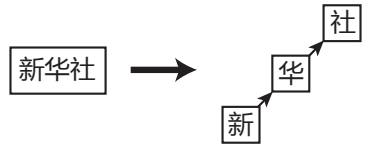

Figure 2.3: Example of the intra-word dependencies.

word dependencies of the Chinese word "新华社". Zhang et al. [4] extend this idea and propose a joint parser resolving both intra-word and inter-word dependencies. Note that their parser requires two stacks for characters and words, along with two beams.

2.2.4 BiLSTM-based Model

In Section 2.2.3, we describe a neural network model with feature extraction. Unfortunately, although this model is fast and very accurate, it has two problems: (1) the neural network cannot see the whole sentence information. (2) it relies on feature engineering. To solve these problems, Kiperwasser and Goldberg [44] propose a biLSTM neural network parsing model. Surprisingly, their model uses very few features, and biLSTM is applied to represent the context of the features. Their neural network consists of three parts: biLSTM, a feature extraction function and a multilayer perceptron (MLP). First, all tokens in the sentences are converted to embeddings. Second, the biLSTM reads all embeddings of the sentence. Third, the feature function extracts the feature representa-

tions of tokens from the biLSTM layer. Finally, an MLP with one hidden layer outputs the transition scores of the transition-based parser.

In this paper, we propose a Chinese joint parsing model with simple and global features using n-gram biLSTM and a simple feature extraction function. The model is described in Figure 2.4. We consider that Chinese sentences consist of tokens, including words, UNKs and incomplete tokens, which can have some meanings and are useful for parsing. Such tokens appear in many parts of the sentence and have arbitrary lengths. To capture them, we propose the n-gram biLSTM. The n-gram biLSTM read through characters $c_i \cdots c_{i+n-1}$ of the sentence (c_i is the i-th character). For example, the 1-gram biLSTM reads each character, and the 2-gram biLSTM reads two consecutive characters $c_i c_{i+1}$. After the n-gram forward LSTM reads character string $c_i \cdots c_{i+n-1}$, it next reads $c_{i+1} \cdots c_{i+n}$. The backward LSTM reads from $c_{i+1} \cdots c_{i+n}$ toward $c_i \cdots c_{i+n-1}$. Therefore, the biLSTM read the n-gram character strings starting from one character after the previous character string. This allows models to capture any n-gram character strings which start from arbitrary positions in the sentence in the input sentence.[2] All n-gram inputs to biLSTM are given by the embeddings of words and characters or the dynamically generated embeddings of character strings, as described in Section 2.2.2. Although these arbitrary n-gram tokens produce UNKs, character string embeddings can capture similarities among them. Following the biLSTM layer, the feature function extracts the corresponding outputs of the biLSTM layer. We summarize the features in Table 2.3. Finally, MLP and the softmax function outputs the transition probability. We use an MLP with three hidden layers as for the model in Section 2.2.3. We train this neural network with the loss function for the greedy training.

2.2.5 The Dependency Label Inference Model

We also propose an inference model of dependency labels that represents the types of dependencies given dependency arcs between head and their modifiers. This model takes inputs of the word and POS tag sequence and the unlabeled dependency tree of the input sentence. The model consists of a single layer biLSTM and a feed forward neural network for inferencing a label for each pair of the head and modifier words. Fig. 2.5 represents

[2] At the end of the sentence of length N, character strings $c_i \cdots c_N (N < i + n - 1)$, which are shorter than n characters, are used.

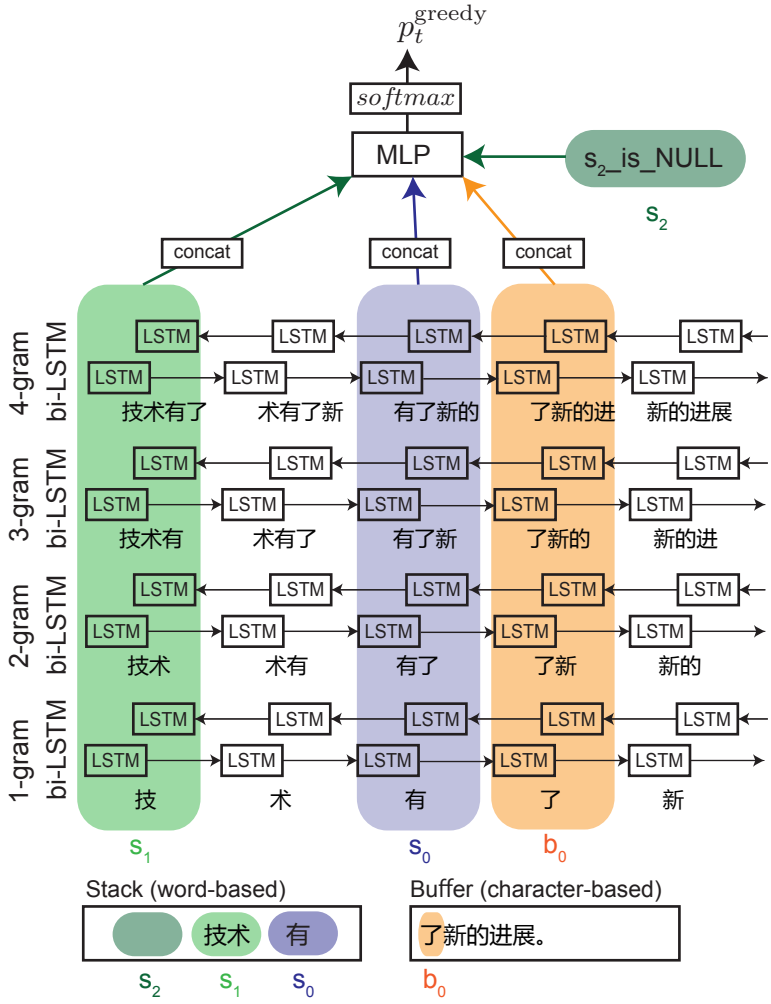

Figure 2.4: The biLSTM model. Similar to the neural network with feature inputs, the embeddings of the character strings are used when the n-gram character string is not listed in the pre-trained word embeddings. The greedy output is obtained at the second top layer. Input character strings are translated to word embeddings if the embeddings of the character strings exist. Otherwise, the mean of character embeddings are inputted instead of word embeddings and the small embeddings of the length of words.

Model	Features
4 features	s0w, s1w, s2w, b0c
8 features	s0w, s1w, s2w, b0c
	s0r0w, s0l0w, s1r0w, s1l0w

Table 2.3: Features for the biLSTM models. All features are words and characters. We experiment both four and eight features models.

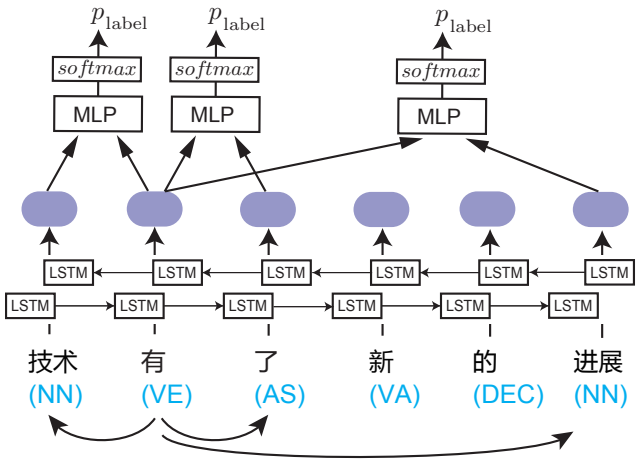

Figure 2.5: The labeling model of dependency labels given dependency trees. The model predicts dependency labels for pairs of heads and their modifiers.

the entire neural network.

To avoid the UNK problems, this model uses the embeddings of words, characters and character sequences and obtains the representations of the input word sequence as the same with the models in previous sections. We then convert the sequence of POS tags into embeddings, concatenate with representations of the input word sequence and input them to the biLSTM. We extract the representations of head and modifier words, input them to the two-layer feed forward neural network. Finally the softmax function at the top of FNN outputs the predicted distributions of dependency labels.

		#snt	#oov
CTB-5	Train	18k	-
	Dev.	350	553
	Test	348	278
CTB-7	Train	31k	-
	Dev.	10k	13k
	Test	10k	13k

Table 2.4: Summary of datasets.

2.3 Experiments

2.3.1 Experimental Settings

We use the Penn Chinese Treebank 5.1 (CTB-5) and 7 (CTB-7) datasets to evaluate our models, following the splitting of Jiang et al. [50] for CTB-5 and Wang et al. [51] for CTB-7. The statistics of datasets are presented in Table 2.4. We use the Chinese Gigaword Corpus for embedding pre-training. Our model is developed for unlabeled dependencies. The development set is used for parameter tuning. Following Hatori et al. [3] and Zhang et al. [4], we use the standard word-level evaluation with F1-measure. The POS tags and dependencies cannot be correct unless the corresponding words are correctly segmented.

We trained three models: SegTag, SegTagDep and Dep. SegTag is the joint word segmentation and POS tagging model. SegTagDep is the full joint segmentation, tagging and dependency parsing model. Dep is the dependency parsing model which is similar to Weiss et al. [38] and Andor et al. [2], but uses the embeddings of character strings. Dep compensates for UNKs and segmentation errors caused by previous word segmentation using embeddings of character strings. We will examine this effect later.

Most experiments are conducted on GPUs, but some of beam decoding processes are performed on CPUs because of the large mini-batch size. The neural network is implemented with Theano.

Model	Seg	POS
Hatori+12 SegTag	97.66	93.61
Hatori+12 SegTag(d)	98.18	94.08
Hatori+12 SegTagDep	97.73	94.46
Hatori+12 SegTagDep(d)	98.26	94.64
M. Zhang+14 EAG	97.76	94.36
Y. Zhang+15	98.04	94.47
SegTag	**98.41**	**94.84**

Table 2.5: Joint segmentation and POS tagging scores. Both scores are in F-measure. In Hatori et al. [3], (d) denotes the use of dictionaries. All scores for previous models are taken from Hatori et al. [3], Zhang et al. [4] and Zhang et al. [5].

2.3.2 Results

Joint Segmentation and POS Tagging

First, we evaluate the joint segmentation and POS tagging model (SegTag). Table 2.5 compares the performance of segmentation and POS tagging using the CTB-5 dataset. We compare our model to three previous approaches: Hatori et al. [3], Zhang et al. [4] and Zhang et al. [5]. Our SegTag joint model is superior to these existing models, including Hatori et al. [3]'s model with rich dictionary information, in terms of both segmentation and POS tagging accuracy.

Joint Segmentation, POS Tagging and Dependency Parsing

Table 2.6 presents the results of our full joint model. We employ the greedy trained full joint model SegTagDep(g) and the beam decoding model SegTagDep. All scores for the existing models in this table are taken from Zhang et al. [4]. Though our model surpasses the previous best end-to-end joint models in terms of segmentation and POS tagging, the dependency score is slightly lower than the previous models. The scores are improved when the global beam search of [2] is applied. The greedy model SegTagDep(g) achieves slightly lower scores than beam models, although this model works considerably fast

Model	Seg	POS	Dep
Hatori+12	97.75	94.33	81.56
M. Zhang+14 EAG	97.76	94.36	**81.70**
SegTagDep(g)	98.24	94.49	80.15
SegTagDep	**98.37**	**94.83**	81.42

Table 2.6: Joint Segmentation, POS Tagging and Dependency Parsing. Hatori et al. [3]'s CTB-5 scores are reported in Zhang et al. [4]. EAG in Zhang et al. [4] denotes the arc-eager model. (g) denotes greedy trained models.

because it does not use beam decoding.

Pipeline of Our Joint SegTag and Dep Model

We use our joint SegTag model for the pipeline input of the Dep model (SegTag+Dep). For simplification, the SegTag model is trained greedily, whereas Dep is trained and tested by the beam cost function with beams of size 4. Table 2.7 presents the results. Our SegTag+Dep model performs best in terms of the dependency and other scores. The SegTag+Dep model is better than the full joint model. This is because most segmentation errors of these models occur around named entities. Hatori et al. [3]'s alignment step assumes the intra-word dependencies in words, while named entities do not always have them. For example, SegTag+Dep model treats named entity "海赛克", a company name, as one word, while the SegTagDep model divides this to "海" (sea) and "赛克", where "赛克" could be used for foreigner's name. For such words, SegTagDep prefers `SH` because `AP` has size-2 step of the character appending and intra-word dependency resolution, which does not exist for named entities. This problem could be solved by adding a special transition `AP_named_entity` which is similar to `AP` but with size-1 step and used only for named entities. Additionally, Zhang et al. [4]'s STD (arc-standard) model works slightly better than Hatori et al. [3]'s fully joint model in terms of the dependency score. Zhang et al. [4]'s STD model is similar to our SegTag+Dep because they combine a word segmentator and a dependency parser using "deque" of words.

Model	Seg	POS	Dep
Hatori+12	97.75	94.33	81.56
M. Zhang+14 STD	97.67	94.28	81.63
M. Zhang+14 EAG	97.76	94.36	81.70
Y. Zhang+15	98.04	94.47	82.01
SegTagDep	98.37	94.83	81.42
SegTag+Dep	**98.41**	**94.84**	**82.36**

Table 2.7: The SegTag+Dep model. Note that the model of Zhang et al. [5] requires other base parsers.

Model	Dep
Dep(g)-cs	80.51
Dep(g)	80.98

Table 2.8: SegTag+Dep(g) model with and without character strings (cs) representations. Note that we compare these models with greedy training for simplicity's sake.

Effect of Character String Embeddings

Finally, we compare the two pipeline models of SegTag+Dep to show the effectiveness of using character string representations instead of "UNK" embeddings. We use two dependency models with greedy training: Dep(g) for dependency model and Dep(g)-cs for dependency model without the character string embeddings . In the Dep(g)-cs model, we use the "UNK" embedding when the embeddings of the input features are unavailable, whereas we use the character string embeddings in model Dep(g). The results are presented in Table 2.8. When the models encounter unknown tokens, using the embeddings of character strings is better than using the "UNK" embedding.

Model	Seg	POS	Dep
Hatori+12	95.42	90.62	73.58
M. Zhang+14 STD	95.53	90.75	**75.63**
SegTagDep(g)	96.06	90.28	73.98
SegTagDep	95.86	**90.91**	74.04
SegTag+Dep(g)	**96.18**	90.90	74.74
SegTag+Dep	**96.18**	90.90	75.12

Table 2.9: Results from SegTag+Dep and SegTagDep applied to the CTB-7 corpus. (g) denotes greedy trained models.

CTB-7 experiments

We also test the SegTagDep and SegTag+Dep models on CTB-7. In these experiments, we notice that the MLP with four hidden layers performs better than the MLP with three hidden layers, but we could not find definite differences in the experiments in CTB-5. We speculate that this is caused by the difference in the training set size. We present the final results with four hidden layers in Table 2.9.

Bi-LSTM Model

We experiment the n-gram bi-LSTMs models with four and eight features listed in Table 2.3. These features are much fewer than the previous models including Hatori et al. [3] and Zhang et al. [4]. Although the features are fewer than previous models, the models could capture tokens which are not included in these features through the bi-LSTMs layer. We summarize the result in Table 2.10. The greedy bi-LSTM models perform slightly worse than the previous models, but they do not rely on feature engineering. This suggests that the bi-LSTMs with fewer features can extract representations that are as good as those of feature-engineered models.

Model	Seg	POS	Dep
Hatori+12	97.75	94.33	81.56
M. Zhang+14 EAG	97.76	94.36	81.70
SegTagDep (g)	98.24	94.49	80.15
Bi-LSTM 4feat.(g)	97.72	93.12	79.03
Bi-LSTM 8feat.(g)	97.70	93.37	79.38

Table 2.10: Bi-LSTM feature extraction model. "4feat." and "8feat." denote the use of four and eight features.

Model	CTB-5	CTB-7
SegTag+Dep (UAS)	82.6	75.1
SegTag+Dep (LAS)	80.0	72.3

Table 2.11: Unlabeled attachment scores (UAS) and labeled attachment scores (LAS) in experiments of CTB-5 and CTB-7. The results of LAS becomes correct if and only if the word segmentation and word dependency of the head and modifier words are correct. The punctuations are removed from the evaluations.

The Inference of the Dependency Labels

We finally present the result of the dependency labeling model for the dependency analysis with labels. We do our experiments of labeling with the results of our best model SegTag+Dep in unlabeled dependencies. We obtain the labeled attachment scores from character inputs without any syntactic and semantic information in CTB-5 and CTB-7. We present the results on Table 2.11.

We also present the detailed analysis of accuracies per label types. We show the percentages of accurate label attachments for the head and modifier words pairs that the original unlabeled dependencies are correct in Table 2.12. In general, the modifiers

	PRD	VC	SUB	OBJ	VMOD	DEP	NMOD	PMOD	AMOD	SBAR
CTB-5	87.4	91.7	94.3	95.7	94.6	94.5	98.5	97.5	95.9	90.7
CTB-7	93.6	84.8	90.6	94.2	97.0	97.0	98.1	97.6	99.1	96.2

Table 2.12: The percentages of accurate labels per correct unlabeled dependencies for label types in CTB5 and CTB7 experiments. We remove the "P" labels to punctuations and the "ROOT" labels to the ROOT entities because these labels are obvious from their definition.

(MOD in tag names) have higher accuracies, while the subjects (SUB) and objects (OBJ), which represents the global structures in the sentence have lower accuracies.

2.4 Related Work

Zhang and Clark [52] propose an incremental joint word segmentation and POS tagging model driven by a single perceptron. Zhang and Clark [53] improve this model by using both character and word-based decoding. Hatori et al. [54] propose a transition-based joint POS tagging and dependency parsing model. Zhang et al. [55] propose a joint model using character structures of words for constituency parsing. Wang et al. [56] also propose a lattice-based joint model for constituency parsing. Zhang et al. [5] propose joint segmentation, POS tagging and dependency re-ranking system. This system requires base parsers. In neural joint models, Zheng et al. [57] propose a neural network-based Chinese word segmentation model based on tag inferences. They extend their models for joint segmentation and POS tagging. Zhu et al. [58] propose the re-ranking system of parsing results with recursive convolutional neural network.

2.5 Conclusion

We propose the joint parsing models by the feature-based and bi-LSTM neural networks. Both of them use the character string embeddings. The character string embeddings help

to capture the similarities of incomplete tokens. We also explore the neural network with few features using n-gram bi-LSTMs. Our SegTagDep joint model achieves better scores of Chinese word segmentation and POS tagging than previous joint models, and our SegTag and Dep pipeline model achieves state-of-the-art score of dependency parsing. The bi-LSTM models reduce the cost of feature engineering.

Chapter 3

PAS Analysis with Neural Generative Approaches

Japanese predicate-argument structure (PAS) analysis involves zero anaphora resolution, which is notoriously difficult because they rely on the huge knowledge that is covered by large corpora. To improve the performance of Japanese PAS analysis, it is straightforward to increase the size of corpora annotated with PAS. However, since it is prohibitively expensive, it is promising to take advantage of a large amount of raw corpora. In this paper, we propose a novel Japanese PAS analysis model based on semi-supervised adversarial training with a raw corpus. In our experiments, our model outperforms existing state-of-the-art models for Japanese PAS analysis.

3.1 Introduction

In pro-drop languages, such as Japanese and Chinese, pronouns are frequently omitted when they are inferable from their contexts and background knowledge. The natural language processing (NLP) task for detecting such omitted pronouns and searching for their antecedents is called zero anaphora resolution. This task is essential for downstream NLP tasks, such as information extraction and summarization.

Especially, pronouns representing speakers or listeners are dropped in natural sentences. Such pronoun-dropping appears as null arguments of each predicates and predicates themselves are rarely dropped in Japanese. Since dropped arguments are closely

correlated to its predicates, many studies of Japanese anaphora resolution are based on predicate argument structure (PAS) analysis. PAS analysis is a task of finding an omitted argument for a predicate. PAS analysis is a task to find an argument for each case of a predicate. For Japanese PAS analysis, the *ga* (nominative, NOM), *wo* (accusative, ACC) and *ni* (dative, DAT) cases are generally handled. These cases are roughly correspond to *who*, *what* and *whom*, respectively. To develop models for Japanese PAS analysis, supervised learning methods using annotated corpora have been applied on the basis of morpho-syntactic clues. However, omitted pronouns have few clues and thus these models try to learn relations between a predicate and its (omitted) argument from the annotated corpora. The annotated corpora consist of several tens of thousands sentences, and it is difficult to learn predicate-argument relations or selectional preferences from such small-scale corpora. The state-of-the-art models for Japanese PAS analysis achieve an accuracy of around 50% for zero pronouns [59, 60, 61, 62, 63].

A promising way to solve this data scarcity problem is enhancing models with a large amount of raw corpora. There are two major approaches to using raw corpora: extracting knowledge from raw corpora beforehand [64, 60] and using raw corpora for data augmentation [65].

In traditional studies on Japanese PAS analysis, selectional preferences are extracted from raw corpora beforehand and are used in PAS analysis models. For example, Sasano and Kurohashi [64] propose a supervised model for Japanese PAS analysis based on case frames, which are automatically acquired from a raw corpus by clustering predicate-argument structures. However, case frames are not based on distributed representations of words and have a data sparseness problem even if a large raw corpus is employed. Some recent approaches to Japanese PAS analysis combines neural network models with knowledge extraction from raw corpora. Shibata et al. [60] extract selectional preferences by an unsupervised method that is similar to negative sampling [21]. They then use the pre-extracted selectional preferences as one of the features to their PAS analysis model. The PAS analysis model is trained by a supervised method and the selectional preference representations are fixed during training. Using pre-trained external knowledge in the form of word embeddings has also been ubiquitous. However, such external knowledge is overwritten in the task-specific training.

The other approach to using raw corpora for PAS analysis is data augmentation. Liu

et al. [65] generate pseudo training data from a raw corpus and use them for their zero pronoun resolution model. They generate the pseudo training data by dropping certain words or pronouns in a raw corpus and assuming them as correct antecedents. After generating the pseudo training data, they rely on ordinary supervised training based on neural networks.

In this paper, we propose a neural semi-supervised model for Japanese PAS analysis. We adopt neural adversarial training to directly exploit the advantage of using a raw corpus. Our model consists of two neural network models: a generator model of Japanese PAS analysis and a so-called "validator" model of the generator prediction. The generator neural network is a model that predicts probabilities of candidate arguments of each predicate using RNN-based features and a head-selection model [66]. The validator neural network gets inputs from the generator and scores them. This validator can score the generator prediction even when PAS gold labels are not available. We apply supervised learning to the generator and unsupervised learning to the entire network using a raw corpus.

Our contributions are summarized as follows: (1) a novel adversarial training model for PAS analysis; (2) learning from a raw corpus as a source of external knowledge; and (3) as a result, we achieve state-of-the-art performance on Japanese PAS analysis.

3.2 Task Description

Japanese PAS analysis determines essential case roles of words for each predicate: *who* did *what* to *whom*. In many languages, such as English, case roles are mainly determined by word order. However, in Japanese, word order is highly flexible. In Japanese, major case roles are the nominative case (NOM), the accusative case (ACC) and the dative case (DAT), which roughly correspond to Japanese surface case markers: が(ga), を(wo), and に(ni). These case markers are often hidden by topic markers, and case arguments are also often omitted.

We explain two detailed tasks of PAS analysis: case analysis and zero anaphora resolution. In Table 3.1, we show four example Japanese sentences and their PAS labels. PAS labels are attached to nominative, accusative and dative cases of each predicate. Sentence (1) has surface case markers that correspond to argument cases.

Sentence (2) is an example sentence for case analysis. Case analysis is a task to find hidden case markers of arguments that have direct dependencies to their predicates. Sentence (2) does not have the nominative case marker が(ga). It is hidden by the topic case marker は(wa). Therefore, a case analysis model has to find the correct NOM case argument 列車(train).

Sentence (3) is an example sentence for zero anaphora resolution. Zero anaphora resolution is a task to find arguments that do not have direct dependencies to their predicates. At the second predicate "巻き込まれた"(was involved), the correct nominative argument is "タクシー"(taxi), while this does not have direct dependencies to the second predicate. A zero anaphora resolution model has to find "タクシー"(taxi) from the sentence, and assign it to the NOM case of the second predicate.

In the zero anaphora resolution task, some correct arguments are not specified in the article. This is called as *exophora*. We consider "author" and "reader" arguments as exophora [67]. They are frequently dropped from Japanese natural sentences. Sentence (4) is an example of dropped nominative arguments. In this sentence, the nominative argument is "あなた" (you), but "あなた" (you) does not appear in the sentence. This is also included in zero anaphora resolution. Except these special arguments of exophora, we focus on intra-sentential anaphora resolution in the same way as [60, 61, 62, 63]. We also attach `NULL` labels to cases that predicates do not have.

3.3 Model

3.3.1 Generative Adversarial Networks

Generative adversarial networks are originally proposed in image generation tasks [31, 68, 69]. In the original model in Goodfellow et al. [31], they propose a generator G and a discriminator D. The discriminator D is trained to divide the real data distribution $p_{data}(\mathbf{x})$ and images generated from the noise samples $\mathbf{z}^{(i)} \in \mathcal{D}_{\mathbf{z}}$ from noise prior $p(\mathbf{z})$. The discriminator loss is

$$\begin{aligned}\mathcal{L}_D = -\big(&\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] \\ &+\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]\big) \ , \end{aligned} \tag{3.1}$$

and they train the discriminator by minimizing this loss while fixing the generator G. Similarly, the generator G is trained through minimizing

$$\mathcal{L}_G = \frac{1}{|\mathcal{D}_{\mathbf{z}}|} \sum_i \Big[\log \Big(1 - D(G(\mathbf{z}^{(i)})) \Big) \Big] \ , \tag{3.2}$$

while fixing the discriminator D. By doing this, the discriminator tries to discriminate the generated images from real images, while the generator tries to generate images that can deceive the adversarial discriminator. This training scheme is applied for many generative tasks including sentence generation [70], machine translation [71], dialog generation [72], and text classification [73].

3.3.2 Proposed Adversarial Training Using Raw Corpus

Japanese PAS analysis and many other syntactic analyses in NLP are not purely generative, and we can make use of a raw corpus instead of the numerical noise distribution $p(\mathbf{z})$. In this work, we use an adversarial training method using a raw corpus, combined with ordinary supervised learning using an annotated corpus. Let $\mathbf{x}_l \in \mathcal{D}_l$ indicate labeled data and $p(\mathbf{x}_l)$ indicate their label distribution. We also use unlabeled data $\mathbf{x}_{ul} \in \mathcal{D}_{ul}$ later. Our generator G can be trained by the cross entropy loss with labeled data:

$$\mathcal{L}_{G/SL} = -\mathbb{E}_{\mathbf{x}_l, y \sim p(\mathbf{x}_l)} \big[\log G(\mathbf{x}_l) \big] \ . \tag{3.3}$$

Supervised training of the generator works by minimizing this loss. Note that we follow the notations of Subramanian et al. [70] in this subsection.

In addition, we train a so-called *validator* against the generator errors. We use the term "validator" instead of "discriminator" for our adversarial training. Unlike the discriminator that is used for dividing generated images and real images, our validator is used to score the generator results. Assume that $\mathbf{y}_l$ is the true labels and $G(\mathbf{x}_l)$ is the predicted label distribution of data $\mathbf{x}_l$ from the generator. We define the labels of the generator errors as:

$$q(G(\mathbf{x}_l), \mathbf{y}_l) = \delta_{\arg\max[G(\mathbf{x}_l)],\ \mathbf{y}_l} \ , \tag{3.4}$$

where $\delta_{i,j} = 1$ only if $i = j$, otherwise $\delta_{i,j} = 0$. This means that q is equal to 1 if the argument that the generator predicts is correct, otherwise 0. We use this generator

error for training labels of the following validator. The inputs of the validator are both the generator outputs $G(\mathbf{x})$ and data $\mathbf{x} \in \mathcal{D}$. The validator can be written as $V(G(\mathbf{x}))$. The validator V is trained with labeled data $\mathbf{x}_l$ by

$$\mathcal{L}_{V/SL} = -\mathbb{E}_{\mathbf{x}_l, y \sim q(G(\mathbf{x}_l), \mathbf{y}_l)} \left[\log V(G(\mathbf{x}_l)) \right] \ , \tag{3.5}$$

while fixing the generator G. This equation means that the validator is trained with labels of the generator error $q(G(\mathbf{x}_l), \mathbf{y}_l)$.

Once the validator is trained, we train the generator with an unsupervised method. The generator G is trained with unlabeled data $\mathbf{x}_{ul} \in \mathcal{D}_{ul}$ by minimizing the loss

$$\mathcal{L}_{G/UL} = -\frac{1}{|\mathcal{D}_{ul}|} \sum_i \left[\log V(G(\mathbf{x}_{ul}^{(i)})) \right] \ , \tag{3.6}$$

while fixing the validator V. This generator training loss using the validator can be explained as follows. The generator tries to increase the validator scores to 1, while the validator is fixed. If the validator is well-trained, it returns scores close to 1 for correct PAS labels that the generator outputs, and 0 for wrong labels. Therefore, in Equation (3.6), the generator tries to predict correct labels in order to increase the scores of fixed validator. Note that the validator has a sigmoid function for the output of scores. Therefore output scores of the validator are in $[0, 1]$.

We first conduct the supervised training of generator network with Equation (3.3). After this, following Goodfellow et al. [31], we use k-steps of the validator training and one-step of the generator training. We also alternately conduct l-steps of supervised training of the generator. The entire loss function of this adversarial training is

$$\mathcal{L} = \mathcal{L}_{G/SL} + \mathcal{L}_{V/SL} + \mathcal{L}_{G/UL} \ . \tag{3.7}$$

Our contribution is that we propose the validator and train it against the generator errors, instead of discriminating generated data from real data. Salimans et al. [68] explore the semi-supervised learning using adversarial training for K-classes image classification tasks. They add a new class of images that are generated by the generator and classify them.

Miyato et al. [74] propose virtual adversarial training for semi-supervised learning. They exploit unlabeled data for continuous smoothing of data distributions based on the

adversarial perturbation of Goodfellow et al. [75]. These studies, however, do not use the counterpart neural networks for learning structures of unlabeled data.

In our Japanese PAS analysis model, the generator corresponds to the head-selection-based neural network for Japanese anaphora resolution. Figure 3.2 shows the entire model. The labeled data correspond to the annotated corpora and the labels correspond to the PAS argument labels. The unlabeled data correspond to raw corpora. We explain the details of the generator and the validator neural networks in Sec.3.3.3 and Sec.3.3.4 in turn.

3.3.3 Generator of PAS Analysis

The generator predicts the probabilities of arguments for each of the NOM, ACC and DAT cases of a predicate. As shown in Figure 3.3, the generator consists of a sentence encoder and an argument selection model. In the sentence encoder, we use a three-layer bidirectional-LSTM (bi-LSTM) to read the whole sentence and extract both global and local features as distributed representations. The argument selection model consists of a two-layer feedforward neural network (FNN) and a softmax function.

For the sentence encoder, inputs are given as a sequence of embeddings $v(x)$, each of which consist of word x, its inflection from, POS and detailed POS. They are concatenated and fed into the bi-LSTM layers. The bi-LSTM layers read these embeddings in forward and backward order and outputs the distributed representations of a predicate and a candidate argument: h_{pred_j} and h_{arg_i}. Note that we also use the exophora entities, i.e., an author and a reader, as argument candidates. Therefore, we use specific embeddings for them. These embeddings are not generated by the bi-LSTM layers but are directly used in the argument selection model.

We also use path embeddings to capture a dependency relation between a predicate and its candidate argument as used in Roth and Lapata [76]. Although Roth and Lapata [76] use a one-way LSTM layer to represent the dependency path from a predicate to its potential argument, we use a bi-LSTM layer for this purpose. We feed the embeddings of words and POS tags to the bi-LSTM layer. In this way, the resulting path embedding represents both predicate-to-argument and argument-to-predicate paths. We concatenate the bidirectional path embeddings to generate $h_{\mathrm{path}_{ij}}$, which represents the dependency relation between the predicate j and its candidate argument i.

For the argument selection model, we apply the argument selection model [66] to evaluate the relation between a predicate and its potential argument for each argument case. In the argument selection model, a single FNN is repeatedly used to calculate scores for a child word and its head candidate word, and then a softmax function calculates normalized probabilities of candidate heads. We use three different FNNs that correspond to the NOM, ACC and DAT cases. These three FNNs have the same inputs of the distributed representations of j-th predicate h_{pred_j}, i-th candidate argument h_{arg_i} and path embedding $h_{\mathrm{path}_{ij}}$ between the predicate j and candidate argument i. The FNNs for NOM, ACC and DAT compute the argument scores $s^{\mathrm{case}_k}_{\mathrm{arg}_i,\mathrm{pred}_j}$, where $\mathrm{case}_k \in \{\mathrm{NOM}, \mathrm{ACC}, \mathrm{DAT}\}$. Finally, the softmax function computes the probability $p(\mathrm{arg}_i|\mathrm{pred}_j,\mathrm{case}_k)$ of candidate argument i for case k of j-th predicate as:

$$p(\mathrm{arg}_i|\mathrm{pred}_j,\mathrm{case}_k) = \frac{\exp\left(s^{\mathrm{case}_k}_{\mathrm{arg}_i,\mathrm{pred}_j}\right)}{\sum_{\mathrm{arg}_i} \exp\left(s^{\mathrm{case}_k}_{\mathrm{arg}_i,\mathrm{pred}_j}\right)}. \tag{3.8}$$

Our argument selection model is similar to the neural network structure of Matsubayashi and Inui [63]. However, Matsubayashi and Inui [63] does not use RNNs to read the whole sentence. Their model is also designed to choose a case label for a pair of a predicate and its argument candidate. In other words, their model can assign the same case label to multiple arguments by itself, while our model does not. Since case arguments are almost unique for each case of a predicate in Japanese, Matsubayashi and Inui [63] select the argument that has the highest probability for each case, even though probabilities of case arguments are not normalized over argument candidates. The model of Ouchi et al. [62] has the same problem.

3.3.4 Validator

We exploit a validator to train the generator using a raw corpus. It consists of a two-layer FNN to which embeddings of a predicate and its arguments are fed. For predicate j, the input of the FNN is the representations of the predicate h'_{pred_j} and three arguments $\left\{h'^{\,\mathrm{NOM}}_{\mathrm{pred}_j}, h'^{\,\mathrm{ACC}}_{\mathrm{pred}_j}, h'^{\,\mathrm{DAT}}_{\mathrm{pred}_j}\right\}$ that are inferred by the generator. The two-layer FNN outputs three values, and then three sigmoid functions compute the scores of scalar values in a range of $[0, 1]$ for the NOM, ACC and DAT cases: $\left\{s'^{\,\mathrm{NOM}}_{\mathrm{pred}_j}, s'^{\,\mathrm{ACC}}_{\mathrm{pred}_j}, s'^{\,\mathrm{DAT}}_{\mathrm{pred}_j}\right\}$. These

scores are the outputs of the validator $D(x)$. We use dropout of 0.5 at the FNN input and hidden layer.

The generator and validator networks are coupled by the attention mechanism, or the weighted sum of the validator embeddings. As shown in Equation (3.8), we compute a probability distribution of candidate arguments. We use the weighted sum of embeddings $v'(x)$ of candidate arguments to compute the input representations of the validator:

$$\begin{aligned} h'^{\,\mathrm{case}_k}_{\mathrm{pred}_j} &= E_{\mathbf{x}\sim p(\mathrm{arg}_i)}[v'(\mathbf{x})] \\ &= \sum_{\mathrm{arg}_i} p(\mathrm{arg}_i|\mathrm{pred}_j,\mathrm{case}_k) v'(\mathrm{arg}_i). \end{aligned}$$

This summation is taken over candidate arguments in the sentence and the exophora entities. Note that we use embeddings $v'(x)$ for the validator that are different from the embeddings $v(x)$ for the generator, in order to separate the computation graphs of the generator and the validator neural networks except the joint part. We use this weighted sum by the softmax outputs instead of the argmax function. This allows the backpropagation through this joint. We also feed the embedding of a predicate to the validator:

$$h'_{\mathrm{pred}_j} = v'(\mathrm{pred}_j). \tag{3.9}$$

Note that the validator is a simple neural network compared with the generator. The validator has limited inputs of predicates and arguments and no inputs of other words in sentences. This allows the generator to overwhelm the validator during the adversarial training.

3.3.5 Implementation Details

The neural networks are trained using backpropagation. The backpropagation has been done to the word and POS tags. We use Adam [49] at the initial training of the generator network for the gradient learning rule. In adversarial learning, Adagrad [48] is suitable because of the stability of learning. We use pre-trained word embeddings from 100M sentences from Japanese web corpus by word2vec [21]. Other embeddings and hidden weights of neural networks are randomly initialized.

For adversarial training, we first train the generator for two epochs by the supervised method, and train the validator while fixing the generator for another epoch. This is

Type	Value
Size of hidden layers of FNNs	1,000
Size of Bi-LSTMs	256
Dim. of word embedding	100
Dim. of POS, detailed POS, inflection form tags	10, 10, 9
Minibatch size for the generator and validator	16, 1

Table 3.1: Parameters for neural network structure and training.

because the validator training preceding the generator training makes the validator result worse. After this, we alternately do the unsupervised training of the generator ($L_{G/UL}$), k-times of supervised training of the validator ($L_{V/SL}$) and l-times of supervised training of the generator ($L_{G/SL}$).

We use the $N(L_{G/UL})/N(L_{G/SL}) = 1/4$ and $N(L_{V/SL})/N(L_{G/SL}) = 1/4$, where $N(\cdot)$ indicates the number of sentences used for training. Also we use minibatch of 16 sentences for both supervised and unsupervised training of the generator, while we do not use minibatch for validator training. Therefore, we use $k = 16$ and $l = 4$. Other parameters are summarized in Table 3.1.

3.4 Experiments

3.4.1 Experimental Settings

Following Shibata et al. [60], we use the KWDLC (Kyoto University Web Document Leads Corpus) corpus [77] for our experiments.[1] This corpus contains various Web documents, such as news articles, personal blogs, and commerce sites. In KWDLC, lead three sentences of each document are annotated with PAS structures including zero pronouns. For a raw corpus, we use a Japanese web corpus created by Hangyo et al. [77], which has no duplicated sentences with KWDLC. This raw corpus is automatically parsed by the Japanese dependency parser KNP.

[1]The KWDLC corpus is available at `http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KWDLC`

KWDLC	# snt	# of dep	# of zero
Train	11,558	9,227	8,216
Dev.	1,585	1,253	821
Test	2,195	1,780	1,669

Table 3.2: KWDLC data statistics: the numbers of sentences, the predications of case analysis and zero anaphora resolution in each corpus split. "of dep" represents the case analysis and the arguments have direct dependencies to their predicates. "of zero" represents the zero anaphora resolution.

KWDLC	NOM	ACC	DAT
# of dep	7,224	1,555	448
# of zero	6,453	515	1,248

Table 3.3: KWDLC training data statistics: the numbers of predictions for each case. "of dep" represents the case analysis and the arguments have direct dependencies to their predicates. "of zero" represents the zero anaphora resolution.

We focus on intra-sentential anaphora resolution, and so we apply a preprocess to KWDLC. We regard the anaphors whose antecedents are in the preceding sentences as `NULL` in the same way as Ouchi et al. [59], Shibata et al. [60]. Tables 3.2 and 3.3 list the statistics of KWDLC.

We use the exophora entities, i.e., an author and a reader, following the annotations in KWDLC. We also assign author/reader labels to the following expressions in the same way as Hangyo et al. [67], Shibata et al. [60]:

author "私" (I), "僕" (I), "我々" (we), "弊社" (our company)

reader "あなた" (you), "君" (you), "客" (customer), "皆様" (you all)

Note that author and reader expressions are often omitted in natural Japanese sentences. So it is reasonable to predict these exophora in our analysis. We also add a `NULL` for

	Case	Zero
Ouchi+ 2015	76.5	42.1
Shibata+ 2016	89.3	53.4
Gen	91.5	56.2
Gen+Adv	**92.0**‡	**58.4**‡

Table 3.4: The results of case analysis (Case) and zero anaphora resolution (Zero). We use F-measure as an evaluation measure. ‡ denotes that the improvement is statistically significant at $p < 0.05$, compared with Gen using paired t-test.

cases that predicates do not have. Precisely speaking, the NULL case of a predicate can be divided into two types. The first type is that the predicate normally requires that case but it does not appear in the sentence. The second type is that the predicate does not have that case in ordinal uses. In the first type of predicates, the model need to find out that the missing arguments are not in the current sentence. In the later case of predicates, the model makes use of the general knowledge of the predicates. This is the case that the extracted knowledge is effective.

Following Ouchi et al. [59] and Shibata et al. [60], we conduct two kinds of analysis: (1) case analysis and (2) zero anaphora resolution. Case analysis is the task to determine the correct case labels when predicates and their arguments have direct dependencies but their case markers are hidden by surface markers, such as topic markers. Zero anaphora resolution is a task to find certain case arguments that do not have direct dependencies to their predicates in the sentence.

Following Shibata et al. [60], we exclude predicates that the same arguments are filled in multiple cases of a predicate. This is relatively uncommon and 1.5 % of the whole corpus are excluded. Predicates are marked in the gold dependency parses. Candidate arguments are just other tokens than predicates. This setting is also the same as Shibata et al. [60].

Folllwoing the previous studies, all performances are evaluated with micro-averaged F-measure scores [60].

	Case analysis			Zero anaphora resolution		
Model	NOM	ACC	DAT	NOM	ACC	DAT
Ouchi+ 2015	87.4	40.2	27.6	48.8	0.0	10.7
Shibata+ 2016	94.1	75.6	30.0	57.7	17.3	37.8
Gen	**95.3**	83.6	39.7	60.7	30.4	41.2
Gen+Adv	**95.3**	**85.4**	**51.5**	**62.3**	**31.1**	**44.6**

Table 3.5: The detailed results of case analysis and zero anaphora resolution for the NOM, ACC and DAT cases. Our models outperform the existing models in all cases. All values are evaluated with F-measure.

	Case	Zero
Gen	91.5	56.2
Gen+Aug	91.2	57.0
Gen+Adv	**92.0**‡	**58.4**‡

Table 3.6: The comparisons of Gen+Adv with Gen and the data augmentation model (Gen+Aug) for case analysis (Case) and zero anaphora resolution (Zero). All values are evaluated in F-measure following the previous experiments. ‡ denotes that the improvement is statistically significant at $p < 0.05$, compared with Gen+Aug.

3.4.2 Experimental Results

We compare two models: the supervised generator model (Gen) and the proposed semi-supervised model with adversarial training (Gen+Adv). We also compare our models with two previous models: Ouchi et al. [59] and Shibata et al. [60], whose performance on the KWDLC corpus is reported.

Table 3.4 lists the experimental results. Our models (Gen and Gen+Adv) outperformed the previous models. Furthermore, the proposed model with adversarial training (Gen+Adv) was significantly better than the supervised model (Gen).

3.4.3 Comparison with Data Augmentation Model

We also compare our GAN-based approach with data augmentation techniques. A data augmentation approach is used in Liu et al. [65]. They automatically process raw corpora and make drops of words following some rules. However, it is difficult to directly apply their approach to Japanese PAS analysis because Japanese zero-pronoun depends on syntactic trees. If we make some drops of arguments of predicates in sentences, this can cause lacks of nodes in syntactic trees. If we prune some branches of dependency trees of the sentence in this way, this causes the data bias problem.

Instead of the approach of Liu et al. [65], we use existing training corpora and word embeddings for the data augmentation. Our augmentation idea is that swapping some words in the training corpus with their related words from the pre-trained word embeddings. First we randomly choose an argument word w in the training corpus and then swap it with another word w' with the probability of $p(w, w')$. We choose top-20 nearest words to the original word w in the pre-trained word embedding as candidates of swapped words. The probability is defined as $p(w, w') \propto [v(w)^\top v(w')]^r$, where $r = 10$. This probability is normalized by top-20 nearest words in the pre-trained word embeddings. We then merge this generated corpus and the original training corpus for training the PAS model in the same way with the supervised Gen model. We conducted several experiments and found that the model trained with the same amount of the generated corpus as the training corpus achieved the best result.

Table 3.6 shows the results of the data augmentation model and the GAN-based model. Our Gen+Adv model performs better than the data augmented model. Note that our data augmentation model does not use raw corpora directly.

3.4.4 Discussion

Result Analysis

We report the detailed performance for each case in Table 3.5. Among the three cases, zero anaphora resolution of the ACC and DAT cases is notoriously difficult. This is attributed to the fact that these ACC and DAT cases are fewer than the NOM case in the corpus as shown in Table 3.3. However, we can see that our proposed model, Gen+Adv, performs much better than the previous models especially for the ACC and DAT cases.

Although the number of training instances of ACC and DAT is much smaller than that of NOM, our semi-supervised model can learn PAS for all three cases using a raw corpus. This indicates that our model can work well in resource-poor cases.

We analyzed the results of Gen+Adv by comparing with Gen and the model of Shibata et al. [60]. Here, we focus on the ACC and DAT cases because their improvements are notable.

- “パックは 洗って、 分別して リサイクルに 出さなきゃいけないので 手間がかかる。“
  It is bothersome to wash, classify and recycle spent packs.

In this sentence, the predicates “洗って” (wash), “分別して” (classify), “(リサイクルに) 出す” (recycle) takes the same ACC argument, “パック” (pack). This is not so easy for Japanese PAS analysis because the actual ACC case marker “を” (wo) of “パック” (pack) is hidden by the topic marker “は” (wa). The Gen+Adv model can detect the correct argument while the model of Shibata et al. [60] fails. In the Gen+Adv model, each predicate gives a high probability to “パック” (pack) as an ACC argument and finally chooses this. We found many examples similar to this and speculate that our model captures a kind of selectional preferences.

The next example is an error of the DAT case by the Gen+Adv model.

- “各専門分野も お任せ下さい。”
  please leave every professional field (to ϕ)

The gold label of this DAT case (to ϕ) is NULL because this argument is not written in the sentence. However, the Gen+Adv model judged the DAT argument as “author”. Although we cannot specify ϕ as “author” only from this sentence, “author” is a possible argument depending on the context.

Validator Analysis

We also evaluate the performance of the validator during the adversarial training with raw corpora. Figure 3.4 shows the validator performance and the generator performance of Zero on the development set. Since our model is trained thought the adversarial learning, or the mini-max game of the two neural networks, the validator performance can be

evaluated with the outputs of the previous generator. In a simple viewpoint, when the generator performs better, the validator performs worse because the validator is trained against the errors of generator outputs.

We notice that the NOM case and the other two cases have different curves in both graphs. In the validator scores, the curve of NOM decreases slightly, while ACC curves increase for the first few epochs. The DAT curve follows the ACC curve thought it fluctuates. In the generator scores, the curve of NOM increases much in the first few epochs, and it slightly increases in later epochs. Other curves drop for the first few epochs and increase much in the later epochs. This can be explained by the speciality of the NOM case. The NOM case has much more author/reader expressions than the other cases. The prediction of author/reader expressions depends not only on selectional preferences of predicates and arguments but on the whole of sentences. Therefore the validator that relies only on predicate and argument representations cannot predict author/reader expressions well.

In the ACC and DAT cases, the scores of the generator and validator increase in the first epochs. This suggests that the validator learns the weakness of the generator and vice versa. However, in later epochs, the scores of the generator increase with fluctuation, while the scores of the validator saturates. This suggests that the generator gradually becomes stronger than the validator.

3.5 Related Work

3.5.1 Japanese PAS Analysis

We did our experiments on the KWDLC corpus [77]. Hangyo et al. [77] introduced the expressions of the reader and writer for the candidates of zero pronounces. Shibata et al. [60] proposed a neural network-based PAS analysis model using local and global features. Their model is based on the non-neural model of Ouchi et al. [59]. Shibata et al. [60] achieved state-of-the-art performances at that time on the case analysis and zero anaphora resolution in KWDLC corpus. They use external resources to extract selectional preferences. Since our model uses external resources, we compare our model with the models of Shibata et al. [60] and Ouchi et al. [59].

Ouchi et al. [62] proposed a semantic role labeling-based PAS analysis model using Grid-RNNs. Matsubayashi and Inui [63] proposed a case label selection model with feature-based neural networks. They conducted their experiments on NAIST Text Corpus (NTC) [78, 61]. NTC consists of newspaper articles, and does not include the annotations of author/reader expressions that are common in Japanese natural sentences, while KWDLC covers wide varieties of documents collected from the web. In addition, there are several important differences between NTC and KWDLC. (1) The granularity of words and tags are different. This causes several problems when someone tries to train a model and export it to the other dataset. (2) NTC consists of newspaper articles, while KWDLC covers wide varieties of documents collected from the web. Therefore, KWDLC might be suitable for general purposes. (3) NTC does not include author/reader expressions, while KWDLC includes the annotations of frequent drops of these. Actually the drops of author/reader expressions are ubiquitous in Japanese natural sentences. NTC does not include these because sentences in newspaper articles tend to obey specific formats.[2] (4) NTC contains more annotation errors than the creators of the corpus expected as described in [61]. Therefore, the scores of NTC are not comparable with those of KWDLC.

KWDLC includes the annotations of frequent drops of author/reader expressions that are common in Japanese natural sentences. It is more likely that newswire sentences have multiple arguments for a single case because of coordinate structures. This is a problem of corpus domains. Therefore if you apply some analysis models for newspaper articles or legal corpus, author/reader expressions are uncommonly omitted from the sentences and the models trained with non-author/reader expressions succeed. Otherwise, the models with author/reader expressions succeed in webblogs or daily conversations corpus.

3.6 Conclusion

We proposed a novel Japanese PAS analysis model that exploits a semi-supervised adversarial training. The generator neural network learns Japanese PAS and selectional

[2] In many cases, using many pronouns of authors and readers sound strange or even rude in Japanese. However, in some domains including newspaper articles and legal documents, sentences are written with specific forms and pronouns are rarely dropped.

preferences, while the validator is trained against the generator errors. This validator enables the generator to be trained from raw corpora and enhance it with external knowledge. In the future, we will apply this semi-supervised training method to other NLP tasks.

	Predicate	NOM	ACC	DAT
(1) タクシーが$_{\text{NOM}}$ 客を$_{\text{ACC}}$ 駅に$_{\text{DAT}}$ 送った 。	送った	タクシー	客	駅
takushi-ga kyaku-wo eki-ni okutta .	okutta	takushi	kyaku	eki
A taxi carried passengers to the station.	sent/carried	taxi	passenger	station
(2) その 列車は 荷物を$_{\text{ACC}}$ 運んだ 。	運んだ	[列車	荷物	
sono ressha-wa nimotsu-wo hakonda.	hakonda	ressha	nimotsu	NULL
The train also carried baggages.	carried	train]	baggage	
(3) タクシーが$_{\text{NOM}}$ 客を$_{\text{ACC}}$ 乗せた とき 事故に$_{\text{DAT}}$ 巻き込まれた 。	乗せた	タクシー	客	
takushi-ga kyaku-wo noseta toki jiko-ni makikomareta.	noseta	takushi	kyaku	NULL
When the taxi picked up passengers, it was involved in the accident.	picked up	taxi	passenger	
	巻き込まれた	[タクシー		事故
	makikomareta	takushi	NULL	jiko
	was involved	taxi]		accident
(4) この 列車に は 乗れません 。	乗れません	[あなた		列車
kono ressha-ni-wa noremasen.	noremasen	anata	NULL	ressha
You can not take this train.	can not take	you]		train

Figure 3.1: Examples of Japanese sentences and their PAS analysis. In sentence (1), case markers (*ga*, *wo*, *ni*) correspond to NOM, ACC, and DAT. In example (2), the correct case marker is hidden by the topic marker (*wa*) In sentence (3), the NOM argument of the second predicate (*makikomareta*) that is dropped. NULL indicates that the predicate does not have the corresponding case argument or that the case argument is not written in the sentence.

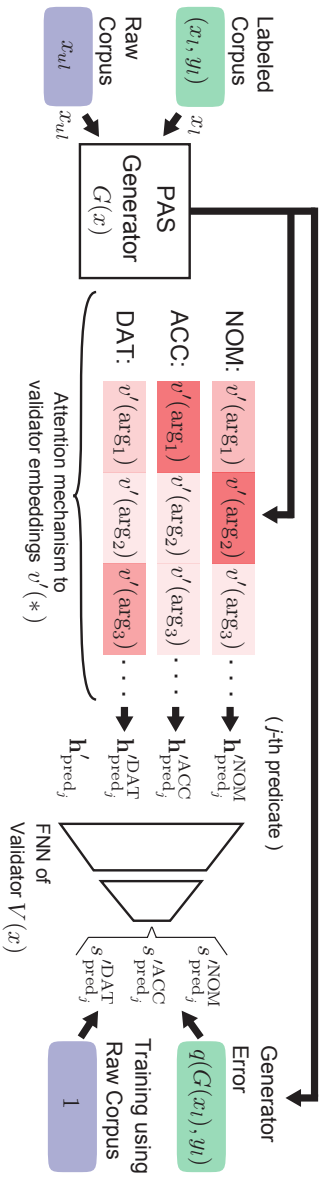

Figure 3.2: The overall model of adversarial training with a raw corpus. The PAS generator $G(x)$ and validator $V(x)$. The validator takes inputs from the generator as a form of the attention mechanism. The validator itself is a simple feed-forward network with inputs of j-th predicate and its argument representations: $\{h'_{\mathrm{pred}_j}, h'^{\mathrm{case}_k}_{\mathrm{pred}_j}\}$. The validator returns scores for three cases and they are used for both the supervised training of the validator and the unsupervised training of the generator. The supervised training of the generator is not included in this figure.

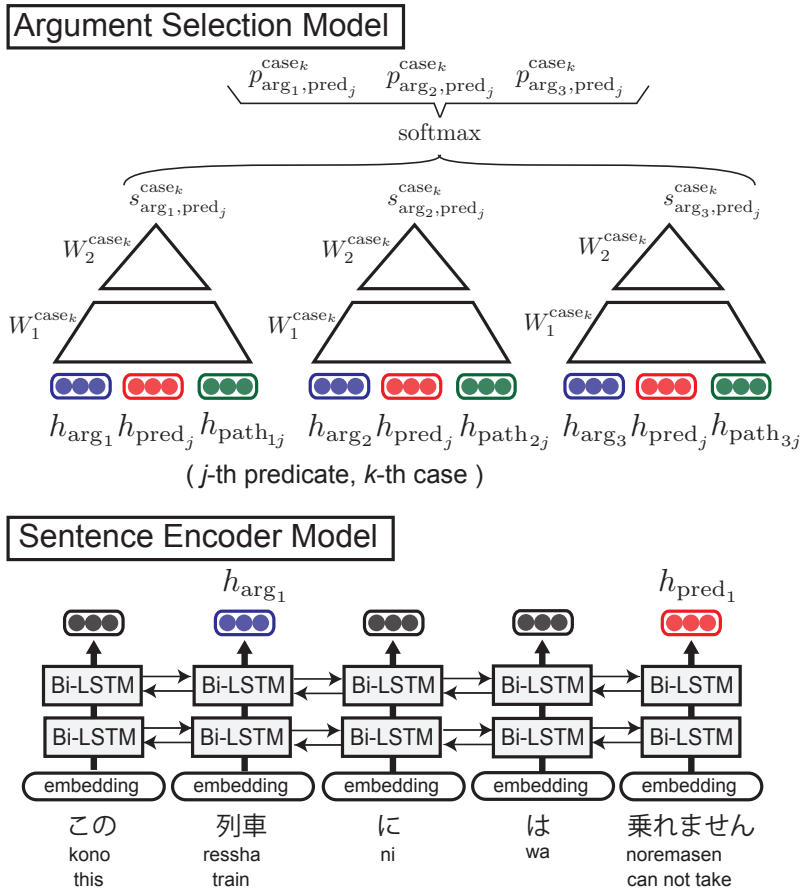

Figure 3.3: The generator of PAS. The sentence encoder is a three-layer bi-LSTM to compute the distributed representations of a predicate and its arguments: h_{pred_i} and h_{arg_i}. The argument selection model is two-layer feedforward neural networks to compute the scores, $s^{\mathrm{case}_k}_{\mathrm{arg}_i,\mathrm{pred}_j}$, of candidate arguments for each case of a predicate.

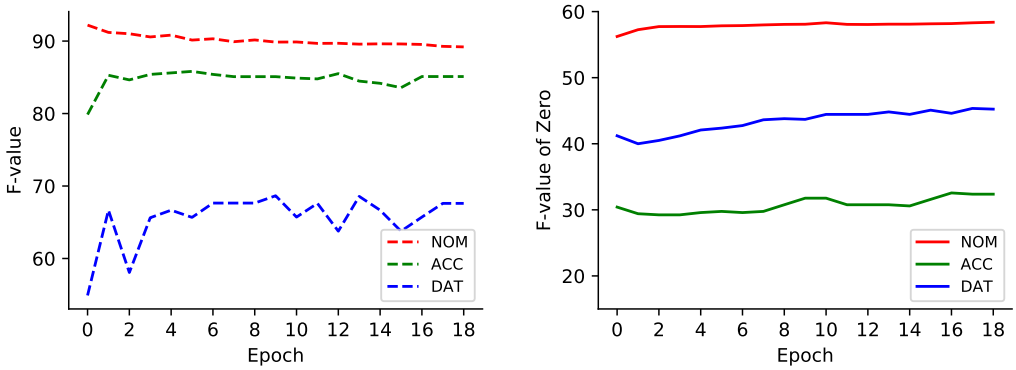

Figure 3.4: Left: the development scores of the validator during adversarial training epochs. Right: the development scores of the generator for the zero pronounce resolutions during adversarial training epochs.

Chapter 4

Semantic Analysis with Reinforcement Learning

In Semantic Dependency Parsing (SDP), semantic relations form directed acyclic graphs, rather than trees. We propose a new iterative predicate selection (IPS) algorithm for SDP. Our IPS algorithm combines the graph-based and transition-based parsing approaches in order to handle *multiple* semantic head words. We train the IPS model using a combination of multi-task learning and task-specific policy gradient training. Trained this way, IPS achieves a new state of the art on the SemEval 2015 Task 18 datasets. Furthermore, we observe that policy gradient training learns an easy-first strategy.

Semantic dependency parsing (SPD) assigns directed acyclic graphs to words in sentences, reflecting their semantic structure according to a linguistic formalism. Because words may have *multiple* semantic heads, head-selection parsing algorithms from syntactic dependency parsing cannot be directly applied. We propose a novel iterative predicate selection (IPS) algorithm. We show that we can learn better IPS state representations using multi-task learning, and that IPS error propagation can be reduced using policy gradient training. By combining multi-task learning and reinforcement learning, our IPS architecture achieves a new state of the art across three linguistic formalisms. In our analysis of our experiments, we make two important observations: (a) Our architecture learns to adopt an easy-first strategy, and (b) in the context of multi-task learning and drop-out, fine-tuning with reinforcement learning is surprisingly robust across hyper-parameters.

4.1 Introduction

Dependency parsers assign syntactic structures to sentences in the form of trees. Semantic dependency parsing (SDP), first introduced in the SemEval 2014 shared task [10], in contrast, is the task of assigning *semantic* structures in the form of directed acyclic graphs to sentences. SDP graphs consist of binary semantic relations, connecting semantic predicates and their arguments. A notable feature of SDP is that words can be the semantic arguments of multiple predicates. For example, in the English sentence: "The man went back and spoke to the desk clerk" – the word "man" is the subject of the two predicates "went back" and "spoke". SDP formalisms typically express this by two directed arcs, from the two predicates to the argument. This yields a directed acyclic graph that expresses various relations among words. However, the fact that SDP structures are directed acyclic graphs means that we cannot apply standard dependency parsing algorithms to SDP.

Standard dependency parsing algorithms are often said to come in two flavors: Transition-based parsers score transitions between states, and gradually build up dependency graphs on the side. Graph-based parsers, in contrast, score all candidate edges directly and apply decoders to the resulting weight matrices. The two types of parsing algorithms have different advantages [8], with transition-based parsers often having more problems with error propagation and, as a result, with long-distance dependencies. This paper presents a compromise between transition-based and graph-based parsing, called *iterative predicate selection* (IPS) – inspired by head selection algorithms for dependency parsing [66] – and show that error propagation, for this algorithm, can be reduced by a combination of multi-task and reinforcement learning.

Multi-task learning is motivated by the fact that there are several linguistic formalisms for SDP. Fig. 4.1 shows the three formalisms used in the shared task. The DELPH-IN MRS (DM) formalism derives from DeepBank [79] and minimal recursion semantics [80]. Predicate-Argument Structure (PAS) is a formalism based on the Enju HPSG parser [81] and is generally considered slightly more syntactic of nature than the other formalisms. Prague Semantic Dependencies (PSD) are extracted from the Czech-English Dependency Treebank [82]. There are several overlaps between these linguistic formalisms, and we show below that parsers, using multi-task learning strategies, can take

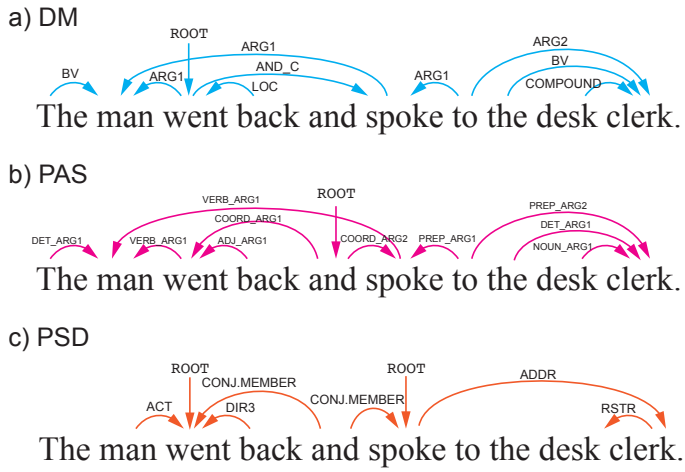

Figure 4.1: Semantic dependency parsing arcs of DM, PAS and PSD formalisms.

advantage of these overlaps or synergies during training. Specifically, we follow Peng et al. (2017) in using multi-task learning to learn representations of parser states that generalize better, but we go beyond their work, using a new parsing algorithm and showing that we can subsequently use reinforcement learning to prevent error propagation and tailor these representations to specific linguistic formalisms.

Contributions In this paper, (i) we propose a new parsing algorithm for semantic dependency parsing (SDP) that combines transition-based and graph-based approaches; (ii) we show that multi-task learning of state representations for this parsing algorithm is superior to single-task training; (iii) we improve this model by task-specific policy gradient fine-tuning; (iv) we achieve a new state of the art result across three linguistic formalisms; finally, (v) we show that policy gradient fine-tuning learns an easy-first strategy, which reduces error propagation.

4.2 Related Work

There are generally two kinds of dependency parsing algorithms, namely transition-based parsing algorithms and graph-based ones [8, 52]. In graph-based parsing, a model is

trained to score all possible dependency arcs between words, and decoding algorithms are subsequently applied to find the most likely dependency graph. The Eisner algorithm [84] is often used for finding the most likely dependency trees, whereas the AD^3 algorithm [85] is used for finding SDP graphs in Peng et al. (2017). During training, the loss is computed after decoding, leading the models to reflect a structured loss. The advantage of graph-based algorithms is that there is no real error propagation to the extent the decoding algorithms are global inference algorithm, but this also means that reinforcement learning is not easily applicable to graph-based parsing. In transition-based parsing, the model is typically taught to follow a gold transition path to obtain a perfect dependency graph during training. This training paradigm has the obvious limitation that the model only ever gets to see states that are on gold transition paths, and error propagation is therefore likely to happen when the parser predicts wrong transitions leading to unseen states [8, 86]. Wang et al. (2018) proposed a similar transition-based parsing model for SDP; they modified some transitions of the Arc-Eager algorithm [88] to create multi-head graphs. There has been several attempts to train transition-based parsers with reinforcement learning approaches. Zhang and Chan (2009) applied SARSA [90] to an Arc-Standard model, using the SARSA updates to fine-tune a model that was pre-trained using a feed-forward neural network. Fried and Klein (2018), more recently, presented experiments with applying policy gradient learning to several constituency parsers, including the recent proposed RNNG transition-based parser [92]. In their experiments, however, the models trained with the policy gradient method do not always perform better than models trained with supervised learning. We hypothesize this is due to credit assignment being difficult in transition-based parsing. Initial errors often make it impossible to recover correct parse trees. Our proposed model explores multiple transition paths at once and avoids making risky decisions in the initial transitions; we also pre-train our model with supervised learning techniques to avoid sampling from irrelevant states at the early stages of training.

4.3 Model

4.3.1 Iterative Predicate Selection

We propose a new semantic dependency parsing algorithm, based on the head-selection algorithm for syntactic dependency parsing [66]. Head selection iterates over sentences, fixing the head of a word w in each iteration, ignoring w in future iterations. This is possible for dependency parsing because each word has a unique head-word, including the root of the sentence, which is attached to an artificial root symbol. However, in SDP, words may attach to multiple head-words or *semantic predicates* whereas other words may not attach to any semantic predicates. Thus, we propose an iterative predicate selection (IPS) parsing algorithm, as a generalization of head-selection in SDP.

The proposed algorithm is formalized as follows. First, we define transition operations for all words in a sentence. For the i-th word w_i in a sentence, the model selects one transition t_i^τ from the set of possible transitions T_i^τ for each transition time step τ. Generally, the possible transitions T_i for the i-th word are expressed as follows:

$$\{\mathrm{NULL}, \mathrm{ARC}_{i,\mathrm{ROOT}}, \mathrm{ARC}_{i,1}, \cdots, \mathrm{ARC}_{i,n}\}$$

where $\mathrm{ARC}_{i,j}$ is a transition to create an arc from the j-th word to the i-th word, encoding that the semantic predicate w_j takes w_i as an semantic argument. `NULL` is a special transition that does not create an arc. The set of possible transitions T_i^τ for the i-th word at time step τ is a subset of possible transitions T_i that satisfy two constraints: (i) no arcs can be reflexive, i.e., w_i cannot be an argument of itself, and (ii) the new arc must not be a member of the set of arcs A^τ comprising the partial parse graph $\mathbf{y}^\tau$ constructed at time step τ. Therefore, we obtain: $T_i^\tau = T_i/(\mathrm{ARC}_{i,i} \cup A^\tau)$. The model then creates semantic dependency arcs by iterating over the sentence as follows:

1 For word w_i, select a head arc from T_i^τ.

2 Update the semantic dependency parse graph.

3 If all words select `NULL`, the parser halts. Otherwise, it returns to **1**.

Fig. 4.2 shows the transitions of IPS parsing algorithm during parsing a sentence "The man went back and spoke to the desk clerk." with DM. There are several paths

from the initial state to the final parsing state, depending on the orders of creating the arcs. This is a part of the non-deterministic oracle problem [86]. In IPS parsing, some arcs are easy to predict; others are very hard to predict. Long-distance arcs are generally difficult to predict, but they are very important for several down-stream applications, including reordering for machine translation [93]. Since long-distance arcs are harder to predict, and transition-based parsers are prone to error propagation, several easy-first strategies have been introduced, both in supervised [94] and unsupervised dependency parsing [95]. We take a more principled approach, learning this kind of strategy using reinforcement learning. We observe, however, that the learned strategies exhibit a clear easy-first preference.

4.3.2 Neural Model

Neural network consists of the encoder model of the input sentences, the encoder of the dependency table and the predicate-selection model. Our neural network model is designed to find transitions for all words in a sentence. Fig. 4.3 shows the overall neural network. It consists of the encoder model of the input sentence and the partial SDP graph, as well as a multi-layered perceptron (MLP) for the semantic head-selection of each word. Firstly the encoder of the sentence the representations of all words in a sentence. Secondly, the feed-forward neural network computes the scores of semantic dependency relations between words and their predicate candidates. Finally, the softmax function choose transitions for all words. We apply both the supervised and reinforcement learning for this neural network.

Sentence Encoder We employ bidirectional long short-term memory (BiLSTM) layers for encoding words in sentences. A BiLSTM consists of two LSTMs that reads the sentence forward and backward, and concatenates their output before passing it on. For a sequence of tokens $[w_1, \cdots, w_n]$, the inputs of tokens in a sentence are words, POS tags and the lemmas.[1] They are mapped to d^e-dimensional embedding vectors by an embedding matrix, concatenated to form $3d^e$-dimensional vectors and used as the input of BiLSTMs. Finally, we obtain the hidden representations of all words $[h(w_1), \cdots, h(w_n)]$

[1] In the analysis of our experiments, we include an ablation test, where we leave out lemma information for a more direct comparison with one of our baselines.

from the three-layer BiLSTMs. We use three-layer stacked BiLSTMs. We also use special embeddings h_{NULL} for the NULL transition and h_{ROOT} for the ROOT of the sentence. These vectors have the same dimension d' as the hidden representations h_{w_i} from BiLSTMs. We use a sequence of vectors $H = [h_{\mathrm{NULL}}, h_{\mathrm{ROOT}}, h_{w_1}, \cdots, h_{w_n}]$ for predicate candidates. These representations of the input sentence are independent from semantic dependency structures

Encoder of Semantic Dependencies The SDP graph is stored in a semantic dependency matrix $D \in \{0,1\}^{n\times(n+1)}$ for a sentence of n words. In the initial state, all cells in the semantic dependency matrix are 0. The rows of the semantic dependency matrix M represent arguments and the columns represent head-candidates, including the ROOT of the sentence, which is represented by the first column of the matrix. For each transition for a word, the model fills in one cell in a row, if the transition is not NULL. A cell $D[i,j]$ is set to 1, when the model predicts that the $i-1$th word is an argument of the jth word or ROOT (if $j=0$). The shape of the semantic dependency matrix is $D \in \{0,1\}^{n\times(n+1)}$ for a sentence of n words. In the initial state, all cells in the semantic dependency matrix are 0.

The parser updates the dependency table during parsing. This semantic dependency matrix is used for both storing parsing results and the feature inputs of parsing states in later transitions.

The model encode the semantic dependency matrix. While the representations of the input sentence are independent from the transition histories, these inputs of dependencies are affected by extracted semantic dependency structures. We also use the semantic dependency matrix to inform the feedforward neural network whether pairs of words and their predicate candidates have already connected by a semantic dependency arcs or not.

We use a single BiLSTM layer for encoding the semantic dependency matrix. Firstly, we convert the semantic dependency matrix D into a rank three tensor $D' \in \mathbb{R}^{n\times(n+1)\times d^e}$. Here e is the dimension of embedding vectors of words and n is the number of words in a sentence. The model constructs D' using the following method: For (i,j)-th cell d_{ij} of the matrix D, the model creates the (i,j)-th element d'_{ij} of tensor D' with the embedding vector of the semantic head-word if they have a directed semantic dependency arc. Otherwise the model swaps the (i,j)-th cell with the special embedding vector w_{NONE}

representing NONE:

$$d'_{ij} = \begin{cases} w_{j-1} & (d_{i,j} = 1) \\ w_{\mathrm{NONE}} & (d_{i,j} = 0) \end{cases} \tag{4.1}$$

and we assign $w_0 = w_{\mathrm{ROOT}}$. [2] In this D' tensor, D'_i contains the representations of the existing predicates for i-th word. For the i-th word, the model obtains the sequence of hidden representations $h^d_{i,j}$ of existing head-words that depends on the position j of the head-candidates by a single layer BiLSTM.

$$[h^d_{i,1}, \cdots, h^d_{i,n+1}] = \mathrm{BiLSTM}(T_i) \tag{4.2}$$

. Note that the BiLSTM reads the semantic heads of the i-th word separately and therefore the model make use of word-wise dependency information. Finally, we concatenate the hidden representation of the NULL transition as follows.

$$H^d = [h^d_{\mathrm{NULL}}, h^d_{*,1}, \cdots, h^d_{*,n+1}] \tag{4.3}$$

We also employ dependency flags that directly encode the semantic dependency matrix and indicate whether the corresponding arcs are already created or not. Flag representations F' are also three-rank tensors, consisting of two hidden representations: f_{ARC} and f_{NOARC} of d^f-dimensional vectors and expressed as follows.

$$f'_{ij} = \begin{cases} f_{\mathrm{ARC}} & (d_{ij} = 1) \\ f_{\mathrm{NOARC}} & (d_{ij} = 0) \end{cases} \tag{4.4}$$

We concatenate the flag representation $f_{\mathrm{NULL}} \in \mathbb{R}^{d^f}$ for the NULL transition at the first axis.

$$F = [f_{\mathrm{NULL}}; f'_{*,1}, \cdots, f_{*,n+1}] \tag{4.5}$$

We do not use BiLSTMs for these flags. These flags reflect the current state of the semantic dependency matrix.

[2] We define that $d_{i,0}$ corresponds to ROOT arcs in semantic dependency matrices. Therefore the model swaps the (i, j)-th cell with w_{j-1} if they have an arc.

Predicate Selection Model The semantic predicate selection model comprises an MLP with inputs from the encoder of the sentence and the semantic dependency matrix: the sentence representation H, the semantic dependency representation H^d, and the dependency flag F. They are rank three tensors and concatenated at the third axis. Formally, the score s_{ij} of the i-th word and the j-th transition is expressed as follows.

$$s_{ij} = \mathrm{MLP}([h_i, h_j, h^d_{ij}, f_{ij}]) \tag{4.6}$$

The model computes the probability of the transition t_j for each word i by applying a softmax function.

$$p_i(t_j) = \mathrm{softmax}_j(s_{ij}) \tag{4.7}$$

This probability of transitions for word w_i is normalized over transition scores s_{i*} with softmax function. These transition probabilities are defined for each word in a sentence. For the MLP, we use a concatenation of a three-layer MLP, a two-layer MLP and a matrix multiplication as follows.

$$\mathrm{MLP}(\mathbf{x}) = W^3_3 a(W^3_2 a(W^3_1 \mathbf{x})) + W^2_2 a(W^2_1 \mathbf{x}) + W^1_1 \mathbf{x} \tag{4.8}$$

W^*_*, are matrices or vectors used in this MLP. Here, we use this MLP for predicting a scalar score s_{ij}; therefore, W^3_3, W^2_2, W^1_1 are vectors. $a(\cdot)$ is an activation function that consists of Dropout and ReLU.

For supervised learning, we use a cross entropy loss

$$L^\tau(\theta) = -\sum_i \sum_T l_i \log p_i(T|\mathbf{y}^\tau) \tag{4.9}$$

for the sentence $\mathbf{x}$ of the parse tree $\mathbf{y}^\tau$ of the time step τ. Here l_i is a label of transitions for the i-th word and θ represents all trainable parameters. This label is drawn from gold transitions and the possible gold transitions may be multiple. This is known as a part of the non-deterministic oracle problem [86]. In supervised learning, we apply random selections from gold transitions to create transition labels.

Labeling Model We also develop a semantic dependency labeling neural network. This neural network consists of three-layer stacked BiLSTMs and a MLP for predicting a semantic dependency label between words and their predicates. We use a MLP that is

defined in Eq.4.8. Note that the output dimension of this MLP is the number of semantic dependency labels. The input of this MLP is the hidden representations of a word i and its predicates j: $[h_i, h_j]$ extracted from the stacked BiLSTMs. The score $s'_{ij}(l)$ of the label l for the arc from predicate j to word i is predicted as follows.

$$s'_{ij}(l) = \mathrm{MLP}'([h_i, h_j]) \tag{4.10}$$

This is trained through the softmax cross entropy loss of supervised learning.

4.3.3 Reinforcement Learning

Policy Gradient Reinforcement learning is a method for learning to iteratively act according to a dynamic environment in order to optimize future rewards. In our context, the agent corresponds to the neural network model predicting the transition probabilities $p(t_i^\tau)$ that are used in the parsing algorithm. The environment includes the current parse graph $\mathbf{y}^\tau$, and the rewards r^τ are computed by comparing the predicted parse graph to the gold parse graph $\mathbf{y}^g$.

We adapt a variation of the policy gradient method [36] for IPS parsing. The objective function to maximize rewards is

$$J(\theta) = E_\pi \left[r_i^\tau \right] \tag{4.11}$$

and transition policy for i-th word is given by the probability of transitions $\pi \sim p_i(t_i^\tau | \mathbf{y}^\tau)$. The gradient of Eq.4.11 is given as follows.

$$\nabla J(\theta) = E_\pi \left[r_i^\tau \nabla \log p_i(t_i^\tau | \mathbf{y}^\tau) \right] \tag{4.12}$$

When we compute this gradient, given a policy π, we approximate the expectation E_π for any transition sequence with a single transition path $\mathbf{t}$ that is sampled from policy π:

$$\nabla J(\theta) \approx \sum_{t_i^\tau \in \mathbf{t}} [r_i^\tau \nabla \log p_i(t_i^\tau | \mathbf{y}^\tau)] \tag{4.13}$$

We summarize our policy gradient learning algorithm for SDP in Algorithm 1. For the i-th word at time step τ, the model samples one transition t_i^τ from the set of possible transitions T_i, following the transition probability of π. After sampling t_i^τ, the model

updates the parse graph to $\mathbf{y}^{\tau+1}$ and computes the reward r_i^τ. For words in a sentence, the model compute a pair of the transition and reward. [3] When NULL becomes the most likely transition for all words, or the time step exceeds the maximum number of time steps allowed, we return the current parse tree. We then update the parameters of our model with the gradients computed from the sampled transitions and their rewards for each time step. We update parameters for each time step to reduce memory requirements.

Note how the cross entropy loss and the policy gradient loss are similar, if we do not sample from the policy π, and rewards are non-negative. However, these are the important differences between supervised learning and reinforcement learning: (1) Reinforcement learning uses sampling of transitions. This allows our model to explore transition paths that supervised models would never follow. (2) In supervised learning, decisions are independent of the current time step τ, while in reinforcement learning, decisions depend on τ. This means that θ parameters should be updated *after* the parser finishes parsing the input sentence. (3) Labels must be non-negative in supervised learning, while rewards can be negative in reinforcement learning.

Overall, the cross entropy loss is able to optimize for choosing transitions given a configuration, while the policy gradient objective function is able to optimize the entire sequence of transitions drawn from sampling according to the policy. We demonstrate the usefulness of reinforcement learning in our experiments below.

Rewards for SDP In reinforcement learning of games or robotics, rewards are often game scores or points the players gain. However, rewards are not straightforwardly defined when reinforcement learning technics are applied to natural language processing tasks. We introduce intermediate rewards, given during parsing, at different time steps. The reward r_i^τ of the i-th word is determined as shown in Table 4.1. The model gets a positive reward for creating a new correct arc to the i-th word, or if the model for the first time chooses a NULL transition after all arcs to the i-th word are correctly created. The model gets a negative reward when the model creates wrong arcs. When our model chooses NULL transitions for the i-th word before all gold arcs are created, the reward r_i^τ becomes 0. However, the model does not necessarily take penalties or negative rewards

[3] For the i-th word that all gold arcs to it are already created and the argmax transition is NULL, the model skip the i-th word because there are no further transitions.

Reward	Transitions
$r_i^\tau = 1$	(1) The model creates a new correct arc from a semantic predicate to the i-th word. (2) The first time the model chooses the NULL transition after all gold arcs to the i-th word have been created, and no wrong arcs to the i words have not been created.
$r_i^\tau = -1$	(3) The model creates a wrong arc from a semantic predicate candidate to the i-th word.
$r_i^\tau = 0$	(4) All other transitions.

Table 4.1: Rewards in SDP policy gradient.

for the i-th word in later transitions and the entire transitions do not necessarily finish at this time. In later transitions, the model can get more rewards when the model create correct arcs to the i-th word or the model choose NULL transition after all arcs to the i-th word are correctly created. Intuitively, this contributes the stability of the reward-based learning.

4.3.4 Implementation Details

We use 100-dimensional, pre-trained Glove [22] word vectors. Words or lemmas in the training corpora that do not appear in pre-trained embeddings are associated with randomly initialized vector representations. Embeddings of POS tags and other special symbol are also randomly initialized. We apply Adam as our optimizer. Preliminary experiments show that mini-batching led to a degradation in performance. When we apply policy gradient, we pre-train our model using supervised learning. We then use policy gradient to task-specific fine-tuning our model. We also find that updating parameters of BiLSTM and word embeddings during policy gradient makes training quite unstable. Therefore we fix the BiLSTM parameters during policy gradient. In our multi-task learning set-up, we apply multi-task learning of the shared stacked BiLSTMs [30, 96] in supervised learning. We use task-specific MLPs for the three different linguistic formalisms: DM, PAS and PSD. Therefore, when we train our model with a multi-task and

Name	Value
Encoder BiLSTM hidden layer size	600
Dependency LSTM hidden layer size	200
The dimensions of embeddings d^e,d^f	100, 128
MLPs hidden layer size	4000
Dropout rate in MLPs	0.5
Maximum transitions of policy gradient	10

Table 4.2: Hyper-parameters in our experiments.

reinforcement learning condition, we train the shared BiLSTM using multi-task learning beforehand, and then we fine-tune the task-specific MLPs with policy gradient. We summarize the rest of our hyper-parameters in Table 4.2.

4.4 Experiments

We use the SemEval 2015 Task18 [97] SDP dataset for evaluating our model. The training corpus contains 33,964 sentences from the WSJ corpus; the development and in-domain test were taken from the same corpus and consist of 1,692 and 1,410 sentences, respectively. The out-of-domain test set of 1,849 sentences is drawn from Brown corpus. All sentences are annotated with three semantic formalisms: DM, PAS and PSD. We use the standard splits of the datasets [98, 99]. Following standard evaluation practice in semantic dependency parsing, all scores are *micro-averaged* F-measures [83, 87] with labeled attachment scores (LAS).

The system we propose is the IPS parser trained with a multi-task objective and fine-tuned using reinforcement learning. This is referred to as *IPS+ML+RL* in the results tables. To highlight the contributions of the various components of our architecture, we also report ablation scores for the IPS parser without multi-task training and reinforcement learning (*IPS*), with only multi-task training (*IPS+ML*) and with only reinforcement learning (*IPS+RL*). We compare our proposed system with three state-of-the-art SDP parsers: Freda3 of Peng et al. (2017), the ensemble model in Wang et al. (2018) and Peng et al. (2018). In Peng et al. (2018), they use syntactic dependency trees, while we do not use them in our models.

Model	DM	PAS	PSD	Avg.
Peng+ 17 Freda3	90.4	92.7	78.5	88.0
Wang+ 18 Ens.	90.3	91.7	78.6	86.9
Peng+ 18	91.6	-	78.9	-
IPS	91.1	92.4	78.6	88.2
IPS +ML	91.2	92.4	78.8	88.3
IPS +RL	91.6‡	**92.8**‡	79.2‡	88.7‡
IPS +ML +RL	**91.9**‡	**92.8**‡	**79.3**‡	**88.8**‡

Table 4.3: Labeled parsing performance on in-domain test data. Avg. is the *weighted* score of three formalisms. ‡ of the *+RL* models represents that the scores are statistically significant at $p < 10^{-3}$ with their non-*RL* counterparts.

The results of our experiments in in-domain dataset are also shown in Table 4.3. We observe that our basic *IPS* model achieve competitive scores in DM and PAS parsing. We assume that this is because our parser exploits the softmax function for selecting arcs rather to external graph-decoding algorithms, such as AD^3. Multi-task learning of the shared BiLSTM (*IPS+ML*) leads to small improvements across the board, which is consistent with the results of multi-task learning results of Peng et al. (2017). The model trained with reinforcement learning (*IPS+RL*) performs better than the model trained by supervised learning (*IPS*). These differences are already significant. Most importantly, the combination of multi-task learning and policy gradient-based reinforcement learning (*IPS+ML+RL*) achieves the best results among all IPS models and the previous state of the art models, by some margin. We also obtain similar results for the out-of-domain datasets, as shown in Table 4.4. All improvements of reinforcement learning are statistically significant in paired t-test.

Evaluating Our Parser without Lemma Since our baseline [83] does not rely on neither lemma or any syntactic information, we also make a comparison of *IPS+ML* and *IPS+ML+RL* trained with word and POS embeddings, but without lemma embeddings. The results are given in Table 4.5. We see that our model is still better on average

Model	DM	PAS	PSD	Avg.
Peng+ 17 Freda3	85.3	**89.0**	76.4	84.4
Peng+ 18	86.7	-	77.1	-
IPS +ML	86.0	88.2	77.2	84.6
IPS +ML +RL	**87.2**‡	88.8‡	**77.7**‡	**85.3**‡

Table 4.4: Labeled parsing performance on out-of-domain test data. Avg. is the micro-averaged score of three formalisms. ‡ of the *+RL* models represents that the scores are statistically significant at $p < 10^{-3}$ with their non-*RL* counterparts.

Model	DM	PAS	PSD	Avg.
Peng+ 17 Freda3	90.4	92.5	78.5	88.0
IPS +ML -Lemma	90.7	92.3	78.3	88.0
IPS +ML +RL -Lemma	**91.2**‡	**92.9**‡	**78.8**‡	**88.5**‡

Table 4.5: The effect of using lemma embeddings on in-domain test datasets. ‡ of *+RL* models represents that the scores are statistically significant at $p < 10^{-3}$ with their non-*RL* counterparts.

and achieves better performance on three formalisms. However, we also notice that the lemma information do not improve the performance in the PAS formalism.

4.4.1 Effect of Reinforcement Learning

In this subsection, we analyze the sequential decision making strategies learned by our policy gradient fine-tuning in the above experiments. Fig. 4.4 shows the distributions of the length of the created arcs in the first, second, third and fourth transitions for all words, in the various IPS models in the development corpus. These distributions show the length of the arcs the models tend to create in the first and later transitions. Since long arcs are harder to predict, an easy-first strategy would typically amount to creating short arcs first.

In supervised learning (*IPS+ML*), there is a slight tendency to create shorter arcs first, but while the ordering is relatively consistent, the differences are very weak. It is slightly more probable that short arcs between neighboring words are created in the first transitions, but the differences with later-transitions are slight.

This is in sharp contrast with the distributions we see for our policy gradient parser (*IPS+ML+RL*). Here, across the board, it is very likely that the first transition connects neighboring words; and very unlikely that neighboring words are connected at later stages. This suggests that reinforcement learning learns an easy-first strategy of predicting short arcs first.

Note that unlike easy-first algorithms in syntactic parsing [86], we do not hardwire an easy-first strategy into our parser; but rather, we learn it from the data, because it optimizes our long-term rewards.

Finally, we give a few examples of the learned easy-first strategy. Fig. 4.5 shows a few sentence excerpts from the development corpus, and the order in which arcs are created. We again compare the model trained with supervised learning (*IPS+ML*) to the model with reinforcement learning (*IPS+ML+RL*). In examples (a) and (b), the *IPS+ML+RL* model creates arcs inside noun phrases first and then creates arcs to the verb. *IPS+ML*, in contrast, creates arcs with inconsistent orders. There are lots of similar examples in the development data. In clause (c), for example, it seems that *IPS+ML+RL* follows a grammatical ordering, while IPS+ML does not. In the last clause (d), it seems that *IPS+ML+RL* first resolves arcs from modifiers, in "*chief financial officer*", then creates an arc from the adjective phrase "*, who will be hired*", and finally creates an arc from the external phrase "*the position of*". Surprisingly, *IPS+ML+RL* completely follows the syntactic dependency structure while we do not feed any syntactic structures during supervised and reinforcement learning.

4.4.2 Effects of Semantic Dependency Matrix

We use the semantic dependency matrix for features of intermediate parsing state, which enable the model to learn from creating arcs. This is a notable advantages of IPS algorithm because the model can know the middle states and choose next transitions. We also observed that if we do not use the representations of the semantic dependency matrix, the performance of the IPS parser become drastically worse in both supervised learning and

reinforcement learning.

4.5 Conclusion

We propose a novel iterative predicate selection (IPS) parsing model for semantic dependency parsing. We apply multi-task learning to learn general representations of parser configurations, and use reinforcement learning for task-specific fine-tuning. In our experiments, our multi-task reinforcement IPS model achieves state-of-the-art results in three SDP linguistic formalisms. We show that reinforcement learning, by exploration of the transition space, enables our model to learn an effective easy-first strategy.

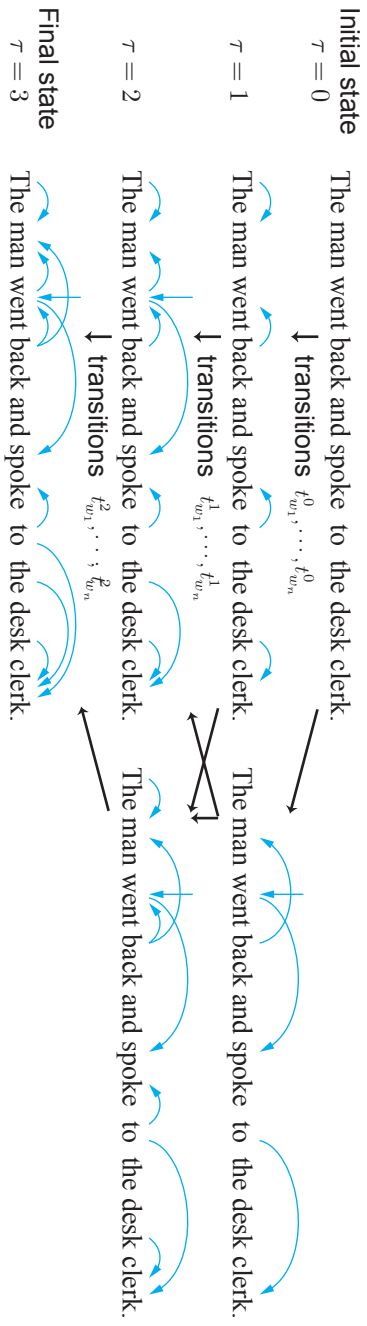

Figure 4.2: Construction of semantic dependency arcs (DM) in the IPS parsing algorithm. Parsing begins from the initial state and proceeds to the final state following one of several paths. In the left path, the model resolves adjacent arcs first. In contrast, in the right path, distant arcs that rely on the global structure are resolved first.

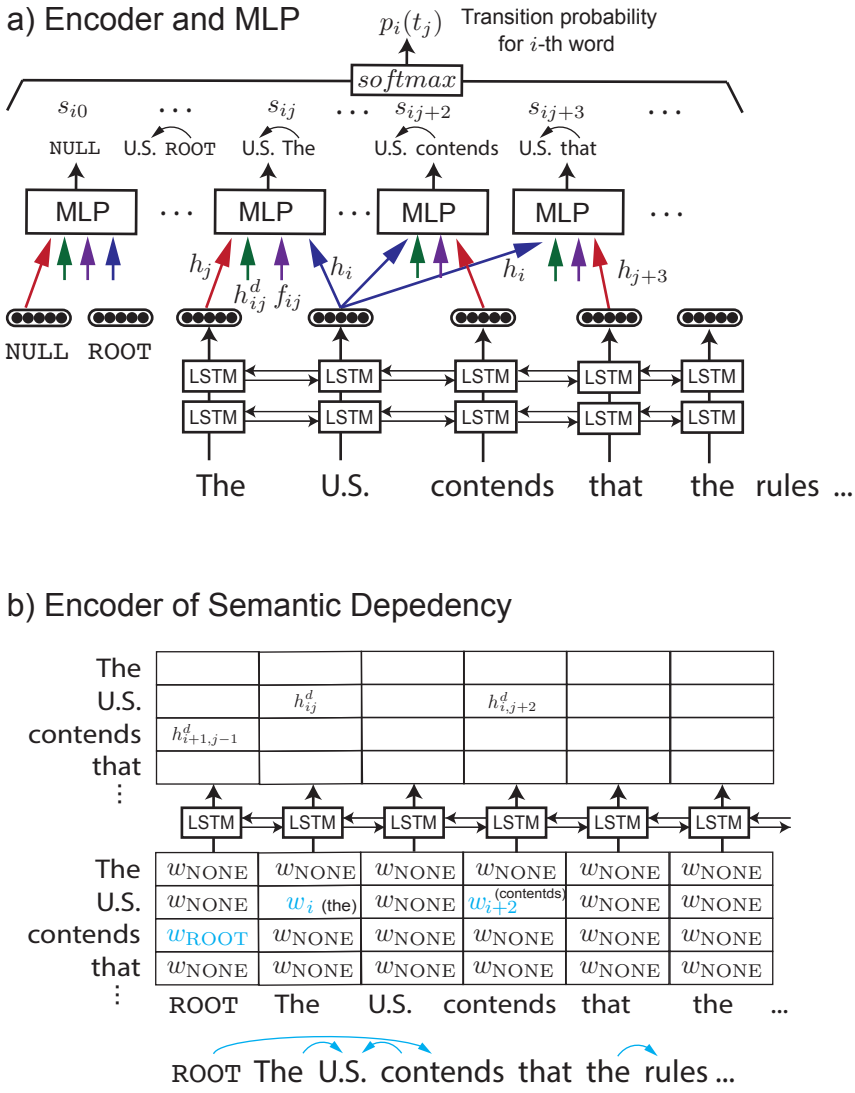

Figure 4.3: Our network architecture: (a) The encoder of the sentence for the hidden representation of h_i and h_j and the MLP for transition probabilities. (b) The encoder of the semantic dependency matrix for the representation of h_{ij}^d. The MLP also takes the arc flag representation f_{ij} (see text for explanation).

Algorithm 1 Policy gradient learning for IPS Algorithm

Input: Sentence $\mathbf{x}$ with an empty parsing tree $\mathbf{y}^0$.

Let a time step $\tau = 0$ and finish flags $f_* = 0$.
for $0 \leq \tau <$ the number of maximum iterations **do**
  Compute π^τ and argmax transitions $\hat{t}_i = \arg\max \pi_i^\tau$.
  if $\forall i$; $\hat{t}_i^\tau = \mathrm{NULL}$ **then**
    break
  end if
  for i-th word in a sentence **do**
    if check a finish flag $f_i = 1$ **then**
      continue
    end if
    if all arcs to word i are correctly created in $\mathbf{y}^\tau$ and $\hat{t}_i = \mathrm{NULL}$ **then**
      Let a flag $f = 1$
      continue
    end if
    Sample t_i^τ from π_i^τ.
    Update the parsing tree $\mathbf{y}^\tau$ to $\mathbf{y}^{\tau+1}$.
    Compute a new reward r_i^τ from $\mathbf{y}^\tau$, $\mathbf{y}^{\tau+1}$ and $\mathbf{y}^g$.
  end for
  Store a pair of the state, transitions and rewards for words $\{\mathbf{y}^\tau, t_*^\tau, r_*^\tau\}$.
end for
Shuffle pairs of $\{\mathbf{y}^\tau, t_*^\tau, r_*^\tau\}$ for a time step τ.
for a pair $\{\mathbf{y}^{\tau'}, t_*^\tau, r_*^{\tau'}\}$ of time step τ' **do**
  Compute gradient and update parameters.
end for

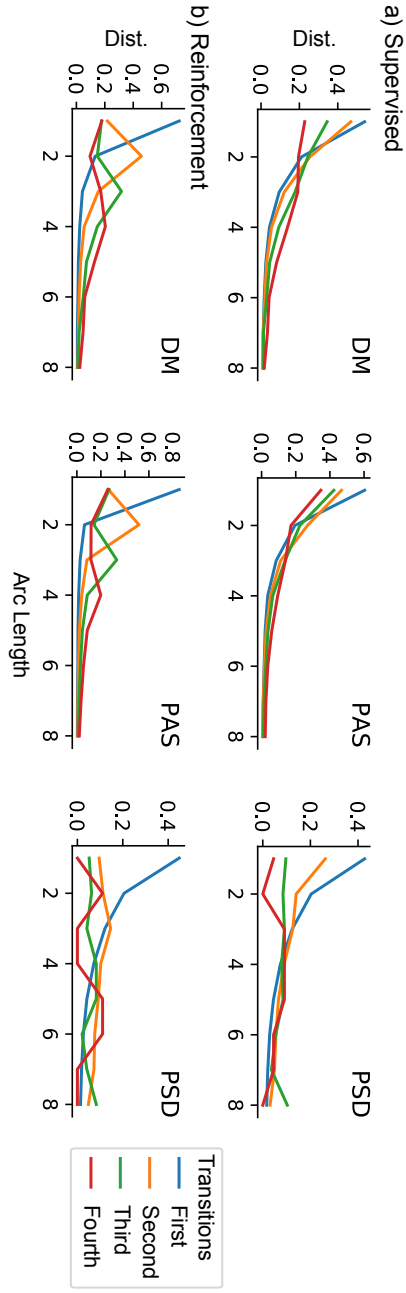

Figure 4.4: Distributions of the length of arcs: (a) Supervised learning (*IPS+ML*). (b) Reinforcement learning (*IPS+ML+RL*). The four lines correspond to the first to fourth transitions in the derivations.

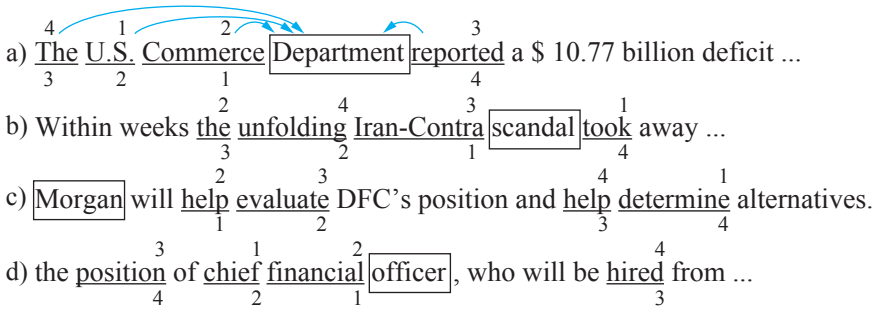

Figure 4.5: Examples of clauses parsed with DM formalism. The underlined words are the semantic predicates of the argument words in rectangles in the annotation. The superscript numbers are the orders of creating arcs by *IPS+ML* and the subscript numbers are the orders by *IPS+ML+RL*. In the clause (a), we show a partial SDP graph to visualize the SDP arcs.

Chapter 5

Conclusion

5.1 Overview

In this thesis, I introduced three different models in the fundamental analysis of NLP models. These models are from syntactic to semantic analyses: joint syntactic analysis, case analysis and zero anaphora resolution and semantic dependency parsing.

In the first joint syntactic analysis, I applied the joint model of word segmentation, POS tagging and dependency parsing. This achieves the state-of-the-art performance of Chinese syntactic analysis and leads the multi-task learning models of the modern NLP models.

After we investigate the joint syntactic model, we proceed to the syntactic and semantic analysis: case analysis and zero pronounce resolution. This task largely depends on selectional preference: the combinations of predicates and their arguments. We proposed the generative adversarial neural network-based model for semi-supervised learning and extraction of selectional preferences. In the experiments of KWDLC corpus, we achieve state-of-the-art performance in Japanese case analysis and zero anaphora resolution.

We notice that many of the previous models rely on hand-engineered parsing structures for semantic analysis. We propose a new semantic dependency parsing model and train it with reinforcement learning. Reinforcement learning allows the model to select the optimal parsing way depending on the data structure.

In this decade, neural networks have turned to be the most sophisticated way to represent natural language entities. They have succeeded in many supervised tasks. Indeed

if you train your model with only limited supervised datasets, your model would acquire not the general knowledge or the common sense but task-specific representations. In other words, if you develop models with supervised learning, your task was "solved" when your datasets were created. Actually, it is possible that such task-specific models solve NLP tasks without common sense. Multi-task learning contributes neural networks to learn knowledge that is shared among several tasks, while it gives up task-specific turning in many cases. If neural network obtains the common sense as human have, it would become the most sophisticated model for general tasks. In this thesis, we explorer the basis of those models.

5.2 The Relations of The Proposed models

Our proposed models are from syntactic to semantic analysis. We choose the suitable neural network approaches for each of the analyses. We emphasize that these neural approaches can be widely applied for the many fundamental analyses tasks. For example, the multi-task and joint neural networks are used not only joint syntactic analysis but also the later analyses. The PAS analysis model jointly resolves the Japanese case analysis and zero anaphora resolutions. Our semantic dependency parsing model employs the multi-task learning of the DM, PAS and PSD formalisms.

The generative adversarial neural network and the reinforcement learning approaches are also closely correlated and the generative adversarial learning can be interpreted with reinforcement learning viewpoints. The validator of the generative adversarial neural network is used for extracting the reward signals of the reinforcement learning. In the semantic dependency parsing model, rewards are given by specific rules, while the PAS analyses model exploits the validator for predicting rewards.

These approaches are closely related and further studies should be done based on these approaches.

5.3 Future Work

5.3.1 The Joint Model of Fundamental Analysis and Application

In this thesis, we focus on the fundamental analysis of natural texts and propose a joint model of them. However, such models are often combined with the systems in the NLP applications. In machine translation studies, recent models often take end-to-end approaches while the fundamental analysis models are attempted to be used with. In other application tasks such as chatbot studies, the entire system consists of small models such as the fundamental analysis of input texts, searching models and text generation models. In both studies, the fundamental analysis is indeed essential because the meaning of the input texts determines the next actions of the system.

However, models are independently developed in current systems. This is because the multi-task learning datasets that can be used for both the fundamental analysis and applications do not exist in many cases. Although this is a really important research issue, it is nearly prohibited to create such datasets because of the annotation costs. The next studies of NLP would be *how to fulfil the gap between the language understanding models and application models?* Reinforcement learning of models under surrounding models as environments would be one possible answer.

5.3.2 Neural Network and Future

The idea of neural network originally comes from the network of neuron cells in the cortex of the human brains. Such neurons are modeled as spiking neurons with ordinal differential equations for their activation or often called as *spikes* for a kind of binary outputs. Today, many researchers use neural networks that use different formalisms, for example, ReLU function for their activation and continuous value outputs. Instead of their simplicities, these neural networks outperform many existing models in many machine learning tasks including those of natural language processing.

However, it seems that the current neural network has a critical difference from those of biological neural networks, or especially human brains, rather to those model differences. Neural network models lack the common sense of our world. From an NLP viewpoint, the lack of common sense means the lack of external knowledge. In this thesis, we explorer the models to compensate for the lack of external knowledge with the

generative approach. Reinforcement learning is another possible solution for models to learn external knowledge through interaction with the external world.

In conclusion, we consider that how to teach neural networks common sense would be the most important problem to be solved in the next decades. Natural language processing must be a key to solve this, but this problem might be solved with the combinations of artificial intelligence studies including natural language processing.

Bibliography

[1] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 160–167, 2008.

[2] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proceedings of the ACL*, pages 2442–2452, 2016.

[3] Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the ACL*, pages 1045–1053, 2012.

[4] Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. Character-level chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the ACL*, pages 1326–1336, 2014.

[5] Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. Randomized greedy inference for joint segmentation, pos tagging and dependency parsing. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligenc*, pages 42–52, 2015.

[6] Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. Morphological analysis for unsegmented languages using recurrent neural network language model. In

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2292–2297. Association for Computational Linguistics, 2015.

[7] Sadao Kurohashi and Makoto Nagao. A syntactic analysis method of long japanese sentences based on coordinate structure' detection. *Natural Language Processing of Japan*, 1(1):35–57, 1994.

[8] Ryan McDonald and Joakim Nivre. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on EMNLP-CoNLL*, pages 122–131, 2007.

[9] Sriram Pemmaraju and Steven Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica ®*. Cambridge University Press, New York, NY, USA, 2003. ISBN 0521806860.

[10] Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland, August 2014.

[11] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[12] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20 (3):273–297, September 1995. ISSN 0885-6125.

[13] Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the ACL*, pages 111–118, 2004.

[14] Geoffrey E. Hinton, J. L. McClelland, and D. E. Rumelhart. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, pages Vol.1, p.12, 1986.

[15] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–, October 1986.

[16] Shun-ichi Amari. A theory of adaptive pattern classifiers. *IEEE Trans. Electronic Computers*, 16:299–307, 1967.

[17] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, pages 14(2):179–211, 1990.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667.

[19] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[20] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. volume abs/1301.3781, 2013.

[22] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[23] Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2011.

[24] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751, 2014.

[25] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[26] Makoto Nagao. A framework of a mechanical translation between japanese and english by analogy principle. In *Proc. Of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, New York, NY, USA, 1984. Elsevier North-Holland, Inc.

[27] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1243–1252, 2017.

[28] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[30] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the ACL (Short Papers)*, pages 231–235, 2016.

[31] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[32] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 214–223, 2017.

[33] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. In *Machine Learning*, pages 279–292, 1992.

[34] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. 1994.

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. volume abs/1312.5602, 2013.

[36] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. pages 5–32. Springer, 1992.

[37] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on EMNLP*, pages 740–750, 2014.

[38] David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the ACL*, pages 323–333, 2015.

[39] Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the ACL*, pages 1213–1222, 2015.

[40] Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. Improved transition-based parsing and tagging with neural networks. In *Proceedings of the EMNLP*, pages 1354–1359, 2015.

[41] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the ACL*, pages 334–343, 2015.

[42] Miguel Ballesteros, Chris Dyer, and Noah A. Smith. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the EMNLP*, pages 349–359, 2015.

[43] Xiaoqing Zheng, Haoyuan Peng, Yi Chen, Pengjing Zhang, and Zhang Wenqiang. Character-based parsing with convolutional neural network. In *Proceedings of the NAACL:HLT*, page 153, 2015.

[44] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *TACL*, 4:313–327, 2016. ISSN 2307-387X.

[45] James Cross and Liang Huang. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany, August 2016. Association for Computational Linguistics.

[46] Joakim Nivre. Incrementality in deterministic dependency parsing. In Frank Keller, Stephen Clark, Matthew Crocker, and Mark Steedman, editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, 2004a.

[47] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814, 2010.

[48] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Number UCB/EECS-2010-24, 2010.

[49] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the ACL*, 2015.

[50] Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904, 2008.

[51] Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 309–317, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing.

[52] Yue Zhang and Stephen Clark. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the EMNLP*, pages 562–571, 2008.

[53] Yue Zhang and Stephen Clark. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the EMNLP*, pages 843–852, 2010.

[54] Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224. Asian Federation of Natural Language Processing, 2011.

[55] Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. Chinese parsing exploiting characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 125–134, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[56] Zhiguo Wang, Chengqing Zong, and Nianwen Xue. A lattice-based framework for joint chinese word segmentation, pos tagging and parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 623–627, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[57] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on EMNLP*, pages 647–657, 2013.

[58] Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of the ACL*, pages 1159–1168, 2015.

[59] Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. Joint case argument identification for japanese predicate argument structure analysis. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*

and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 961–970, Beijing, China, July 2015. Association for Computational Linguistics.

[60] Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. Neural network-based model for japanese predicate argument structure analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1235–1244. Association for Computational Linguistics, August 2016.

[61] Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai, and Julien Kloetzer. Intra-sentential subject zero anaphora resolution using multi-column convolutional neural network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1244–1254, Austin, Texas, November 2016. Association for Computational Linguistics.

[62] Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. Neural modeling of multi-predicate interactions for japanese predicate argument structure analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1591–1600, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[63] Yuichiroh Matsubayashi and Kentaro Inui. Revisiting the design issues of local models for japanese predicate-argument structure analysis. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 128–133, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing.

[64] Ryohei Sasano and Sadao Kurohashi. A discriminative approach to japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 758–766, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing.

[65] Ting Liu, Yiming Cui, Qingyu Yin, Wei-Nan Zhang, Shijin Wang, and Guoping Hu. Generating and exploiting large-scale pseudo training data for zero pro-

noun resolution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 102–111, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[66] Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. Dependency parsing as head selection. In *Proceedings of the ACL*, pages 665–676, Valencia, Spain, April 2017.

[67] Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 924–934, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[68] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2234–2242. Curran Associates, Inc., 2016.

[69] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. 2015.

[70] Sandeep Subramanian, Sai Rajeswar, Francis Dutil, Chris Pal, and Aaron Courville. Adversarial generation of natural language. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 241–251, Vancouver, Canada, August 2017.

[71] Denny Britz, Quoc Le, and Reid Pryzant. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126, Copenhagen, Denmark, September 2017.

[72] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on EMNLP*, pages 2157–2169, Copenhagen, Denmark, September 2017.

[73] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. In *Proceedings of the ACL*, pages 1–10, Vancouver, Canada, July 2017.

[74] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing by virtual adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[75] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[76] Michael Roth and Mirella Lapata. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the ACL*, pages 1192–1202, Berlin, Germany, August 2016.

[77] Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. Building a diverse document leads corpus annotated with semantic relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 535–544, Bali,Indonesia, November 2012. Faculty of Computer Science, Universitas Indonesia.

[78] Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. Annotating a japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the Linguistic Annotation Workshop, LAW@ACL 2007, Prague, Czech Republic, June 28-29, 2007*, pages 132–139, 2007.

[79] Daniel Flickinger, Yi Zhang, and Valia Kordoni. Deepbank: Adynamically annotated treebank of the wall street journal. In *In Proc. of TLT*, 2012.

[80] Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. Minimal recursion semantics: An in- troduction. In *Research on Language & Computation*, pages 3(4):281—-332, 2005.

[81] Yusuke Miyao, Takashi Ninomiya, and Jun'ichi. Tsujii. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In *In Proceedings of IJCNLP-04*, 2004.

[82] Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3153–3160, May 2012.

[83] Hao Peng, Sam Thomson, and Noah A. Smith. Deep multitask learning for semantic dependency parsing. In *Proceedings of the ACL*, pages 2037–2048, Vancouver, Canada, July 2017.

[84] J. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *COLING*, 1996.

[85] Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. Dual decomposition with many overlapping components. In *Proceedings of the 2011 Conference on EMNLP*, pages 238–249, Edinburgh, Scotland, UK., July 2011.

[86] Yoav Goldberg and Joakim Nivre. Training deterministic parsers with non-deterministic oracles. pages 403–414, 2013.

[87] Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. A neural transition-based approach for semantic dependency graph parsing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[88] Joakim Nivre and Mario Scholz. Deterministic dependency parsing of english text. In *Proceedings of Coling 2004*, pages 64–70. COLING, 2004b.

[89] Lidan Zhang and Kwok Ping Chan. Dependency parsing with energy-based reinforcement learning. In *Proceedings of the IWPT*, pages 234–237, Paris, France, October 2009.

[90] Leemon C. Baird III. Reinforcement learning through gradient descent. School of Computer Science Carnegie Mellon University, 1999.

[91] Daniel Fried and Dan Klein. Policy gradient as a proxy for dynamic oracles in constituency parsing. In *Proceedings of the ACL*, pages 469–476, 2018.

[92] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the NAACL: HLT*, pages 199–209, San Diego, California, June 2016.

[93] Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. Using a dependency parser to improve smt for subject-object-verb languages. In *Proceedings of HLT:NAACL*, pages 245–253, Boulder, Colorado, June 2009.

[94] Yoav Goldberg and Michael Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: NAACL*, pages 742–750, Los Angeles, California, June 2010.

[95] Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on EMNLP*, pages 1281–1290, 2011.

[96] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the EMNLP*, pages 1923–1933, 2017.

[97] Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado, June 2015. Association for Computational Linguistics.

[98] S. C. Almeida Mariana and F. T. MartinsGoldberg André. Lisbon: Evaluating turbosemanticparser on multiple languages and out-of-domain data. 2015.

[99] Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and XiaojunWan. Peking: Building semantic de- pendency graphs with a hybrid parser. 2015.

[100] Hao Peng, Sam Thomson, and Noah A. Smith. Backpropagating through structured argmax using a spigot. In *Proceedings of the 56th Annual Meeting of the ACL*, pages 1863–1873, 2018.

List of Publications

[1] Shuhei Kurita, Daisuke Kawahara and Sadao Kurohashi, Neural Adversarial Training for Semi-supervised Japanese Predicate-argument Structure Analysis, In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL2018)*, Melbourne, Australia, (July 2018).

[2] Shuhei Kurita, Daisuke Kawahara and Sadao Kurohashi, Neural Joint Model for Transition-based Chinese Syntactic Analysis, In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL2017)*, Vancouver, Canada, (July 2017). Out-standing paper.

[3] Ryota Kobayashi, Shuhei Kurita, Katsunori Kitano, Kenji Mizuseki, Barry J Richmond and Shigeru Shinomoto, Reconstructing Neuronal Circuitry from Parallel Spike Trains, bioRxiv. `https://www.biorxiv.org/content/early/2018/05/30/334078`.

[4] Shuhei Kurita, Daisuke Kawahara and Sadao Kurohashi, Neural Network-based Chinese Joint Syntactic Analysis, Journal of Natural Language Processing in Japan. (March 2019). To appear.

[5] Ryota Kobayashi, Shuhei Kurita, Katsunori Kitano, Kenji Mizuseki, Barry J. Richmond and Shigeru Shinomoto, Estimating synaptic connections from parallel spike trains, In *Proceedings of The 28th Annual Conference of the Japanese Neural Network Society*, Okinawa, Japan, (October 2018).

[6] R. Kobayashi, S. Kurita, Y. Yamanaka, K. Kitano, K. Mizuseki, B.J. Richmond and S. Shinomoto, A method for estimating synaptic connections from parallel spike trains, In *Proceedings of 13th International Neural Coding Workshop, NC2018*, Torino, Italia, (September 2018).

[7] Shuhei Kurita, Daisuke Kawahara and Sadao Kurohashi, Joint Analysis of word segmentation, POS tagging and Syntactic Parsing, In *Proceedings of 23th Annual Conference of Association for Natural Language Processing*, Tsukuba, Japan, (March 2017).

[8] Ryota Kobayashi, Shuhei Kurita, Yuzuru Yamanaka, Katsunori Kitano and Shigeru Shinomoto, Testing statistical significance of synaptic connectivity, In *Proceedings of 12th International Neural Coding Workshop, NC2016*, Cologne, Germany, (August 2016).

[9] Shuhei Kurita, Yuzuru Yamanaka, Katsunori Kitano and Shigeru Shinomoto, Minimal time length of spike trains for the inference of connectivity, In *Proceedings of 24th Annual Conference of Japanese Neural Network Society*, Oral and Poster, Hakodate, Japan, (August 2014).

[10] Shuhei Kurita, Ryota Kobayashi, Katsunori Kitano and Shigeru Shinomoto, Connectivity Inference using simulated spike data of neural networks, In *Proceedings of 69th Annual Conference of Physics Society of Japan*, Kanagawa, Japan, (March 2013).