

FLEXIBLE STRUCTURED PREDICTION IN NATURAL LANGUAGE  
PROCESSING WITH PARTIALLY ANNOTATED CORPORA

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Xiao Zhang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL  
STATEMENT OF DISSERTATION APPROVAL

Dr. Dan Goldwasser

Department of Computer Science

Dr. Jean Honorio

Department of Computer Science

Dr. Jennifer Neville

Department of Computer Science

Dr. Alex Pothén

Department of Computer Science

Approved by:

Dr. Sunil Prabhakar

Head of the School Graduate Program

To my dear grandfather, who just passed away in Jan 2020. As a barefooted accountant, he nourished my childhood and cultivated my spirit.

## ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude to a lot of people without whose support and help I would not have been able to complete my Ph.D. studies.

First, I would like to thank my major professor, Dr. Dan Goldwasser, for the advising and support throughout the years of my PhD studies. I want to thank him for giving me the freedom to choose the research topics that I am most interested in while keeping me going in the right direction, and for all the valuable discussion on improving and deepening the research. I also would like to thank my other Ph.D. committee members, Drs. Jean Honorio, Jennifer Neville and Alex Pothén, for all their help and advice. Additionally I would like to thank Dr. Gregory Francis from the department of psychological sciences, for his mentoring and guidance. I am extending my thanks to Drs. Anindra Bhadra, Xiao Wang and Faming Liang from the department of statistics, for their teaching and collaborations, which greatly broadened my views of machine learning.

I would like to thank all the former and current students in the Purdue NLP Group, especially Maria Leonor Pacheco, I-ta Lee, and Rajkumar Pujari for their discussion, collaboration and friendship. I would like to extend my thanks to Drs. Pablo Robles-Granda, Xuejiao Kang, Yongyang Yu, Hao Peng, Lei Ceng, Zhiwei Zhang, Zhengyi Zhang and Meng-Lin Wu for their support and friendship after I started my study in the Department of Computer Science at Purdue, to Xuankang Lin, Daniel Kumor, Xin Cheng, Ellen Lai, Mengyue Hang, Linjie Li, Nitin Jain, Peng Hao from the department of Computer Science and to Wei Deng, Jiapeng Liu, Zhanyu Wang from the department of Statistics and many others for their discussion, collaboration, companion and friendship during my Ph.D. studies.

I would like to thank collaborators and friends outside of Purdue for their help, support and mentoring during my study at Purdue and external internships, especially Drs. Kewei Tu, Hyokun Yun and Manish Marwah. I am also extending my thanks to Sherrill Chen, Dr. Yong Jiang, Dr. Wenshu Zhang, Kushal Lakhotia and many others for their friendship and help.

Finally I would like to thank my beloved fiancée and my family for their love and support, especially during my sickness in the last year. Without you, it will not be possible to recover and write this dissertation.

A special thanks to all the faculty and staff members at Purdue university during this pandemic period of the COVID-19 virus, who are making their best efforts to maintain things on and keep us safe and healthy. Without you, it is not possible to write a dissertation during this time.

During my Ph.D. studies I was supported in part by research assistantships funded by DARPA ASED and Google and in part by teaching assistantships in the Department of Computer Science at Purdue University.

## PREFACE

The basis for this dissertation originally stemmed from my curiosity to understand the relationship between the natural language, the human brain and the computer. To my own surprise, my past education on Linguistics, Psychology and Computer Science finally found a way to converge during my Ph.D. studies—Natural Language Processing (NLP). I was really fortunate to have found such a subject to pursue the Ph.D. degree, which aggregates all my knowledge obtained in the previous years.

Living in such a digital world with huge volumes of data growing in an exponential scale, there is a great need for us to use all types of available data to help process the information. How to use the Big Data effectively and efficiently in NLP? This question motivated my passion to seek out the answers during my Ph.D. studies. This dissertation is a record of the path I took during this fantastic and endless journey.

## TABLE OF CONTENTS

	Page
LIST OF TABLES . . . . .	x
LIST OF FIGURES . . . . .	xii
ABBREVIATIONS . . . . .	xiv
ABSTRACT . . . . .	xv
1 INTRODUCTION . . . . .	1
1.1 Three Types of Approaches to Structured Prediction with Reduced Supervision . . . . .	3
1.2 Related Work . . . . .	4
1.2.1 Learning with Multi-task and Modular Architecture . . . . .	4
1.2.2 Sequence Labeling with Latent Variables . . . . .	5
1.2.3 Tree Induction with Latent Variables . . . . .	5
1.2.4 Limitations . . . . .	6
1.2.5 Other Related Areas . . . . .	6
1.3 Dissertation Overview . . . . .	9
1.3.1 Dissertation Organization . . . . .	10
2 PRELIMINARIES . . . . .	11
2.1 Dependency Graphs and Trees . . . . .	11
2.2 Variational Autoencoder (VAE) . . . . .	12
2.3 Tree Conditional Random Field . . . . .	12
3 MODULAR NEURAL ARCHITECTURE FOR STRUCTURED PREDIC- TION WITH REDUCED SUPERVISION . . . . .	14
3.1 Introduction . . . . .	14
3.2 Related Works . . . . .	18
3.3 Architectures for Sequence Prediction . . . . .	18
3.3.1 CRF Layer . . . . .	19
3.4 Functional Decomposition of Composite Tasks . . . . .	20
3.4.1 Twofold Modular Model . . . . .	21
3.4.2 Two-fold Modular Infusion Model . . . . .	23
3.4.3 Guided Gating Infusion . . . . .	24
3.5 Learning with Full and Partial Labels . . . . .	24
3.6 Experimental Evaluation . . . . .	25
3.6.1 Experimental Settings . . . . .	26

	Page
3.6.2	Q1: Monolithic vs. Modular Learning . . . . . 28
3.6.3	Q2: Partial Labels as Weak Supervision . . . . . 31
3.7	Conclusion . . . . . 33
4	NEURAL CONDITIONAL RANDOM FIELDS AUTOENCODER . . . . . 35
4.1	Introduction . . . . . 36
4.2	Related Work . . . . . 38
4.3	Neural CRF Autoencoder . . . . . 39
4.3.1	Neural CRF Encoder . . . . . 40
4.3.2	Generative Decoder . . . . . 43
4.4	A Unified Learning Framework . . . . . 44
4.4.1	Unified Loss Functions for Labeled and unlabeled Data . . . . . 44
4.5	Mixed Expectation-Maximization Algorithm . . . . . 45
4.6	Experiments . . . . . 49
4.6.1	Experimental Settings . . . . . 49
4.6.2	Supervised Learning . . . . . 50
4.6.3	Semi-supervised Learning . . . . . 53
4.7	Conclusion . . . . . 55
5	SEMI-SUPERVISED AUTOENCODING DEPENDENCY PARSING . . . . . 57
5.1	Introduction . . . . . 58
5.2	Related Work . . . . . 60
5.3	Graph-based Dependency Parsing . . . . . 61
5.3.1	Scoring Function Using Neural Architecture . . . . . 62
5.4	Locally Autoencoding Parser (LAP) . . . . . 64
5.4.1	Incorporating POS and External Embeddings . . . . . 65
5.5	Globally Autoencoding Parser (GAP) . . . . . 65
5.5.1	Discriminative Component: the Encoder . . . . . 66
5.5.2	Generative Component: the Decoder . . . . . 66
5.5.3	A Unified Supervised and Unsupervised Learning Framework . . . . . 67
5.5.4	Learning . . . . . 68
5.5.5	Tractable Inference . . . . . 69
5.6	Experiments . . . . . 71
5.6.1	Experimental Settings . . . . . 71
5.6.2	Semi-Supervised Dependency Parsing on WSJ Dataset . . . . . 72
5.6.3	Semi-supervised Dependency Parsing on the UD Dataset . . . . . 74
5.7	Comparison of Complexity . . . . . 75
5.8	Conclusion . . . . . 75
6	CONCLUSIONS . . . . . 77
6.1	Summary . . . . . 77
6.2	Contributions . . . . . 78
6.3	Future Work . . . . . 79



	Page
REFERENCES . . . . .	82
A Additional Experimental Results for the Modular Neural Architecture . . .	103
A.1 Examples of Task Decomposition . . . . .	103
A.2 Full Experimental Results on Targeted Sentiment . . . . .	104
A.3 Experiments on NER . . . . .	104
A.4 Additional Experiments on Knowledge Integration . . . . .	105
A.5 Convergence Analysis . . . . .	106
B Supplementary Materials for LAP and GAP . . . . .	109
B.1 ELBO of LAP’s Original Objective . . . . .	109
B.2 Details of the Auxiliary Posterior . . . . .	110
B.2.1 Mean Field Approximation and Annealing . . . . .	110
B.2.2 Empirical Bayesian Treatment . . . . .	111
B.3 Proof of the Convexity of GAP’s ELBO . . . . .	111
B.4 Marginalization and Expectation of Latent Parse Trees . . . . .	112
B.5 Details of the Tractable Algorithm for GAP . . . . .	113
VITA . . . . .	117
PUBLICATIONS . . . . .	118

## LIST OF TABLES

Table	Page
3.1 This table compares our models with the competing models on the targeted sentiment task. The results are on the full prediction of both segmentation and sentiment. . . . .	29
3.2 This table compares our models with recent results on the Aspect Sentiment datasets. . . . .	30
3.3 This table compares our models with competing models on the subjective sentiment task. . . . .	30
4.1 This table shows the statistics of different UD languages used in our experiments, including the number of tokens, and the number of sentences in training (train), development (dev) and testing (test) set respectively. .	49
4.2 This table shows supervised learning performance of POS tagging on 8 UD languages using different models . . . . .	51
4.3 This table shows semi-supervised learning accuracy of POS tagging on 8 UD languages. HEM means hard-EM, used as a self-training approach, and OL means only 20% of the labeled data is used and no unlabeled data is used. . . . .	52
5.1 This table shows statistics of multiple languages we used in our experiments: the numbers of sentences in the training, development and testing set. . . . .	71
5.2 In this table we compare different models on multiple languages from UD. Models were trained in a fully supervised fashion with labeled data only (noted as “L”) or semi-supervised (notes as “L+U”). “ST” stands for self-training. . . . .	72
5.3 This table compares the model performance on the WSJ dataset with 10% labeled data. “L” means only 10% labeled data is used, while “L+U” means both 10% labeled and 90% unlabeled data are used. . . . .	73
5.4 This table compares the complexity of different models in this table, in which $l$ is the length of the sentence. . . . .	75
A.1 This table compares the performance on the targeted sentiment task . .	104

Table	Page
A.2 This table compares our models with several state-of-the-art systems on the CoNLL 2003 English NER dataset. . . . .	105
A.3 This table compares our models with recent results on the 2002 CoNLL Dutch and Spanish NER datasets. . . . .	106

## LIST OF FIGURES

Figure	Page
3.1 This figure shows a target-sentiment decomposition example with segmentation and sentiment. . . . .	21
3.2 This figure shows three modular models for task decomposition. In all of them, blue blocks are the segmentation modules, detecting entity location and segmentation, and yellow blocks are the type modules, recognizing the entity type or sentiment polarity. Green blocks are the final decision modules, integrating all the decisions. (G) refers to “Guided Gating”. . . . .	22
3.3 This figure shows modular knowledge integration experimental results on the Targeted Sentiment Datasets. The x-axis is the amount of percentage of the third fold of full labels. The “non-modularized” means we only provide fully labeled data from the third fold. . . . .	32
3.4 The fully labeled data were fixed to 20% of the whole training set, and data with only segmentation information (Magenta), or with only type information (Orange) were gradually added. Then the model was tested on the full prediction test. The LSTM-CRF model can only use fully labeled data as it does not decompose the task. . . . .	33
3.5 This figure shows domain transfer experimental results with fixed 20% in-domain data from aspect sentiment and various amounts of out-of-domain data from target sentiment, shown on the x-axis. . . . .	34
4.1 On the left is a generalized autoencoder, of which the lower half is the encoder and the upper half is the decoder. On the right is an illustration of the graphical model of our NCRFAE model. The yellow squares are interactive (transition) potentials among labels, and the green squares represent the unary (emission) potentials generated by the neural networks. . . . .	39
4.2 A demonstration of the neural CRF encoder. $\mathbf{l}_t$ and $\mathbf{r}_t$ are the output of the forward and backward character-level LSTM of the word at position $t$ in a sentence, and $\mathbf{e}_t$ is the word-level embedding of that word. $\mathbf{u}_t$ is the concatenation of $\mathbf{e}_t$ , $\mathbf{l}_t$ and $\mathbf{r}_t$ , denoted by blue dashed arrows. . . . .	43
4.3 This figure shows an example from the test set to compare the predicted results of our NCRFAE model, the NCRF model and the LSTM model. . . . .	51

Figure	Page
4.4 UD English and Croatian POS tagging accuracy versus increasing proportion of unlabeled sequences using 20% labeled data. The green straight line is the performance of the neural CRF, trained over the labeled data. . . . .	54
4.5 This figure shows the performance of the NCRFAE model on different proportion of labeled and unlabeled data. The green line shows the results on only labeled data, and the red line on both labeled and unlabeled data. The difference between the red line and the green line are gradually vanishing.	55
5.1 This figure shows a dependency tree: directional arcs represent head-modifier relation between tokens. . . . .	58
5.2 This figure illustrates two different parsers. (a) LAP uses continuous latent variable to form the dependency tree (b) GAP treats the dependency tree as the latent variable. . . . .	63
5.3 This figure illustrates the arc scoring matrix, in which each entry represents the $(h(head) \rightarrow m(modifier))$ score. . . . .	63
5.4 This figure illustrates how tractable inference can be done by marginalization in a arc-decomposed manner. . . . .	69
A.1 This figure shows an example of NER decomposition. . . . .	103
A.2 This figure illustrates an extra example of target sentiment decomposition.	103
A.3 This figure shows experimental results on modular knowledge integration on the Dutch and Spanish NER datasets. . . . .	106
A.4 This figure shows experimental results on modular knowledge integration on the Subjective Polarity Disambiguation datasets. . . . .	107
A.5 This figure compares the convergence over the development set on the English NER dataset. The x-axis is number of epochs and the y-axis is the F1-score. . . . .	108
A.6 This figure compares the convergence over the development set on the subjective polarity disambiguation datasets. The x-axis is number of epochs and the y-axis is the F1-score. . . . .	108

## ABBREVIATIONS

DAG	Directed Acyclic Graph
NLP	Natural Language Processing
CRF	Conditional Random Fields
POS	Part of Speech
NER	Named Entity Recognition
ELBO	Evidence Lower Bound
EM	Expectation-Maximization
MEMM	Maximum Entropy Markov Models
NCRFAE	Neural CRF Autoencoder
VAE	Variational Autoencoder
SRL	Semantic Role Labeling
LAP	Local Autoencoding Parser
GAP	Global Autoencoding Parser
HMM	Hidden Markov Model
DNN	Deep Neural Networks
PCFG	Probabilistic Context-Free Grammar
LSTM	Long Short Term Memory network
RNN	Recurrent Neural Network

## ABSTRACT

Zhang, Xiao Ph.D., Purdue University, May 2020. Flexible Structured Prediction in Natural Language Processing with Partially Annotated Corpora. Major Professor: Dan Goldwasser.

Structured prediction makes coherent decisions as structured objects to present the interrelations of these predicted variables. They have been widely used in many areas, such as bioinformatics, computer vision, speech recognition, and natural language processing. Machine Learning with reduced supervision aims to leverage the laborious and error-prone annotation effects and benefit the low-resource languages. In this dissertation we study structured prediction with reduced supervision for two sets of problems, sequence labeling and dependency parsing, both of which are representatives of structured prediction problems in NLP. We investigate three different approaches.

The first approach is learning with modular architecture by task decomposition. By decomposing the labels into location sub-label and type sub-label, we designed neural modules to tackle these sub-labels respectively, with an additional module to infuse the information. The experiments on the benchmark datasets show the modular architecture outperforms existing models and can make use of partially labeled data together with fully labeled data to improve on the performance of using fully labeled data alone.

The second approach builds the neural CRF autoencoder (NCRFAE) model that combines a discriminative component and a generative component for semi-supervised sequence labeling. The model has a unified structure of shared parameters, using different loss functions for labeled and unlabeled data. We developed a variant of the EM algorithm for optimizing the model with tractable inference. The experiments

on several languages in the POS tagging task show the model outperforms existing systems in both supervised and semi-supervised setup.

The third approach builds two models for semi-supervised dependency parsing, namely local autoencoding parser (LAP) and global autoencoding parser (GAP). LAP assumes the chain-structured sentence has a latent representation and uses this representation to construct the dependency tree, while GAP treats the dependency tree itself as a latent variable. Both models have unified structures for sentence with and without annotated parse tree. The experiments on several languages show both parsers can use unlabeled sentences to improve on the performance with labeled sentences alone, and LAP is faster while GAP outperforms existing models.



## 1 INTRODUCTION

Structured prediction makes coherent decisions as structured objects based on given input(s). Predicting such structured output, rather than a single discrete or real value, is characterized by the interrelations of these predicted variables. These interrelations are usually represented as functional dependencies in different domains, which means the prediction of one output is heavily dependent on the decision of other output(s).

Structured prediction naturally emerges from many real-world problems such that it has numerous applications. For instance, in bioinformatics, RNA sequences have chain-like structure and their secondary structures are modeled as tree-like structures. In computer vision (CV), object recognition on an image depends on the recognition of other objects on the same image. In speech recognition, articulation in sequential sentences are constrained by its preceding token(s) and the succeeding one(s) as well. Particularly in the field this dissertation will focus on, natural language processing (NLP), structured output is a prominent phenomenon. Several problem arises under the guise of sequence tagging, as the input data are often sequences that are sentences of text. Part of Speech (POS), as an example, describes the class label of tokens in a sentence, expressing the type of each of them. Dependency tree is another form of structured output in NLP, where sentences are translated into syntactic trees that are directed acyclic graph (DAG) with a single root. These trees depict the relationships among the words in the sentence, despite the absolute order of them as a sequence. Single decisions in both POS and dependency trees rely on other decisions around them.

Structured prediction is a special area of machine learning, and it faces particular challenges notoriously known to researchers. Generally speaking, it is difficult to train structured prediction models, and the inference process lacks scalability. In machine learning, researchers built intelligent algorithms which are able to generalize from

previously seen examples. Traditionally, in the case of classification, it categorizes newly met examples by learning from past examples; while in the case of regression, it predicts a scalar number from an input based on the historical data. Structured prediction furthers traditional machine learning to the scenarios of structured output where the predictions are structured objects with coherence within themselves. Structured prediction can capture the inter relationships among individual predictions within each structured output and ensure the inner constraints satisfied for each structured object. However, it suffers more from the difficult of training, due to the non-convexity of the loss functions and the complexity of the models themselves in most cases. This hardness is compounded as the training data are scarce, caused by the fact that the annotation of structured output is difficult, and the common supervised learning method for structured prediction models requires large amount of data. The issue of lacking of annotated data particularly lies in the domain of NLP. Human annotation for languages is a laborious process and error-prone. Language understanding and annotation is a high level cognitive task rather than a perceptual level task: not only linguistic expertise is required for accurate annotation, but also heavy cognitive load is borne to the annotators, since the task requires knowledge with correct grammatical structure and/or semantic meanings. This poses an extreme high cost for hiring and training these annotators, and may result in huge hiring expense. Additionally, this results in substantially limited quality, quantity and diversity of the annotated data, which are used as training corpora in NLP. These difficulties especially aggravate for low-resourced languages, as the availability of well-annotated corpora is always a concern. The scarcity of annotated corpora demands new methods that requires less annotation, is able to utilize partially annotated and/or unannotated corpora. Therefore, it makes learning with reduced supervision a desirable way for structured prediction. In this dissertation, we focus on alternative forms of supervision, aiming to reduce the amount of supervision needed for structured prediction task.

In NLP, many problems are cast as structured prediction, from syntax to semantics, organized into a hierarchy. At the bottom of the hierarchy lie pure syntactic problems such as chunking (shallow parsing), POS tagging, dependency parsing, constituent parsing and so on. Higher in the hierarchy, we have shallow semantic tasks such as NER, SRL, sentiment tagging, co-reference resolution and so on. These specific tasks are usually disguised as structured prediction problems. In this dissertation we study sequence labeling including POS tagging, NER and sentiment tagging and parsing with dependency grammar. Sequence labeling and dependency parsing can be regarded as representatives in structured prediction in NLP, and techniques to tackle them may shed light onto other tasks as well. There has been a significant amount of work in studying structured prediction with reduced supervised for sequence labeling and parsing, but new techniques are in demand to fulfill the room that exists for improvement.

### 1.1 Three Types of Approaches to Structured Prediction with Reduced Supervision

Existing common approaches for structured prediction in NLP with reduced supervision majorly fall into three categories as follows:

1. The multi-task and modular approaches try to build models that handle several tasks simultaneously. These approaches usually either aggregate the loss functions of several structured prediction tasks at the output level, or combine a language model at the input level.
2. The sequence labeling with latent variables approaches deal with chain-structured prediction problems by treating the output as latent variables. In a lot of cases, since the chain-structure is presumed the same as the sentence chain, the latent variables are the labels of each tokens in the sentence.
3. The tree induction with latent variables approaches advance chain-structured prediction problems to trees, where the structures are more complex. Due to

the high complexity of syntactic trees, which are actually DAGs, the latent variables usually are the edges in the trees.

The multi-task and modular approaches try to solve structured prediction with reduced supervision by combining multiple NLP tasks together with delicately designed model architectures. However, these approaches have difficulties to deal with data without any annotation. Sequence labeling with latent variables approaches try to solve structured prediction with reduced supervision by relying on statistical models able to handle incomplete data. These approaches are mostly based on generative model frameworks, particularly chain-structured models such as HMM and its variants. The tree induction with latent variables approaches extend from chain-structured representations to more refined and more expressive tree-structured ones but of higher complexity. Based on probabilistic grammars, these approaches try to learn the rule probability, which are the parameters of the grammar.

## 1.2 Related Work

### 1.2.1 Learning with Multi-task and Modular Architecture

Caruana [1997] first described multi-task in detail. Later, Collobert et al. [2011] first successfully applied it to NLP by building a model to various NLP tasks. Eriguchi et al. [2017] and Luong [2016] applied multi-task to machine translation by aggregating auxiliary tasks at the output level in addition to the target task. On the other hand, Toshniwal et al. [2017] and Liu et al. [2018] employed low-level auxiliary tasks at the bottom input level. Modular architectures have been studied in the interaction of CV and NLP, e.g., Andreas et al. [2016] applied a modular model to both images and structured NLP knowledge bases, Hu et al. [2017] proposed End-to-End Module Networks that learn to reason and Yu et al. [2018] described the Modular Attention Network to capture different type of attentions.

### 1.2.2 Sequence Labeling with Latent Variables

Church [1988] and DeRose [1988] first applied HMM to the POS tagging problem. Merialdo [1994] and Elworthy [1994] trained HMMs from labeled data, and used the resulting models as initialization for EM on the unlabeled data, which is the prototype of semi-supervised sequence labeling. Toutanova and Johnson [2007] described a model for Bayesian semi-supervised POS tagging with the Dirichlet distribution to encourage sparsity in the parameter space. Method of moments has also been applied to semi-supervised sequence labeling [Marinho et al., 2016]. Brown cluster and other word sense resources have been shown to be helpful [Stratos and Collins, 2015b]. In addition, self-training, co-training, tri-training and cross-view training as a group of similar techniques are also important roles in semi-supervised sequence labeling [Brill, 1995, Collins and Singer, 1999, Ruder and Plank, 2018, Clark et al., 2018]. Recent attempts by using neural networks are related to generative models [Tran et al., 2016, He et al., 2018, Chen et al., 2018]. Autoencoder related techniques are also important parts of sequence labeling with latent variables approaches [Ammar et al., 2014, Lin et al., 2015, Cheng et al., 2017]. A recent work applied mutual information maximization for POS induction [Stratos, 2019].

### 1.2.3 Tree Induction with Latent Variables

Lari and Young [1990] and Baker [1979] first proposed the inside-outside algorithm for learning PCFG, to build the constituent trees through corpus. EM algorithm is popularized for tree induction, e.g., Klein and Manning [2004] proposed the a variant called DMV (dependency model with valence) to mitigate local traps of the original EM through a good initialization, while Spitzkovsky et al. [2010b] introduced Viterbi EM as a degenerated EM which achieves better results. Tree induction with latent variables can benefit from additional resources such as word clusters [Koo et al., 2008] and statistics of contextual features from larger corpus [Kiperwasser and Goldberg, 2015]. Proper regularization can help the model learn the grammar of

the tree towards the correct direction, e.g., the (un)ambiguity regularization [Tu and Honavar, 2012] and the posterior regularization [Ganchev et al., 2010, Gillenwater et al., 2010]. Several recent attempts using neural networks have also shown the improvement, owing to the expressivity of DNN [Jiang et al., 2016, Han et al., 2019, Shen et al., 2018b,a]. Autoencoder and its variants have shown their strength of combining the discriminative and the generative component [Cai et al., 2017, Corro and Titov, 2018].

#### 1.2.4 Limitations

Though there has been a large body of work tackling structured predictions in NLP with unannotated corpora, most of them either rely on additionally fully annotated data, while the others focus on unsupervised learning, which implies the space for combining these two to reduce the supervision efforts. Second, most of the work either chose to apply the discriminative learning framework or stayed with the generative learning framework, except very few, neglecting the potential of combining these two approaches. At last but not the least, though DNN has been prevailed in machine learning in recent years, very few work investigated the possibility of incorporating DNN to reduce supervision with both annotated and unannotated data.

#### 1.2.5 Other Related Areas

**Structured Prediction** Traditional structured prediction models include global inference based generative model as hidden Markov models (HMM) [Baum and Petrie, 1966] and discriminative model as maximum entropy Markov models (MEMM) [McCallum et al., 2001] and structured perceptrons [Collins, 2002], globally normalized linear models as Conditional Random Fields (CRF) [Lafferty et al., 2001] and structured large-margin approaches with kernel method as structured support vector machine (SSVM) [Tsochantaridis et al., 2004]. Additionally in the SEARN algorithm, Daumé III et al. [2009] cast learning in structured prediction as a search problem.

**Deep Neural Networks** The renovation of DNN in recent years has led to a huge leap in several machine learning tasks. Word embeddings [Mikolov et al., 2013] and its variants [Pennington et al., 2014], unleashed the power of neural networks by introducing low-dimensional dense input representations to replace the traditional high-dimensional sparse input representation of text data, making DNN suitable models for NLP. Several recent works have shown promising results when combining structured prediction models with deep neural networks. A few studies on sequence tagging using neural networks achieved state-of-the-art performance on the benchmark datasets [Ma and Hovy, 2016, Mesnil et al., 2015, Lample et al., 2016]. Deep neural network architectures together with word embeddings have also led to improved performance in both graph-based and transition-based dependency parsing [Nivre, 2014, Pei et al., 2015, Chen and Manning, 2014, Dyer et al., 2015, Weiss et al., 2015, Kiperwasser and Goldberg, 2016]. Andor et al. [2016] showed a globally normalized transition-based neural network model performs well in several structured prediction tasks, including part of speech (POS) tagging, dependency parsing and sentence compression. Durrett and Klein [2015] proposed a globally normalized neural CRF model for graph-based constituent parsing, using neural networks to generate potentials in the model. Wiseman and Rush [2016] extended Daumé III et al. [2009]’s work by building a sequence-to-sequence based neural language model with a beam-search training scheme. In addition, the Adaptive Resonance Theory (ART) [Grossberg, 2013] is also related to the modular architecture.

**Semi-supervised Learning in NLP** Self-training is historically regarded as the oldest approach to semi-supervised learning [Chapelle et al., 2010] and has been widely used in NLP [Yarowsky, 1995, Riloff et al., 2003]. Nigam et al. [2000] successfully applied a generative model using EM algorithm on semi-supervised text classification. Several recent works on semi-supervised learning in NLP include deep generative models for sentence compression [Miao and Blunsom, 2016], adversarial training for text classi-

fication [Miyato et al., 2016], latent variable for semantic parsing [Yin et al., 2018] and recursive autoencoders to predict sentiment distributions [Socher et al., 2011].

**Bayesian Statistics in NLP** Several modern work has adopted the Bayesian statistics framework for adding pre-designed prior as inductive bias to help learning. Dirichlet prior is a popular choice as it can induce strong sparsity, and researchers have applied it with variational inference [Kurihara and Sato, 2006] or MCMC [Goldwater and Griffiths, 2007, Johnson et al., 2007]. In addition, the Dirichlet process prior was proposed by Liang et al. [2007] and Finkel et al. [2007] to produce a smaller grammar size without fixing a number of nonterminal types in PCFG. A different type of prior named logistic normal prior has been used to model the correlations between grammar symbols [Cohen et al., 2008, Cohen and Smith, 2009].

**Cognitive Science and Structured Prediction** Structured prediction is also related to language acquisition in psycholinguistics [Pearl et al., 2010], as shown by evidence, human children learn the words' order and grammar implicitly [Goldwater, 2006, Abend et al., 2017, Pearl and Goldwater, 2016]. Further, structured prediction is also related to Gestalt psychology in cognitive science. In Gestalt psychology, the Principle of Totality elaborates that the conscious experience must be considered globally (by taking into account all the physical and mental aspects of the individual simultaneously) because the nature of the mind demands that each component be considered as part of a system of dynamic relationships. This principle also explains the local and global relationship in structured prediction. Gobet [2016] explained how these rules influence the chunking in languages. In addition, Universal Grammar (UG) [Chomsky, 2007] and Language Acquisition Device (LAD) [Chomsky, 1965] are also related to this topic. As we are constructing mathematical model with the same prototype to learn different languages and to discover a universal grammar set among all languages, evidence are being provided for UG and LAD.



### 1.3 Dissertation Overview

In this dissertation we propose three novel learning approaches with reduced supervision for structured prediction in NLP. Our first approach looks at learning with partial labels by decomposing complex prediction tasks into sub-tasks in sequence labeling tasks, including sentiment tagging and named entity recognition [Zhang and Goldwasser, 2019]. First the labels are decomposed into partial labels—“location” and “type”—that are easier to annotate. Then we build the models with two modules which are trained separately by using the partial labels respectively. An additional module is designed to integrate the two modules to make the final prediction. The advantage is that the model can be trained using partially annotated data if fully labeled data is unavailable, since partially annotated data is relative cheaper and easier to obtain in the real world. Our experiments on the benchmark datasets show the modular architecture outperforms existing models and can make use of partially labeled data together with fully labeled data to improve on the performance using fully labeled data alone.

Modular architecture, however, cannot directly deal with unlabeled data, which is particularly common in low-resource languages. In statistics, one thought to deal with incomplete data is to use the Expectation-Maximization (EM) algorithm [Dempster et al., 1977]. Our second approach expends the EM algorithm by building a model called neural CRF autoencoder (NCRFAE) that combines a discriminative component and a generative component, where the discriminative component uses the input to predict the output and the generative component tries to reconstruct the input based on the output [Zhang et al., 2017]. This model has a unified structure of shared parameters but different loss functions for labeled and unlabeled data. We develop a variant of the EM algorithm to optimize the model with tractable inference. Our experiments on several languages in the POS tagging task show the model outperforms existing systems in both supervised and semi-supervised setup.

Natural languages have more complicated structures than sequences. Our third approach tries to tackle dependency grammar, in which each sentence is represented as a dependency tree. We proposed two different models for semi-supervised dependency parsing, namely local autoencoding parser (LAP) and global autoencoding parser (GAP). LAP assumes the chain-structured sentence has a latent representation and uses this representation to construct the dependency tree, while GAP treats the dependency tree itself as a latent variable. Both models have unified structures for data with and without the annotated parse trees. Our experiments on several languages show both parsers can use unlabeled sentences to improve on the performance using labeled sentences alone, and LAP is faster while GAP outperforms existing models.

### 1.3.1 Dissertation Organization

We organize the rest of this dissertation as follows.

In Chapter 2, we introduce problem definitions and preliminary concepts.

In Chapter 3, we study learning with partial labels in task decomposition based on a cognitively inspired neural model with modular architecture.

In Chapter 4, we present our semi-supervised NCRFAE model for sequence labeling, along with an innovative algorithm that extends the EM algorithm to deal with unlabeled data.

In Chapter 5, we extend our semi-supervised learning approach to dependency parsing, with two models. The first model LAP assumes the sentence has a latent representation to construct the dependency tree; the second model GAP directly treats the parse tree as a latent variable and reconstructs the input from the parse tree.

In Chapter 6, we conclude the dissertation with a summary of contributions and provide potential future research directions.

## 2 PRELIMINARIES

In this chapter, we briefly review dependency graphs and trees, variational autoencoders (VAE) and tree CRFs.

### 2.1 Dependency Graphs and Trees

**Definition 2.1.1 Sentence:** A sentence of length  $l$  including punctuation is a sequence of tokens denoted by:

$$\mathbf{s} = w_0 w_1 \dots w_l. \quad (2.1)$$

Particularly,  $w_0 = \text{ROOT}$  is an artificial root token inserted at the beginning of the sentence.

**Definition 2.1.2 Relation Types:** Define  $\mathbf{r} = r_1 \dots r_m$  to be a finite set of possible dependency relation types that connects any two tokens in a sentence. A relation type  $r \in \mathbf{r}$  is called an arc label.

**Definition 2.1.3 Dependency Graph:** A dependency graph  $\mathbf{g} = (\mathbf{v}, \mathbf{a})$  is a labeled directed graph in the standard graph-theoretic sense and consists of nodes,  $\mathbf{v}$ , and arcs,  $\mathbf{a}$ , such that for sentence  $\mathbf{s} = w_0 w_1 \dots w_l$  and label set  $\mathbf{r}$  we have the following:

1.  $\mathbf{v} \subseteq w_0 w_1 \dots w_l$
2.  $\mathbf{a} \subseteq \mathbf{v} \times \mathbf{r} \times \mathbf{v}$
3. if  $(w_i, r, w_j) \in \mathbf{a}$  then  $(w_i, r', w_j) \notin \mathbf{a} \forall r' \neq r$ .

The arc set  $\mathbf{a}$  represents the labeled dependency relations of the particular graph  $\mathbf{g}$ . Specifically, an arc  $(w_i, r, w_j)$  represents a dependency relation from head  $w_i$  to

dependent  $w_j$  labeled with relation type  $r$ . A dependency graph  $\mathbf{g}$  is thus a set of labeled dependency relations between the words of  $\mathbf{s}$ .

In this dissertation, we focus on unlabeled arc construction so we only consider arcs but not arc types.

## 2.2 Variational Autoencoder (VAE)

The typical VAE is a directed graphical model with latent variables, denoted by  $\mathbf{z}$ . A generative process first generates latent variable  $\mathbf{z}$  from the prior distribution  $\pi(\mathbf{z})$  and the data  $\mathbf{x}$  is recovered from the distribution  $P_\theta(\mathbf{x}|\mathbf{z})$ , parameterized by  $\theta$ . There is also an inference model  $Q(\mathbf{z}|\mathbf{x})$ , which is an auxiliary posterior distribution, used for inferring the most probable latent variable  $\mathbf{z}$  given the input  $\mathbf{x}$ . In our scenario,  $\mathbf{x}$  is an input sequence and  $\mathbf{z}$  is a sequence of latent variables correspondingly.

The VAE framework seeks to maximize the complete log-likelihood  $\log P(\mathbf{x})$  by marginalizing out the latent variable  $\mathbf{z}$ . Since direct parameter estimation of  $\log P(\mathbf{x})$  is usually intractable, a common solution is to maximize its Evidence Lower Bound (ELBO).

## 2.3 Tree Conditional Random Field

The linear chain CRF models an input sequence  $\mathbf{x} = (x_1 \dots x_l)$  of length  $l$  with labels  $\mathbf{y} = (y_1 \dots y_l)$  with globally normalized probability

$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp \mathcal{S}(\mathbf{x}, \mathbf{y})}{\sum_{\tilde{\mathbf{y}} \in \mathcal{Y}} \exp \mathcal{S}(\mathbf{x}, \tilde{\mathbf{y}})},$$

where  $\mathcal{Y}$  is the set of all the possible label sequences, and  $\mathcal{S}(\mathbf{x}, \mathbf{y})$  the scoring function, usually decomposed as emission ( $\sum_{i=1}^l s(x_i, y_i)$ ) and transition ( $\sum_{i=1}^l s(y_i, y_{i+1})$ ) for first-order models.

The tree CRF model generalizes linear chain CRF to trees. In dependency parsing, the tree CRF model tries to resolve which node pairs should be connected with directed edges, such that the set of edges form a tree. The potentials in the depen-

dependency tree take an exponential form, thus the conditional probability of a parse tree  $\mathcal{T}$ , given the input sentence, can be denoted as:

$$P(\mathcal{T}|\mathbf{x}) = \frac{\exp \mathcal{S}(\mathbf{x}, \mathcal{T})}{Z(\mathbf{x})}, \quad (2.2)$$

where  $Z(\mathbf{x}) = \sum_{\tilde{\mathcal{T}} \in \mathbb{T}(\mathbf{x})} \exp \mathcal{S}(\mathbf{x}, \tilde{\mathcal{T}})$  is the partition function that sums over all possible valid dependency trees in the set  $\mathbb{T}(\mathbf{x})$  of the given sentence  $\mathbf{x}$ .

### 3 MODULAR NEURAL ARCHITECTURE FOR STRUCTURED PREDICTION WITH REDUCED SUPERVISION

In this chapter, we first study the annotation types systematically in sentiment tagging and NER and found the labels can be decomposed into sub-labels, while some sub-labels are cognitively easier for annotators to recognize and annotate than the full labels. Modern studies in biological neuroscience and computational neuroscience have revealed the brain processes information using two different paths: the “where” and the “what” paths, and we found sentiment tagging and NER can also be decomposed by “location” in the sentence and “type” of the chunk.

To model the “where” locator and the “what” recognizer, we designed the computational model based on this important discovery by proposing two individual modules for the “where” and “what” tasks separately, and a third module to integrate these two pieces of information to make the final prediction. In addition, the “where” and “what” module can be trained separately by using partially labeled data, in which only the location or the type information is annotated. We also investigated three different ways of connecting the two modules with the decision module and found one of these variants has the best performance.

Our experiments on different datasets show the model can make good use of the information in the partially labeled sentences, together with the fully labeled, to improve the overall accuracy of sentiment tagging and NER prediction for unseen text.

#### 3.1 Introduction

Many natural language processing tasks attempt to replicate complex human-level judgments, which often rely on a composition of several sub-tasks into a unified judg-

ment. For example, in the Targeted-Sentiment task [Mitchell et al., 2013], sentiment polarity scores are assigned to entities depending on the context that they appear in. Given the sentence “according to a CNN poll, Green Book will win the best movie award”, the system has to identify both entities, and associate the relevant sentiment value with each one (neutral with CNN, and positive with Green Book). This task can be viewed as a combination of two tasks, entity identification, locating contiguous spans of words corresponding to relevant entities, and sentiment prediction, specific to each entity based on the context it appears in. Despite the fact that this form of functional task decomposition is natural for many learning tasks, it is typically neglected and learning is defined as a monolithic process, combining the tasks into a single learning problem.

Our goal in this study is to take a step towards modular learning architectures that exploit the learning tasks’ inner structure, and as a result to simplify the learning process and to reduce the annotation effort. We introduce a novel task decomposition approach, learning with partial labels, in which the task output labels decompose hierarchically, into partial labels capturing different aspects, or sub-tasks, of the final task. We show that learning with partial labels can help support weakly-supervised learning when only some of the partial labels are available.

Given the popularity of sequence labeling tasks in NLP, we demonstrate the strength of this approach over several sentiment analysis tasks, adapted for sequence prediction. These include target-sentiment prediction [Mitchell et al., 2013], aspect-sentiment prediction [Pontiki et al., 2016] and subjective text span identification and polarity prediction [Nakov et al., 2013]. To ensure the broad applicability of our approach to other problems, we extend the popular LSTM-CRF [Lample et al., 2016] model that was applied to many sequence labeling tasks\*.

The modular learning process corresponds to a task decomposition, in which the prediction label,  $y$ , is deconstructed into a set of partial labels  $\{y^0, \dots, y^k\}$ , each defining a sub-task, capturing a different aspect of the original task. Intuitively, the individ-

---

\*We also provide analysis for NER in the appendix

ual sub-tasks are significantly easier to learn, suggesting that if their dependencies are modeled correctly when learning the final task, they can constrain the learning problem, leading to faster convergence and a better overall learning outcome. In addition, the modular approach helps alleviate the supervision problem, as often providing full supervision for the overall task is costly, while providing additional partial labels is significantly cheaper. For example, annotating entity segments syntactically is considerably easier than determining their associated sentiment, which requires understanding the nuances of the context they appear in semantically. By exploiting modularity, the entity segmentation partial labels can be used to help improve that specific aspect of the overall task.

Our modular task decomposition approach is partially inspired by findings in cognitive neuroscience, namely the two-streams hypothesis, a widely accepted model for neural processing of cognitive information in vision and hearing [Eysenck and Keane, 2005], suggesting the brain processes information in a modular way, split between a “where” (dorsal) pathway, specialized for locating objects and a “what” (ventral) pathway, associated with object representation and recognition [Mishkin et al., 1983, Geschwind and Galaburda, 1987, Kosslyn, 1987, Rueckl et al., 1989]. Jacobs et al. [1991] provided a computational perspective, investigating the “what” and “where” decomposition on a computer vision task. We observe that this task decomposition naturally fits many NLP tasks and borrow the notation. In the target-sentiment tasks we address in this study, the segmentation tagging task can be considered as a “where”-task (i.e., the location of the entities), and the sentiment recognition as the “what”-task (i.e., the type of the entities).

Our approach is related to multi-task learning [Caruana, 1997], which has been extensively applied in NLP [Toshniwal et al., 2017, Eriguchi et al., 2017, Collobert et al., 2011, Luong, 2016, Liu et al., 2018]. However, instead of simply aggregating the objective functions of several different tasks, we suggest to decompose a single task into multiple inter-connected sub-tasks and then integrate the representation learned into a single module for the final decision. We study several modular neural



architectures, which differ in the way information is shared between tasks, the learning representation associated with each task and the way the dependency between decisions is modeled.

Our experiments were designed to answer two questions. First, can the task structure be exploited to simplify a complex learning task by using a modular approach? Second, can partial labels be used effectively to reduce the annotation effort?

To answer the first question, we conduct experiments over several sequence prediction tasks, and compare our approach to several recent models for deep structured prediction [Lample et al., 2016, Ma and Hovy, 2016, Liu et al., 2018], and when available, previously published results [Mitchell et al., 2013, Zhang et al., 2015, Li and Lu, 2017, Ma et al., 2018a] We show that modular learning indeed helps simplify the learning task compared to traditional monolithic approaches. To answer the second question, we evaluate our model’s ability to leverage partial labels in two ways. First, by restricting the amount of full labels, we have observed the improvement when providing increasing amounts of partial labels for only one of the sub-tasks. Second, we learn the sub-tasks using completely disjoint datasets of partial labels, and show that the knowledge learned by the sub-task modules can be integrated into the final decision module using a small amount of full labels. As we demonstrate in our experiments, this approach leads to better performance and increased flexibility, as it allows us to decouple the learning process and learn the tasks independently.

Our contributions: (1) We provide a general modular framework for sequence learning tasks. While we focus on sentiment tagging task, the framework is broadly applicable to many other tagging tasks, for example, NER [Carreras et al., 2002, Lample et al., 2016] and SRL [Zhou and Xu, 2015], to name a few. (2) We introduce a novel weakly supervised learning approach, learning with partial labels, which exploits the modular structure to reduce the supervision effort. (3) We evaluated our proposed model, in both the fully-supervised and weakly supervised scenarios, over several sentiment analysis tasks.

### 3.2 Related Works

From a technical perspective, our task decomposition approach is related to multi-task learning [Caruana, 1997], specifically, when the tasks share information using a shared deep representation [Collobert et al., 2011, Luong, 2016]. However, most prior works aggregate multiple losses on either different pre-defined tasks at the final layer [Collobert et al., 2011, Luong, 2016], or on a language model at the bottom level [Liu et al., 2018]. This work suggests to decompose a given task into sub-tasks whose integration comprise the original task. To the best of our knowledge, Ma et al. [2018a], focusing on targeted sentiment is the most similar to our approach. They suggest a joint learning approach, modeling a sequential relationship between two tasks, entity identification and target sentiment. We take a different approach viewing each of the model components as a separate module, predicted independently and then integrated into the final decision module.

Other modular neural architectures were recently studied for tasks combining vision and language analysis [Andreas et al., 2016, Hu et al., 2017, Yu et al., 2018], and were tailored for the grounded language setting. To help ensure the broad applicability of our framework, we provide a general modular network formulation for sequence labeling tasks by adapting a neural-CRF to capture the task structure. This family of models, combining structured prediction with deep learning have showed promising results [Gillick et al., 2015, Lample et al., 2016, Ma and Hovy, 2016, Zhang et al., 2015, Li and Lu, 2017], by using rich representations through neural models to generate decision candidates, while utilizing an inference procedure to ensure coherent structured decisions. Our main observation is that modular learning can help alleviate some of the difficulty involved in training these powerful models.

### 3.3 Architectures for Sequence Prediction

Using neural networks to generate emission potentials in CRFs has been successfully applied in several sequence prediction tasks, such as word segmentation [Chen

et al., 2017], NER [Ma and Hovy, 2016, Lample et al., 2016], chunking and POS tagging [Liu et al., 2018, Zhang et al., 2017]. A sequence is represented as a sequence of  $l$  tokens:  $\mathbf{x} = [x_1, x_2, \dots, x_l]$ , in which each token corresponds to a label  $y \in \mathcal{Y}$ , where  $\mathcal{Y}$  is the set of all possible tags. An inference procedure is designed to find the most probable sequence  $\mathbf{y}^* = [y_1, y_2, \dots, y_L]$  by solving, either exactly or approximately, the following optimization problem:

$$\mathbf{y}^* = \arg \max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}).$$

Despite the difference in tasks, these models follow a similar general architecture: (1) Character-level information, such as prefix, suffix and capitalization, is represented through a character embedding layer using a bi-directional LSTM (Bi-LSTM). (2) Word-level information is obtained through a word embedding layer. (3) The two representations are concatenated to represent an input token, used as input to a word-level Bi-LSTM which generates the emission potentials for a succeeding CRF. (4) The CRF is used as an inference layer to generate the globally-normalized probability of possible tag sequences.

### 3.3.1 CRF Layer

A CRF model describes the probability of predicted labels  $\mathbf{y}$ , given a sequence  $\mathbf{x}$  as input, as

$$P_{\Lambda}(\mathbf{y}|\mathbf{x}) = \frac{e^{\Phi(\mathbf{x}, \mathbf{y})}}{Z},$$

where  $Z = \sum_{\tilde{\mathbf{y}}} e^{\Phi(\mathbf{x}, \tilde{\mathbf{y}})}$  is the partition function that marginalize over all possible assignments to the predicted labels of the sequence, and  $\Phi(\mathbf{x}, \mathbf{y})$  is the scoring function, which is defined as:

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_t \phi(\mathbf{x}, y_t) + \psi(y_{t-1}, y_t).$$

The partition function  $Z$  can be computed efficiently via the forward-backward algorithm. The term  $\phi(\mathbf{x}, y_t)$  corresponds to the emission score of a particular tag  $y_t$

at position  $t$  in the sequence, and  $\psi(y_{t-1}, y_t)$  represents the transition score from the tag at position  $t - 1$  to the tag at position  $t$ . In the Neural CRF model,  $\phi(\mathbf{x}, y_t)$  is generated by the aforementioned Bi-LSTM while  $\psi(y_{t-1}, y_t)$  by a transition matrix.

### 3.4 Functional Decomposition of Composite Tasks

To accommodate our task decomposition approach, we first define the notion of partial labels, and then discuss different neural architectures capturing the dependencies between the modules trained over the different partial labels.

**Partial Labels and Task Decomposition:** Given a learning task, defined over an output space  $y \in \mathcal{Y}$ , where  $\mathcal{Y}$  is the set of all possible tags, each specific label  $y$  is decomposed into a set of partial labels,  $\{y^0, \dots, y^k\}$ . We refer to  $y$  as the full label. According to this definition, a specific assignment to all  $k$  partial labels defines a single full label. Note the difference between partially labeled data [Cour et al., 2011], in which instances can have more than a single full label, and our setup in which the labels are partial.

In all our experiments, the partial labels refer to two sub-tasks, (1) a segmentation task, identifying Beginning, Inside and Outside of an entity or aspect. (2) one or more type recognition tasks, recognizing the aspect type and/or the sentiment polarity associated with it. Hence, a tag  $y_t$  at location  $t$  is divided into  $y_t^{seg}$  and  $y_t^{typ}$ , corresponding to segmentation and type (sentiment type here) respectively. Fig. 3.1 provides an example of the target-sentiment task. Note that the sentiment labels do not capture segmentation information.

**Modular Learning architectures:** We propose three different models, in which information from the partial labels can be used. All the models have similar modules types, corresponding to the segmentation and type sub-tasks, and the decision module for predicting the final task. The modules are trained over the partial segmentation ( $\mathbf{y}^{seg}$ ) and type ( $\mathbf{y}^{typ}$ ) labels, and the full label  $\mathbf{y}$  information, respectively.

<b>Text</b>	ABC	News'	Christiane	Amanpour	Exclusive	Interview	with	President	Mubarak
<b>Tag</b>	B-neu	E-neu	B-neu	E-neu	O	O	O	B-neu	E-neu
<b>Seg</b>	B	E	B	E	O	O	O	B	E
<b>Senti</b>	neu	neu	neu	neu	O	O	O	neu	neu

Figure 3.1.: This figure shows a target-sentiment decomposition example with segmentation and sentiment.

These three models differ in the way they share information. Model 1, denoted Twofold Modular, LSTM-CRF-T, is similar in spirit to multi-task learning [Collobert et al., 2011] with three separate modules. Model 2, denoted Twofold modular Infusion, (LSTM-CRF-TI) and Model 3, denoted Twofold modular Infusion with guided gating, (LSTM-CRF-TI(g)) both infuse information flow from two sub-task modules into the decision module. The difference is whether the infusion is direct or going through a guided gating mechanism. The three models are depicted in Fig. 3.2 and described in details in the following paragraphs. In all of these models, the underlying neural architectures are used for the emission potentials when the CRF inference layers are applied on top.

### 3.4.1 Twofold Modular Model

The twofold modular model enhances the original monolithic model by using multi-task learning with shared underlying representations. The segmentation module and the type module are trained jointly with the decision module, and all the modules share information by using the same embedding level representation, as shown in Figure 3.2a. Since the information above the embedding level is independent, the LSTM layers in the different modules do not share information, so we refer to these layers of each module as private.

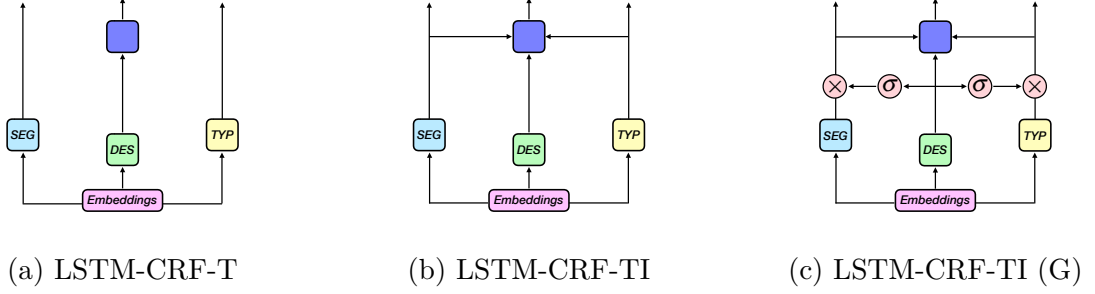


Figure 3.2.: This figure shows three modular models for task decomposition. In all of them, blue blocks are the segmentation modules, detecting entity location and segmentation, and yellow blocks are the type modules, recognizing the entity type or sentiment polarity. Green blocks are the final decision modules, integrating all the decisions. (G) refers to “Guided Gating”.

The segmentation module predicts the segmentation BIO labels at position  $t$  of the sequence by using the representations extracted from its private word level bi-LSTM (denoted as  $\mathcal{H}^{seg}$ ) as emission for a individual CRF:

$$\mathbf{h}_t^{seg} = \mathcal{H}^{seg}(e_t, \vec{\mathbf{h}}_{t-1}^{seg}, \vec{\mathbf{h}}_{t+1}^{seg}),$$

$$\phi(\mathbf{x}, y_t^{seg}) = \mathbf{W}^{seg\top} \mathbf{h}_t^{seg} + \mathbf{b}^{seg},$$

where  $\mathbf{W}^{seg}$  and  $\mathbf{b}^{seg}$  denote the parameters of the segmentation module emission layer, and  $\mathcal{H}^{seg}$  denotes its private LSTM layer.

This formulation allows the model to forge the segmentation path privately through back-propagation by providing the segmentation information  $\mathbf{y}^{seg}$  individually, in addition to the complete tag information  $\mathbf{y}$ .

The type module, using  $\mathbf{y}^{typ}$ , is constructed in a similar way. By using representations from its own private LSTM layers, the type module predicts the sentiment (entity) type at position  $t$  of the sequence :

$$\mathbf{h}_t^{typ} = \mathcal{H}^{typ}(e_t, \vec{\mathbf{h}}_{t-1}^{typ}, \vec{\mathbf{h}}_{t+1}^{typ}),$$

$$\phi(\mathbf{x}, y_t^{typ}) = \mathbf{W}^{typ\top} \mathbf{h}_t^{typ} + \mathbf{b}^{typ}.$$

Both the segmentation information  $\mathbf{y}^{seg}$  and the type information  $\mathbf{y}^{typ}$  are provided together with the complete tag sequence  $\mathbf{y}$ , enabling the model to learn segmentation and type recognition simultaneously using two different paths. Also, the decomposed tags naturally augment more training data to the model, avoiding over-fitting due to more complicated structure. The shared representation beneath the private LSTMs layers are updated via the back-propagated errors from all the three modules.

### 3.4.2 Two-fold Modular Infusion Model

The twofold modular infusion model provides a stronger connection between the functionalities of the two sub-tasks modules and the final decision module, differing significantly from multi-task learning.

In this model, instead of separating the pathways from the decision module as in the previous twofold modular model, the segmentation and the type representation are used as input to the final decision module. The model structure is shown in Figure 3.2b, and can be described formally as:

$$\begin{aligned}\mathbf{i}_t^{seg} &= \mathbf{W}^{seg\top} \mathbf{h}_t^{seg} + \mathbf{b}^{seg}, \\ \mathbf{i}_t^{typ} &= \mathbf{W}^{typ\top} \mathbf{h}_t^{typ} + \mathbf{b}^{typ}, \\ \mathbf{s}_t &= \mathbf{W}^\top [\mathbf{h}_t; \mathbf{i}_t^{seg}; \mathbf{i}_t^{typ}] + \mathbf{b},\end{aligned}$$

where  $\mathbf{s}_t$  is the shared final emission potential to the CRF layer in the decision module, and  $;$  is the concatenation operator, combining the representation from the decision module and that from the type module and the segmentation module.

The term ‘‘Infusion’’ used for naming this module is intended to indicate that both modules actively participate in the final decision process, rather than merely form two independent paths as in the twofold modular model. This formulation provides an alternative way of integrating the auxiliary sub-tasks back into the major task in the neural structure to help improve learning.

### 3.4.3 Guided Gating Infusion

In the previous section we described a way of infusing information from other modules naively by simply concatenating them. But intuitively, the hidden representation from the decision module plays an important role as it is directly related to the final task we are interested in. To effectively use the information from other modules forming sub-tasks, we design a gating mechanism to dynamically control the amount of information flowing from other modules by infusing the expedient part while excluding the irrelevant part, as shown in Figure 3.2c. This gating mechanism uses the information from the decision module to guide the information from other modules, thus we name it as guided gating infusion, which we describe formally as follows:

$$\begin{aligned} \mathbf{i}_t^{seg} &= \sigma(\mathbf{W}_1 h_t + \mathbf{b}_1) \otimes (\mathbf{W}^{seg\top} \mathbf{h}_t^{seg} + \mathbf{b}^{seg}), \\ \mathbf{i}_t^{typ} &= \sigma(\mathbf{W}_2 h_t + \mathbf{b}_2) \otimes (\mathbf{W}^{typ\top} \mathbf{h}_t^{typ} + \mathbf{b}^{typ}), \\ \mathbf{s}_t &= \mathbf{W}^\top [\mathbf{h}_t; \mathbf{i}_t^{seg}; \mathbf{i}_t^{typ}] + \mathbf{b}, \end{aligned}$$

where  $\sigma$  is the logistic sigmoid function and  $\otimes$  is the element-wise multiplication. The  $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$  are the parameters of these gates for guiding, which are updated during the training to maximize the overall sequence labeling performance.

### 3.5 Learning with Full and Partial Labels

Our objective naturally rises from the model we described in the text. Furthermore, as our experiments show, it is easy to generalize this objective, to a “semi-supervised” setting, in which the model has access to only a few fully labeled examples and additional partially labeled examples. E.g., if only segmentation is annotated



but the type information is missing. The loss function is a linear combination of the negative log probability of each sub-tasks, together with the decision module:

$$\begin{aligned} \mathcal{J} = & - \sum_i^N \log P(\mathbf{y}^i | \mathbf{x}^i) + \alpha \log P(\mathbf{y}^{seg(i)} | \mathbf{x}^{(i)}) \\ & + \beta \log P(\mathbf{y}^{typ(i)} | \mathbf{x}^{(i)}), \end{aligned} \quad (3.1)$$

where  $N$  is the number of examples in the training set,  $\mathbf{y}^{seg}$  and  $\mathbf{y}^{typ}$  are the decomposed segmentation and type tags corresponding to the two sub-task modules, and  $\alpha$  and  $\beta$  are the hyper-parameters controlling the importance of the two modules' contributions respectively.

If the training example is fully labeled with both segmentation and type annotated, training is straightforward; if the training example is partially labeled, e.g., only with segmentation but without type, we can set the log probability of the type module and the decision module to 0 and only train the segmentation module. This formulation provides extra flexibility of using partially annotated corpus together with fully annotated to improve the overall performance.

### 3.6 Experimental Evaluation

Our experimental evaluation is designed to evaluate the two key aspects of our model:

Q1: Can the modular architecture alleviate the difficulty of learning the final task? To answer this question, we compare our modular architecture to the traditional neural-CRF model and several recent competitive models for sequence labeling combining inference and deep learning. The results are summarized in Tables 3.1-3.3.

Q2: Can partial labels be used effectively as a new form of weak-supervision? To answer this question we compared the performance of the model when trained using disjoint sets of partial and full labels. The results are summarized in Figures 3.3-3.5.

### 3.6.1 Experimental Settings

#### Datasets

We evaluated our models over three different sentiment tagging tasks adapted for sequence prediction. We also included additional results for multilingual NER in the Appendix for reference.

**Targeted Sentiment Datasets** We evaluated our models on the targeted sentiment datasets released by Mitchell et al. [2013], which consists of entity and sentiment annotations on both English and Spanish tweets. Similar to previous studies [Mitchell et al., 2013, Zhang et al., 2015, Li and Lu, 2017], our task focuses on people and organizations (collapsed into volitional named entities tags) and the sentiment associated with their description in tweets. After this processing, the labels of each tweets are composed of both segmentation (entity spans) and types (sentiment tags).

We used the original 10-fold cross validation splits to calculate averaged F1 score, using 10% of the training set for development. We used the same metrics in Zhang et al. [2015]’s and Li and Lu [2017]’s studies for a fair comparison.

**Aspect Based Sentiment Analysis Datasets** We used the Restaurants dataset provided by SemEval 2016 Task 5 subtask 1, consisting of opinion target (aspect) expression segmentation, aspect classification and matching sentiment prediction. In the original task definition, the three tasks were designed as a pipeline, and assumed gold aspect labels when predicting the matching sentiment labels. Instead, our model deals with the challenging end-to-end setting by casting the problem as a sequence labeling task, labeling each aspect segment with the aspect label and sentiment polarity<sup>†</sup>.

**Subjective Polarity Disambiguation Datasets** We adapted the SemEval 2013 Task 2 subtask A as another task to evaluate our model. In this task, the system is given a marked phrase inside a longer text, and is asked to label its polarity. Unlike the

---

<sup>†</sup>using only the subset of the data containing sequence information

original task, we did not assume the sequence is known, resulting in two decisions, identifying subjective expressions (i.e., a segmentation task) and labeling their polarity, which can be modeled jointly as a sequence labeling task.

### Input Representation and Model Architecture

Following previous studies [Ma and Hovy, 2016, Liu et al., 2018] showing that the word embedding choice can significantly influence performance, we used the pre-trained GloVe 100 dimension Twitter embeddings only for all tasks in the main text. All the words not contained in these embeddings (OOV, out-of-vocabulary words) are treated as an “unknown” word. Our models were deployed with minimal hyper parameters tuning, and can be briefly summarized as: the character embeddings has dimension 30, the hidden layer dimension of the character level LSTM is 25, and the hidden layer of the word level LSTM has dimension 300. Similar to Liu et al. [2018], we also applied highway networks [Srivastava et al., 2015] from the character level LSTM to the word level LSTM. In our pilot study, we shrank the number of parameters in our modular architectures to around one third such that the total number of parameter is similar as that in the LSTM-CRF model, but we did not observe a significant performance change so we kept them as denoted.

### Learning

We used BIOES tagging scheme but only during the training and convert them back to BIO2 for evaluation for all tasks<sup>‡</sup>. Our model was implemented using pytorch [Paszke et al., 2019]. To help improve performance we parallelized the forward algorithm and the Viterbi algorithm on the GPU. All the experiments were run on NVIDIA GPUs. We used the Stochastic Gradient Descent (SGD) optimization of

---

<sup>‡</sup>Using BIOES improves model complexity in Training, as suggested in previous studies. But to make a fair comparison to most previous work, who used BIO2 for evaluation, we converted labels to BIO2 system in the testing stage. (To be clear, using BIOES in the testing actually yields higher f1 scores in the testing stage, which some previous studies used unfairly)

batch size 10, with a momentum 0.9 to update the model parameters, with the learning rate 0.01, the decay rate 0.05; The learning rate decays over epochs by  $\eta/(1+e*\rho)$ , where  $\eta$  is the learning rate,  $e$  is the epoch number, and  $\rho$  is the decay rate. We used gradient clip to force the absolute value of the gradient to be less than 5.0. We used early-stop to prevent over-fitting, with a patience of 30 and at least 120 epochs. In addition to dropout, we used Adversarial Training (AT) [Goodfellow et al., 2014], to regularize our model as the parameter numbers increase with modules. AT improves robustness to small worst-case perturbations by computing the gradients of a loss function w.r.t. the input. In this study,  $\alpha$  and  $\beta$  in Eq. 3.1 are both set to 1.0, and we leave other tuning choices for future investigation. The source code and experimental setup are available online<sup>§</sup>.

### 3.6.2 Q1: Monolithic vs. Modular Learning

Our first set of results are designed to compare our modular learning models, utilize partial labels decomposition, with traditional monolithic models, that learn directly over the full labels. In all three tasks, we compare with strong sequence prediction models, including LSTM-CRF [Lample et al., 2016], which is directly equivalent to our baseline model (i.e., final task decision without the modules), and LSTM-CNN-CRF [Ma and Hovy, 2016] and LSTM-CRF-LM [Liu et al., 2018] which use a richer latent representation for scoring the emission potentials.

**Targeted Sentiment task** The results are summarized in Tab. 3.1. We also compared our models with recently published state-of-the-art models on these datasets. To help ensure a fair comparison with Ma et al. [2018a] which does not use inference, we also included the results of our model without the CRF layer (denoted LSTM-Ti(g)). All of our models beat the state-of-the-art results by a large margin.

---

<sup>§</sup>[https://github.com/cosmozhang/Modular\\_Neural\\_CRF](https://github.com/cosmozhang/Modular_Neural_CRF)

Table 3.1.: This table compares our models with the competing models on the targeted sentiment task. The results are on the full prediction of both segmentation and sentiment.

System	Architecture	Eng.	Spa.
Zhang et al. [2015]	Pipeline	40.06	43.04
	Joint	39.67	43.02
	Collapsed	38.36	40.00
Li and Lu [2017]	SS	40.11	42.75
	+embeddings	43.55	44.13
	+POS tags	42.21	42.89
	+semiMarkov	40.94	42.14
Ma et al. [2018a]	HMBi-GRU	42.87	45.61
baseline	LSTM-CRF	49.89	48.84
This work	LSTM-Ti(g)	45.84	46.59
	LSTM-CRF-T	51.34	49.47
	LSTM-CRF-Ti	51.64	49.74
	LSTM-CRF-Ti(g)	52.15	50.50

**Aspect Based Sentiment** We evaluated our models on two tasks: The first uses two modules, for identifying the position of the aspect in the text (i.e., chunking) and the aspect category prediction (denoted E+A). The second adds a third module that predicts the sentiment polarity associated with the aspect (denoted E+A+S). I.e., for a given sentence, label its entity span, the aspect category of the entity and the sentiment polarity of the entity at the same time <sup>¶</sup>. The results over four languages are summarized in Tab. 3.2. In all cases, our modular approach outperforms all monolithic approaches.

<sup>¶</sup>We adjusted our model for the E+A+S task.

Table 3.2.: This table compares our models with recent results on the Aspect Sentiment datasets.

Models	English		Spanish		Dutch		Russian	
	E+A	E+A+S	E+A	E+A+S	E+A	E+A+S	E+A	E+A+S
LSTM-CNN-CRF	58.73	44.20	64.32	50.34	51.62	36.88	58.88	38.13
LSTM-CRF-LM	62.27	45.04	63.63	50.15	51.78	34.77	62.18	38.80
LSTM-CRF	59.11	48.67	62.98	52.10	51.35	37.30	63.41	42.47
LSTM-CRF-T	60.87	49.59	64.24	52.33	52.79	37.61	64.72	43.01
LSTM-CRF-TI	63.11	50.19	64.40	52.85	53.05	38.07	64.98	44.03
LSTM-CRF-TI(g)	64.74	51.24	66.13	53.47	53.63	38.65	65.64	45.65

Subjective Phrase Identification and Classification In this task, tweets are annotated with sentiment phrases. As in the original SemEval task, it is tested in two settings, in-domain, where the test data also consists of tweets, and out-of-domain, where the test set consists of SMS text messages. We present the experimental results on these datasets in Table 3.3.

Table 3.3.: This table compares our models with competing models on the subjective sentiment task.

Models	Tweets	SMS
LSTM-CNN-CRF	35.82	23.23
LSTM-CRF-LM	35.67	23.25
LSTM-CRF	34.15	26.28
LSTM-CRF-T	35.37	27.11
LSTM-CRF-Ti	36.52	28.05
LSTM-CRF-Ti(g)	37.71	29.24

### 3.6.3 Q2: Partial Labels as Weak Supervision

Our modular architecture is a natural fit for learning with partial labels. Since the modular architecture decomposes the final task into sub-tasks, the absence of certain partial labels is permitted. Under this scenario, only the module corresponding to the available partial labels will be trained while the other parts of the model remain fixed.

This property can be exploited to reduce the supervision effort by defining semi-supervised learning protocols that use partial-labels when the full labels are not available, or too costly to obtain. E.g., in the target sentiment task, segmentation labels are significantly easier to annotate.

To demonstrate this property we conducted two sets of experiments. The first investigates how the decision module can effectively integrate the knowledge independently learned by sub-tasks modules using different partial labels. We quantify this ability by providing varying amounts of full labels to support the integration process. The second studies the traditional semi-supervised setting, where we only have a handful of full labels, but we have a larger amount of partial labels.

**Modular Knowledge Integration** The modular architecture allows us to train each model using data obtained separately for each task, and only use a handful of examples annotated for the final task in order to integrate the knowledge learned by each module into a unified decision. We simulated these settings by dividing the training data into three folds. We associated each one of the first two folds with the two sub-task modules. Each one of these folds only included the partial labels relevant for that sub-task. We then used gradually increasing amounts of the third fold, consisting of the full labels, for training the decision module.

Fig. 3.3 describes the outcome for targeted sentiment task, comparing a non-modular model using only the full labels, with the modular approach, which uses the full labels for knowledge integration. Results show that even when very little full data is available the modular approach can result in significantly improve. Additional

results show the same pattern for subjective phrase identification and classification are included in the Appendix.

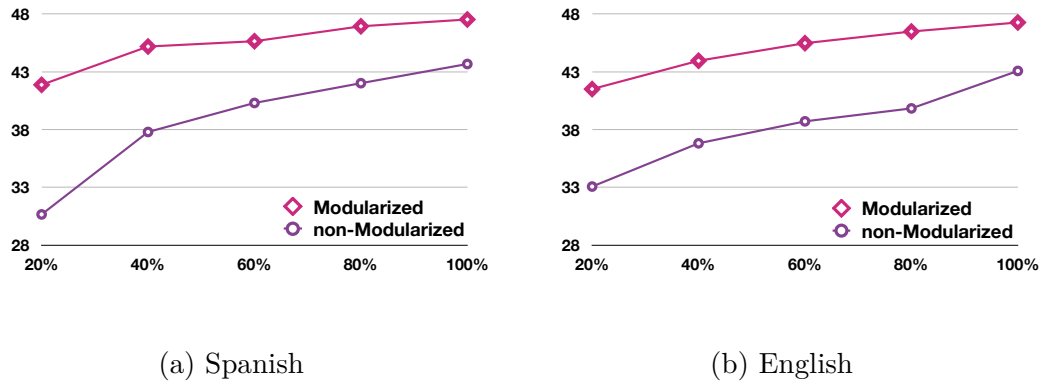


Figure 3.3.: This figure shows modular knowledge integration experimental results on the Targeted Sentiment Datasets. The x-axis is the amount of percentage of the third fold of full labels. The “non-modularized” means we only provide fully labeled data from the third fold.

**Learning with Partially Labeled Data** Partially-labeled data can be cheaper and easier to obtain, especially for low-resource languages. In this set of experiments, we model these settings over the target-sentiment task. The results are summarized in Fig. 3.4. We fixed the amount of full labels to 20% of the training set, and gradually increased the amount of partially labeled data. We studied adding segmentation and type separately. After the model is trained in this routine, it was tested on predicting the full labels jointly on the test set.

**Domain Transfer with Partially Labeled Data** In our final analysis we considered a novel domain-adaptation setup, where we have a small amount of fully labeled in-domain data from aspect sentiment and more out-of-domain data from target senti-



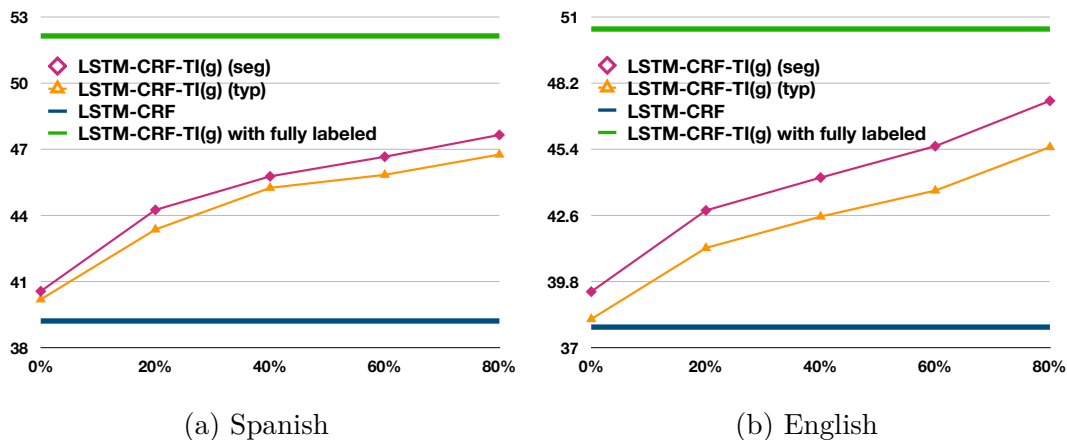


Figure 3.4.: The fully labeled data were fixed to 20% of the whole training set, and data with only segmentation information (Magenta), or with only type information (Orange) were gradually added. Then the model was tested on the full prediction test. The LSTM-CRF model can only use fully labeled data as it does not decompose the task.

ment. However unlike the traditional domain-adaptation settings, the out-of-domain data is labeled for a different task, and only shares one module with the original task.

In our experiments we fixed 20% of the fully labeled data for the aspect sentiment task, and gradually added out-of-domain data, consisting of partial sentiment labels from the target sentiment task. Our model successfully utilized the out-of-domain data to improve the performance on the in-domain task. The results are shown on Fig 3.5.

### 3.7 Conclusion

We present and study several modular neural architectures designed for a novel learning scenario: learning from partial labels. We experiment with several sentiment tagging tasks. Our models, inspired by cognitive neuroscience findings and multi-task learning, suggest a functional decomposition of the original task into simpler

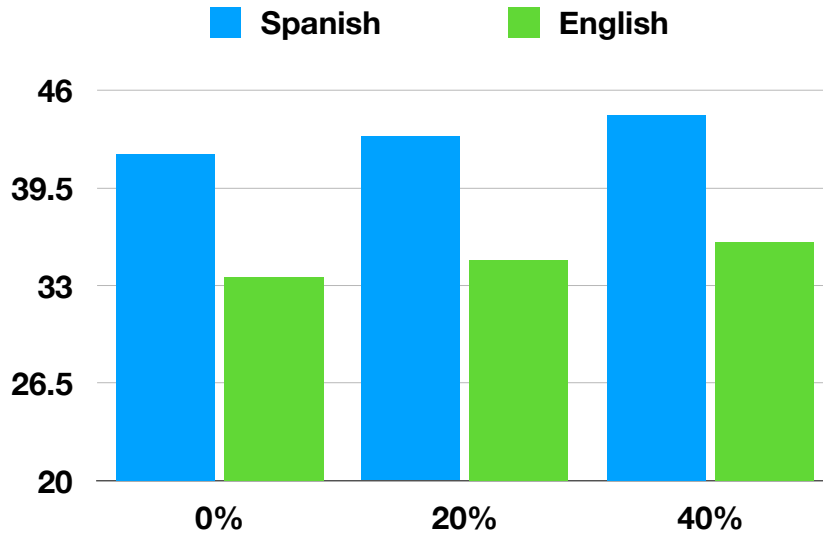


Figure 3.5.: This figure shows domain transfer experimental results with fixed 20% in-domain data from aspect sentiment and various amounts of out-of-domain data from target sentiment, shown on the x-axis.

sub-tasks. We evaluated different methods for sharing information and integrating the modules into the final decision, such that a better model can be learned, while converging faster<sup>‡</sup>. As our experiments show, modular learning can be used with weak supervision, using examples annotated with partial labels only.

The modular approach also provides interesting directions for future research, focusing on alleviating the supervision bottleneck by using large amount of partially labeled data that are cheaper and easy to obtain, together with only a handful amount of annotated data, a scenario especially suitable for low-resource languages.

---

<sup>‡</sup>Results of convergence analysis are provided in the Appendix.

#### 4 NEURAL CONDITIONAL RANDOM FIELDS AUTOENCODER

In the real world, most of the data are not annotated at all. Though partially labeled data is cheaper and easier to obtain than fully labeled data, in a lot scenarios even partially labeled data are scarce, especially when dealing with low-resource languages. The best hope is that we can directly make use of those unannotated data in the learning process. From the perspective of statistical learning, treating those missing labels of the unannotated data as latent variables sheds a light on mitigating this difficulty.

In this chapter, we introduce a novel model—neural CRF autoencoder (NCRFAE)—for semi-supervised sequence labeling, which consists of a discriminative component built upon a CRF model enhanced by DNN and a generative component aiming to reconstruct the input from the prediction. The discriminative component and the generative component complement and constrain each other rather than merely increase the model complexity. The model has a unified structure with shared parameters while using different loss functions for labeled and unlabeled data. Along with the NCRFAE model, we propose a tractable algorithm that extends the Baum-Welsh algorithm [Elworthy, 1994] which is regarded as a sequential version of EM algorithm. This algorithm deals with the missing labels as latent variables and provides an efficient way to calculate their posteriors under a Maximum a Posterior (MAP) framework. By decoupling the parameters in the discriminative encoder and the generative decoder, the model is optimized to escape bad local minimums.

We ran extensive experiments on different languages and found empirically that our model can exploit the hidden information inside the unlabeled sentences, to improve the overall accuracy on using labeled data alone in POS tagging task, consistently across different languages. Additionally, under the supervised setup, our model outperforms existing methods.

## 4.1 Introduction

The recent renaissance of deep learning has led to significant strides forward in several AI fields. In Natural Language Processing (NLP), characterized by highly structured tasks, promising results have been obtained by models that combine deep learning methods with traditional structured learning algorithms [Chen and Manning, 2014, Durrett and Klein, 2015, Andor et al., 2016, Wiseman and Rush, 2016]. These models combine the strengths of neural models, that can score local decisions using a rich non-linear representation, with efficient inference procedures used to combine the local decisions into a coherent global decision. Among these models, neural variants of the Conditional Random Fields (CRF) model [Lafferty et al., 2001] are especially popular. By replacing the linear potentials with non-linear potential generated by using neural networks these models are able to improve performance on several structured prediction tasks [Andor et al., 2016, Peng and Dredze, 2016, Lample et al., 2016, Ma and Hovy, 2016, Durrett and Klein, 2015].

Despite their promise, wider adoption of these algorithms for new structured prediction tasks can be difficult. Neural networks are notoriously susceptible to overfitting unless large amounts of training data are available. This problem is exacerbated in the structured settings, as accounting for the dependencies between decisions requires even more data. Providing it through manual annotation is often a difficult and labor-intensive task.

In this study we tackle this problem, by proposing an end-to-end neural CRF autoencoder (NCRFAE) model for semi-supervised learning on sequence labeling problems.

An autoencoder is a special type of neural networks, modeling the conditional probability  $P(\hat{X}|X)$ , where  $X$  is the original input to the model and  $\hat{X}$  is the reconstructed input [Hinton and Zemel, 1994]. Autoencoders consist of two parts, an encoder projecting the input to a hidden space, and a decoder reconstructing the input from it.

Traditionally, autoencoders are used for generating a compressed representation of the input by projecting it into a dense low dimensional space. In our setting the hidden space consists of discrete variables that comprise the output structure. These generalized settings are described in Figure 4.1a. By definition, it is easy to see that the encoder (lower half in Figure 4.1a) can be modeled by a discriminative model describing  $P(Y|X)$  directly, while the decoder (upper half in Figure 4.1a) naturally fits as a generative model, describing  $P(\hat{X}|Y)$ , where  $Y$  is the label. In our model, illustrated in Figure 4.1b, the encoder is a CRF model with neural networks as its potential generators, while the decoder is a generative model, trying to reconstruct the input.

Our model carries the merit of autoencoders, which can exploit valuable information from unlabeled data. Recent works [Ammar et al., 2014, Lin et al., 2015] suggested using an autoencoder with a CRF model as an encoder in an unsupervised setting. We significantly expand on these works and suggest the following contributions:

1. We propose a unified model seamlessly accommodating both unlabeled and labeled data. While past work focused on unsupervised structured prediction, neglecting the discriminative power of such models, our model easily supports learning in both fully supervised and semi-supervised settings. Accompanying with the model, we additionally developed a variation of the Expectation-Maximization (EM) algorithm, used for optimizing the encoder and the decoder of our model simultaneously.
2. We increase the expressivity of the traditional CRF autoencoder model using neural networks as the potential extractors, thus avoiding the heavy feature engineering necessary in previous works.
3. We demonstrate the advantages of our model empirically, by conducting experiments on the well-known Part of Speech (POS) tagging problem over 8 different languages, including low-resource languages. In the supervised setting, our NCRFAE outperformed the highly optimized NCRF. In the semi-supervised setting, our model was able to successfully utilize unlabeled data, improving on the performance using

only the labeled data alone, and outperforming competing semi-supervised learning algorithms.

Furthermore, our newly proposed algorithm is directly applicable to other sequential learning tasks in NLP, and can be easily adapted to other structured tasks such as dependency parsing or constituent parsing by replacing the forward-backward algorithm with the inside-outside algorithm. All of these tasks can benefit from semi-supervised learning algorithms.\*

## 4.2 Related Work

Neural networks have been successfully applied to many NLP tasks, including tagging [Ma and Hovy, 2016, Mesnil et al., 2015, Lample et al., 2016], parsing [Chen and Manning, 2014], text generation [Sutskever et al., 2011], machine translation [Bahdanau et al., 2015], sentiment analysis [Kim, 2014] and question answering [Andreas et al., 2016]. Most relevant to this work are structured prediction models capturing dependencies between decisions, either by modeling the dependencies between the hidden representations of connected decisions using RNN or LSTM [Vaswani et al., 2016, Katiyar and Cardie, 2016], by explicitly modeling the structural dependencies between output predictions [Durrett and Klein, 2015, Lample et al., 2016, Andor et al., 2016], or by combining the two approaches [Socher et al., 2013, Wiseman and Rush, 2016].

In contrast to supervised latent variable models, such as the Hidden Conditional Random Fields in [Quattoni et al., 2007], which utilize additional latent variables to infer for supervised structure prediction, we do not presume any additional latent variables in our NCRFAE model in both supervised and semi-supervised setting.

The difficulty of providing sufficient supervision has motivated work on semi-supervised and unsupervised learning for many of these tasks [McClosky et al., 2006, Spitkovsky et al., 2010a, Subramanya et al., 2010, Stratos and Collins, 2015a, Marinho

---

\*Our code and experimental set up will be available at <https://github.com/cosmozhang/NCRFAE>

et al., 2016, Tran et al., 2016], including several that also used autoencoders [Ammar et al., 2014, Lin et al., 2015, Miao and Blunsom, 2016, Kociský et al., 2016, Cheng et al., 2017]. In this paper we expand on these works, and suggest a neural CRF autoencoder, that can leverage both labeled and unlabeled data.

### 4.3 Neural CRF Autoencoder

In semi-supervised learning the model needs to utilize both labeled and unlabeled data. Autoencoders offer a convenient way to deal with both types of data in a unified fashion.

A generalized autoencoder (Figure 4.1a) tries to reconstruct the input  $\hat{X}$  given the original input  $X$ , aiming to maximize the complete log probability  $P(\hat{X}|X)$  without knowing the latent variable  $Y$  explicitly. Since we focus on sequential structured prediction problems, the encoding and decoding processes are no longer for a single data point  $(x, y)$  ( $x$  if unlabeled), but for the whole input instance and output sequence  $(\mathbf{x}, \mathbf{y})$  ( $\mathbf{x}$  if unlabeled). Additionally, as our main purpose in this study is to reconstruct the input with precision,  $\hat{\mathbf{x}}$  is just a copy of  $\mathbf{x}$ .



(a) A generalized autoencoder.

(b) The NCRFAE model in this work.

Figure 4.1.: On the left is a generalized autoencoder, of which the lower half is the encoder and the upper half is the decoder. On the right is an illustration of the graphical model of our NCRFAE model. The yellow squares are interactive (transition) potentials among labels, and the green squares represent the unary (emission) potentials generated by the neural networks.

As shown in Figure 4.1b, our NCRFAE model consists of two parts: the encoder (the lower half) that is a discriminative CRF model enhanced by deep neural networks as its potential generator with encoding parameters  $\Lambda$ , describing the probability of a predicted sequence of labels given the input; the decoder (the upper half) that is a generative model with reconstruction parameters  $\Theta$ , modeling the probability of reconstructing the input given a sequence of labels. We present our model formally as follows:

$$\begin{aligned} P_{\Theta, \Lambda}(\hat{\mathbf{x}}|\mathbf{x}) &= \sum_{\mathbf{y}} P_{\Theta, \Lambda}(\hat{\mathbf{x}}, \mathbf{y}|\mathbf{x}) \\ &= \sum_{\mathbf{y}} P_{\Theta}(\hat{\mathbf{x}}|\mathbf{y})P_{\Lambda}(\mathbf{y}|\mathbf{x}), \end{aligned}$$

where  $P_{\Lambda}(\mathbf{y}|\mathbf{x})$  is the probability given by the neural CRF encoder, and  $P_{\Theta}(\hat{\mathbf{x}}|\mathbf{y})$  is the probability produced by the generative decoder.

When making a prediction, the model tries to find the most probable output sequence by performing the following inference procedure using the Viterbi algorithm:

$$\mathbf{y}^* = \arg \max_{\mathbf{y}} P_{\Theta, \Lambda}(\hat{\mathbf{x}}, \mathbf{y}|\mathbf{x}).$$

To clarify, as we focus on POS tagging problems in this study, in the unsupervised setting where the true POS tags are unknown, the labels used for reconstruction are actually the POS tags being induced. The labels induced here are corresponding to the hidden nodes in a generalized autoencoder model.

#### 4.3.1 Neural CRF Encoder

In a CRF model, the probability of predicted labels  $\mathbf{y}$ , given sequence  $\mathbf{x}$  as input is modeled as

$$P_{\Lambda}(\mathbf{y}|\mathbf{x}) = \frac{e^{\Phi(\mathbf{x}, \mathbf{y})}}{Z},$$



where  $Z = \sum_{\tilde{\mathbf{y}}} e^{\Phi(\mathbf{x}, \tilde{\mathbf{y}})}$  is the partition function that marginalize over all possible assignments to the predicted labels of the sequence, and  $\Phi(\mathbf{x}, \mathbf{y})$  is the scoring function, which is defined as:

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_t \phi(\mathbf{x}, y_t) + \psi(y_{t-1}, y_t).$$

The partition function  $Z$  can be computed efficiently via the forward-backward algorithm. The term  $\phi(\mathbf{x}, y_t)$  corresponds to the score of a particular tag  $y_t$  at position  $t$  in the sequence, and  $\psi(y_{t-1}, y_t)$  represents the score of transition from the tag at position  $t - 1$  to the tag at position  $t$ . In our NCRFAE model,  $\phi(\mathbf{x}, y_t)$  is described by deep neural networks while  $\psi(y_{t-1}, y_t)$  by a transition matrix. Such a structure allows for the use of distributed representations of the input, for instance, the word embeddings on a continuous vector space [Mikolov et al., 2013].

Typically in our work,  $\phi(\mathbf{x}, y_t)$  is modeled jointly by a multi-layer perceptron (MLP) that utilizes the word-level information, and a bi-directional long-short term memory (LSTM) neural network [Hochreiter and Uergen Schmidhuber, 1997] that captures the character level information within each word. A bi-directional structure can extract character level information from both directions, with which we expect to catch the prefix and suffix information of words in an end-to-end system, rather than using hand-engineered features. The bi-directional LSTM (BiLSTM) neural network takes as input the character embeddings  $\mathbf{e}_c \in \mathbb{R}^{k_1}$ , where  $k_1$  is the dimensionality of the character embeddings. A normal LSTM can be denoted as:

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_{ei}\mathbf{e}_{c_t} + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i), \\ \mathbf{f}_t &= \sigma(\mathbf{W}_{ef}\mathbf{e}_{c_t} + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f), \\ \mathbf{o}_t &= \sigma(\mathbf{W}_{eo}\mathbf{e}_{c_t} + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o), \\ \mathbf{g}_t &= \text{Relu}(\mathbf{W}_{ec}\mathbf{e}_{c_t} + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c), \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t, \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \end{aligned}$$

where  $\odot$  denotes element-wise multiplication. Then a bi-directional LSTM neural network extends it as follows, by denoting the procedure of generating  $\mathbf{h}_t$  as  $\mathcal{H}$ :

$$\begin{aligned}\vec{\mathbf{h}}_t &= \mathcal{H}(\mathbf{W}_{e\vec{h}}\mathbf{e}_{c_t} + \mathbf{W}_{\vec{h}\vec{h}}\vec{\mathbf{h}}_{t-1} + \mathbf{b}_{\vec{h}}), \\ \overleftarrow{\mathbf{h}}_t &= \mathcal{H}(\mathbf{W}_{e\overleftarrow{h}}\mathbf{e}_{c_t} + \mathbf{W}_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{\mathbf{h}}_{t-1} + \mathbf{b}_{\overleftarrow{h}}),\end{aligned}$$

where  $\mathbf{e}_{c_t}$  here is the character embedding for character  $c$  in position  $t$  in a word.

The inputs to the MLP are the word embeddings  $\mathbf{e}_v \in \mathbb{R}^{k_2}$  for each word  $v$ , where  $k_2$  is the dimensionality of the vector, concatenated with the final representation generated by the Bi-LSTM over the characters of that word:  $\mathbf{u} = [\mathbf{e}_v; \vec{\mathbf{h}}_v; \overleftarrow{\mathbf{h}}_v]$ . In order to leverage the capacity of the CRF model, we use a word and its context together to generate the unary potential. More specifically, we adopt a concatenation  $\mathbf{v}_t = [\mathbf{u}_{t-(w-1)/2}; \dots; \mathbf{u}_{t-1}; \mathbf{u}_t; \mathbf{u}_{t+1}; \dots; \mathbf{u}_{t+(w-1)/2}]$  as the inputs to the MLP model, where  $t$  denotes the position in a sequence, and  $w$  being an odd number indicates the context size. Further, in order to enhance the generality of our model, we add a dropout layer on the input right before the MLP layer as a regularizer. Notice that different from a normal MLP, the activation function of the last layer is no more a softmax function, but a linear function generating the log-linear part  $\phi_t(\mathbf{x}, y_t)$  of the CRF model:

$$\begin{aligned}\mathbf{h}_t &= \text{Relu}(\mathbf{W}\mathbf{v}_t + \mathbf{b}) \\ \phi_t &= \mathbf{w}_y^\top \mathbf{h}_t + b_y.\end{aligned}$$

The transition score  $\psi(y_{t-1}, y_t)$  is a single scalar representing the interactive potential. We use a transition matrix  $\Psi$  to cover all the transitions between different labels, and  $\Psi$  is part of the encoder parameters  $\Lambda$ .

All the parameters in the neuralized encoder are updated when the loss function is minimized via error back-propagation through all the structures of the neural networks and the transition matrix.

The detailed structure of the neural CRF encoder is demonstrated in Fig 4.2. Note that the MLP layer is also interchangeable with a RNN layer or LSTM layer.

But in our pilot experiments, we found a single MLP structure yields better performance, which we conjecture is due to over-fitting caused by the high complexity of the alternatives.

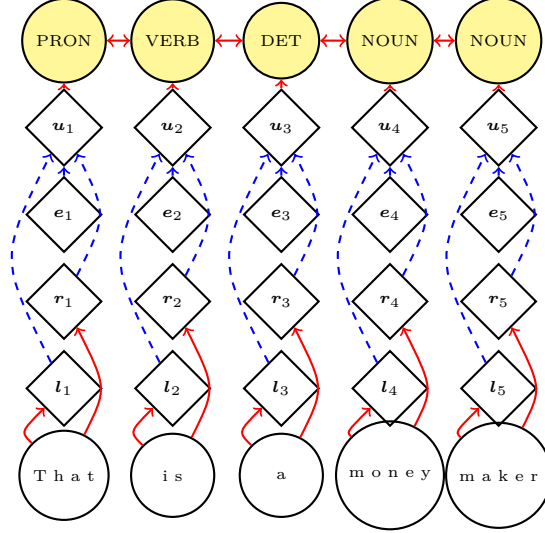


Figure 4.2.: A demonstration of the neural CRF encoder.  $\mathbf{l}_t$  and  $\mathbf{r}_t$  are the output of the forward and backward character-level LSTM of the word at position  $t$  in a sentence, and  $\mathbf{e}_t$  is the word-level embedding of that word.  $\mathbf{u}_t$  is the concatenation of  $\mathbf{e}_t$ ,  $\mathbf{l}_t$  and  $\mathbf{r}_t$ , denoted by blue dashed arrows.

### 4.3.2 Generative Decoder

In our NCRFAE model, we assume the generative process follows several multinomial distributions: each label  $y$  has the probability  $\theta_{y \rightarrow x}$  to reconstruct the corresponding word  $x$ , i.e.,  $P(x|y) = \theta_{y \rightarrow x}$ . This setting naturally leads to a constraint  $\sum_x \theta_{y \rightarrow x} = 1$ . The number of parameters of the decoder is  $|\mathcal{Y}| \times |\mathcal{X}|$ . For a whole sequence, the reconstruction probability is  $P_{\Theta}(\hat{\mathbf{x}}|\mathbf{y}) = \prod_t P(\hat{x}_t|y_t)$ .

#### 4.4 A Unified Learning Framework

We first constructed two loss functions for labeled and unlabeled data using the same model. Our model is trained in an on-line fashion: given a labeled or unlabeled sentence, our NCRFAE optimizes the loss function by choosing the corresponding one. In an analogy to coordinate descent, we optimize the loss function of the NCRFAE by alternatively updating the parameters  $\Theta$  in the decoder and the parameters  $\Lambda$  in the encoder. The parameters  $\Theta$  in the decoder are updated via a variation of the Expectation-Maximization (EM) algorithm, and the the parameters  $\Lambda$  in the encoder are updated through a gradient-based method due to the non-convexity of the neuralized CRF. In contrast to the early autoencoder models [Ammar et al., 2014, Lin et al., 2015], our model has two distinctions: First, we have two loss functions to model labeled example and unlabeled examples; Second, we designed a variant of EM algorithm to alternatively learn the parameters of the encoder and the decoder at the same time.

##### 4.4.1 Unified Loss Functions for Labeled and unlabeled Data

For a sequential input with labels, the complete data likelihood given by our NCRFAE is

$$\begin{aligned} P_{\Theta, \Lambda}(\hat{\mathbf{x}}, \mathbf{y} | \mathbf{x}) &= P_{\Theta}(\hat{\mathbf{x}} | \mathbf{y}) P_{\Lambda}(\mathbf{y} | \mathbf{x}) \\ &= \left[ \prod_t P(\hat{x}_t | y_t) \right] \frac{e^{\Phi(\mathbf{x}, \mathbf{y})}}{Z} \\ &= \frac{e^{\sum_t s_t(\mathbf{x}, \mathbf{y})}}{Z}, \end{aligned}$$

where

$$s_t(\mathbf{x}, \mathbf{y}) = \log P(x_t | y_t) + \phi(\mathbf{x}, y_t) + \psi(y_{t-1}, y_t).$$

If the input sequence is unlabeled, we can simply marginalize over all the possible assignment to labels. The probability is formulated as

$$\begin{aligned} P_{\Theta, \Lambda}(\hat{\mathbf{x}}|\mathbf{x}) &= \sum_{\mathbf{y}} P(\hat{\mathbf{x}}, \mathbf{y}|\mathbf{x}) \\ &= \frac{U}{Z}, \end{aligned}$$

where  $U = \sum_{\mathbf{y}} e^{\sum_t s_t(\mathbf{x}, \mathbf{y})}$ .

Our formulation have two advantages. First, term  $U$  is different from but in a similar form as term  $Z$ , such that to calculate the probability  $P(\hat{\mathbf{x}}|\mathbf{x})$  for an unlabeled sequence, the forward-backward algorithm to compute the partition function  $Z$  can also be applied to compute  $U$  efficiently. Second, our NCRFAE highlights a unified structure of different loss functions for labeled and unlabeled data with shared parameters. Thus during training, our model can address both labeled and unlabeled data well by alternating the loss functions. Using negative log-likelihood as our loss function, if the data is labeled, the loss function is:

$$\begin{aligned} loss_l &= -\log P_{\Theta, \Lambda}(\hat{\mathbf{x}}, \mathbf{y}|\mathbf{x}) \\ &= -\left(\sum_t s_t(\mathbf{x}, \mathbf{y}) - \log Z\right) \end{aligned}$$

If the data is unlabeled, the loss function is:

$$\begin{aligned} loss_u &= -\log P_{\Theta, \Lambda}(\hat{\mathbf{x}}|\mathbf{x}) \\ &= -(\log U - \log Z). \end{aligned}$$

Therefore, during training, based on whether the encountered data is labeled or unlabeled, our model can select the appropriate loss function for learning parameters. In practice, we found for labeled data, using a combination of  $loss_l$  and  $loss_u$  actually yields better performance.

#### 4.5 Mixed Expectation-Maximization Algorithm

The Expectation-Maximization (EM) algorithm [Dempster et al., 1977] was applied to a wide range of problems. Generally, it establishes a lower-bound of the

objective function by using Jensen’s Inequality. It first tries to find the posterior distribution of the latent variables, and then based on the posterior distribution of the latent variables, it maximizes the lower-bound. By alternating expectation (E) and maximization (M) steps, the algorithm iteratively improves the lower-bound of the objective function.

In this section we describe the mixed Expectation-Maximization (EM) algorithm used in our study. Parameterized by the encoding parameters  $\mathbf{\Lambda}$  and the reconstruction parameters  $\mathbf{\Theta}$ , our NCRFAE consists of the encoder and the decoder, which together forms the log-likelihood a highly non-convex function. However, a careful observation shows that if we fix the encoder, the lower bound derived in the E step, is convex with respect to the reconstruction parameters  $\mathbf{\Theta}$  in the M step. Hence, in the M step we can analytically obtain the global optimum of  $\mathbf{\Theta}$ . With respect to the reconstruction parameters  $\mathbf{\Theta}$  by fixing  $\mathbf{\Lambda}$ , we describe our EM algorithm in iteration  $t$  as follows:

In the E-step, we let  $Q(\mathbf{y}_i) = P(\mathbf{y}_i|\mathbf{x}_i, \hat{\mathbf{x}}_i)$ , and treat  $\mathbf{y}_i$  the latent variable as it is not observable in unlabeled data. We derive the lower-bound of the log-likelihood using  $Q(\mathbf{y}_i)$ :

$$\begin{aligned} \sum_i \log P(\hat{\mathbf{x}}_i|\mathbf{x}_i) &= \sum_i \log \sum_{\mathbf{y}_i} Q(\mathbf{y}_i) \frac{P(\hat{\mathbf{x}}_i, \mathbf{y}_i|\mathbf{x}_i)}{Q(\mathbf{y}_i)} \\ &\geq \sum_i \sum_{\mathbf{y}_i} Q(\mathbf{y}_i) \log \frac{P(\hat{\mathbf{x}}_i, \mathbf{y}_i|\mathbf{x}_i)}{Q(\mathbf{y}_i)}, \end{aligned}$$

where  $Q(\mathbf{y}_i)$  is computed using parameters  $\mathbf{\Theta}^{(t-1)}$  in the previous iteration  $t - 1$ .

In the M-step, we try to improve the aforementioned lower-bound using all examples:

$$\begin{aligned}
& \arg \max_{\Theta^{(t)}} \sum_i \sum_{\mathbf{y}_i} Q(\mathbf{y}_i) \log \frac{P_{\Theta^{(t)}}(\hat{\mathbf{x}}_i | \mathbf{y}_i) P_{\Lambda}(\mathbf{y}_i | \mathbf{x}_i)}{Q(\mathbf{y}_i)} \\
& \arg \max_{\Theta^{(t)}} \sum_i \sum_{\mathbf{y}_i} Q(\mathbf{y}_i) \log P_{\Theta^{(t)}}(\hat{\mathbf{x}}_i | \mathbf{y}_i) + \text{const} \\
& \arg \max_{\Theta^{(t)}} \sum_{y \rightarrow x} \log \theta_{y \rightarrow x}^{(t)} \sum_y Q(y) C(y, x) \\
& \arg \max_{\Theta^{(t)}} \sum_{y \rightarrow x} \log \theta_{y \rightarrow x}^{(t)} \mathbb{E}_{y \sim Q} [C(y, x)] \\
& \text{s.t.} \sum_x \theta_{y \rightarrow x}^{(t)} = 1.
\end{aligned}$$

In this formulation, *const* is a constant with respect to the parameters we are updating.  $Q(y)$  is the distribution of a label  $y$  at any position by marginalizing labels at all other positions in a sequence. By denoting  $C(y, x)$  as the number of times that  $(x, y)$  co-occurs,  $\mathbb{E}_{y \sim Q_{\Theta^{(t-1)}}} [C(y, x)]$  is the expected count of a particular reconstruction at any position, which can also be calculated using Baum-Welch algorithm [Welch, 2003], and can be summed over for all examples in the dataset (In the labeled data, it is just a real count). The algorithm we used to calculate the expected count is described in Algorithm 1. Therefore, it can be shown that the aforementioned global optimum can be calculated by simply normalizing the expected counts. In terms of the encoder's parameters  $\Lambda$ , they are first updated via a gradient-based optimization before each EM iteration. Based on the above discussion, our Mixed EM Algorithm is presented in Algorithm 2.

---

**Algorithm 1 Obtain Expected Count ( $T_e$ )**


---

Require: the expected count table  $T_e$

- 1: for an unlabeled data example  $\mathbf{x}_i$  do
  - 2:     Compute the forward messages:  $\alpha(y, t) \quad \forall y, t.$       $\triangleright t$  is the position in a sequence.
  - 3:     Compute the backward messages:  $\beta(y, t) \quad \forall y, t.$
  - 4:     Calculate the expected count for each  $x$  in  $\mathbf{x}_i$ :  $P(y_t|x_t) \propto \alpha(y, t) \times \beta(y, t).$
  - 5:      $T_e(x_t, y_t) \leftarrow T_e(x_t, y_t) + P(y_t|x_t)$       $\triangleright T_e$  is the expected count table.
  - 6: end for
- 

---

**Algorithm 2 Mixed Expectation-Maximization**


---

- 1: Initialize expected count table  $T_e$  using labeled data  $\{\mathbf{x}, \mathbf{y}\}_i^l$  and use it as  $\Theta^{(0)}$  in the decoder.
  - 2: Initialize  $\Lambda^{(0)}$  in the encoder randomly.
  - 3: for  $t$  in *epochs* do
  - 4:     Train the encoder on labeled data  $\{\mathbf{x}, \mathbf{y}\}^l$  and unlabeled data  $\{\mathbf{x}\}^u$  to update  $\Lambda^{(t-1)}$  to  $\Lambda^{(t)}$ .
  - 5:     Re-initialize expected count table  $T_e$  with  $\mathbf{0}$ s.
  - 6:     Use labeled data  $\{\mathbf{x}, \mathbf{y}\}^l$  to calculate real counts and update  $T_e$ .
  - 7:     Use unlabeled data  $\{\mathbf{x}\}^u$  to compute the expected counts with parameters  $\Lambda^{(t)}$  and  $\Theta^{(t-1)}$  and update  $T_e$ .
  - 8:     Obtain  $\Theta^{(t)}$  globally and analytically based on  $T_e$ .
  - 9: end for
- 

This mixed EM algorithm is a combination of the gradient-based approach to optimize the encoder by minimizing the negative log-likelihood as the loss function, and the EM approach to update the decoder's parameters by improving the lower-bound of the log-likelihood.



Table 4.1.: This table shows the statistics of different UD languages used in our experiments, including the number of tokens, and the number of sentences in training (train), development (dev) and testing (test) set respectively.

	English	French	German	Italian	Russian	Spanish	Indonesian	Croatian
Tokens	254830	391107	293088	272913	99389	423346	121923	139023
Train	12543	14554	14118	12837	4029	14187	4477	5792
Dev	2002	1596	799	489	502	1552	559	200
Test	2077	298	977	489	499	274	297	297

## 4.6 Experiments

### 4.6.1 Experimental Settings

**Dataset** We evaluated our model on the POS tagging task, in both the supervised and semi-supervised learning settings, over eight different languages from the UD (Universal Dependencies) 1.4 dataset [McDonald et al., 2013]. The task is defined over 17 different POS tags, used across different languages. We followed the original UD division for training, development and testing in our experiments. The statistics of the data used in our experiments are described in Table 4.1. The UD dataset includes several low-resource languages which are of particular interest to our semi-supervised model.

**Input Representation and Neural Architecture** Our model uses word embeddings as input. In our pilot experiments, we compared the performance on the English dataset of the pre-trained embedding from Google News [Mikolov et al., 2013] and the embeddings we trained directly on the UD dataset using the skip-gram algorithm [Mikolov et al., 2013]. We found these two types of embeddings yield very similar performance on the POS tagging task. So in our experiments, we used embeddings of different languages directly trained on the UD dataset as input to our model, whose

dimension is 200. In the MLP neural network layer, the number of hidden nodes in the hidden layer is 20, which is the same as the hidden layer in the character-level LSTM. The dimension of the character-level embeddings fed into the LSTM layer is 15, which is randomly initialized. In order to incorporate the global information of the input sequence, we set the context window size to 3. The dropout rate for the dropout layer is set to 0.5.

**Learning** We used ADADELTA [Zeiler, 2012] to learn parameters  $\Lambda$  in the encoder, as ADADELTA dynamically adapts learning rate over time using only first order information and has minimal computational overhead beyond vanilla stochastic gradient descent (SGD). The authors of ADADELTA also argue this method appears robust to noisy gradient information, different model architecture choices, various data modalities and selection of hyper-parameters. We observed that ADADELTA indeed had faster convergence than vanilla SGD optimization. In our experiments, we included word embeddings and character embeddings as parameters as well. We used Theano to implement our algorithm, and all the experiments were run on NVIDIA GPUs. To prevent over-fitting, we used the “early-stop” strategy to determine the appropriate number of epochs during training. We did not take efforts to tune those hyper-parameters and they remained the same in both our supervised and semi-supervised learning experiments.

#### 4.6.2 Supervised Learning

In these settings our Neural CRF autoencoder model had access to the full amount of annotated training data in the UD dataset. As described in Section 4.5, the decoder’s parameters  $\Theta$  were estimated using real counts from the labeled data.

We compared our model with existing sequence labeling models including HMM, CRF, LSTM and neural CRF (NCRF) on all the 8 languages. Among these models, the NCRF can be most directly compared to our model, as it is used as the base of

<b>Text</b>	Google	is	a	nice	search	engine	.
<b>Gold</b>	PROPN	VERB	DET	ADJ	NOUN	NOUN	PUNCT
<b>NCRF-AE</b>	PROPN	VERB	DET	ADJ	NOUN	NOUN	PUNCT
<b>NCRF</b>	NOUN	VERB	DET	ADJ	NOUN	NOUN	PUNCT
<b>LSTM</b>	PROPN	VERB	DET	ADJ	VERB	NOUN	PUNCT

Figure 4.3.: This figure shows an example from the test set to compare the predicted results of our NCRFAE model, the NCRF model and the LSTM model.

our model, but without the decoder (and as a result, can only be used for supervised learning).

The results, summarized in Table 4.2, show that our NCRFAE consistently outperformed all other systems, on all the 8 languages, including Russian, Indonesian and Croatian which had considerably less data compared to other languages. Interestingly, the NCRF consistently came second to our model, which demonstrates the efficacy of the expressivity added to our model by the decoder, together with an appropriate optimization approach.

Table 4.2.: This table shows supervised learning performance of POS tagging on 8 UD languages using different models

Models	English	French	German	Italian	Russian	Spanish	Indonesian	Croatian
HMM	86.28%	91.23%	85.59%	92.03%	79.82%	91.31%	89.40%	86.98%
CRF	89.96%	93.40%	86.83%	94.07%	83.38%	91.47%	88.63%	86.90%
LSTM	90.50%	94.16%	88.40%	94.96%	84.87%	93.17%	89.42%	88.95%
NCRF	91.52%	95.07%	90.27%	96.20%	93.37%	93.34%	92.32%	93.85%
NCRFAE	92.50%	95.28%	90.50%	96.64%	93.60%	93.86%	93.96%	94.32%

To better understand the performance difference by different models, we performed error analysis, using an illustrative example, described in Figure 4.3.

Table 4.3.: This table shows semi-supervised learning accuracy of POS tagging on 8 UD languages. HEM means hard-EM, used as a self-training approach, and OL means only 20% of the labeled data is used and no unlabeled data is used.

Models	English	French	German	Italian	Russian	Spanish	Indonesian	Croatian
NCRF (OL)	88.01%	93.38%	90.43%	91.75%	86.63%	91.22%	88.35%	86.11%
NCRFAE (OL)	88.41%	93.69%	90.75%	92.17%	87.82%	91.70%	89.06%	87.92%
HMM- EM	79.92%	88.15%	77.01%	84.57%	72.96%	86.77%	83.61%	77.20%
NCRFAE (HEM)	86.79%	92.83%	89.78%	90.68%	86.39%	91.30%	88.86%	86.55%
NCRFAE	89.43%	93.89%	90.99%	92.85%	88.93%	92.17%	89.41%	89.14%

In this example, the LSTM incorrectly predicted the POS tag of the word “search” as a verb, instead of a noun (part of the NP “nice search engine”), while predicting correctly the preceding word, “nice”, as an adjective. We attribute the error to LSTM lacking an explicit output transition scoring function, which would penalize the ungrammatical transition between “ADJ” and “VERB”.

The NCRF, which does score such transitions, correctly predicted that word. However, it incorrectly predicted “Google” as a noun rather than a proper-noun. This is a subtle mistake, as the two are grammatically and semantically similar. This mistake appeared consistently in the NCRF results, while NCRFAE predictions were correct.

We attribute this success to the superior expressivity of our model: The prediction is done jointly by the encoder and the decoder, as the reconstruction decision is defined over all output sequences, picking the jointly optimal sequence. From another perspective, our NCRFAE model is a combination of discriminative and generative models, in that sense the decoder can be regarded as a soft constraint that supple-

ments the encoder. Such that, the decoder performs as a regularizer to check-balance the choices made by the encoder.

### 4.6.3 Semi-supervised Learning

In the semi-supervised settings we compared our models with other semi-supervised structured prediction models. In addition, we studied how varying the amount of unlabeled data would influence the performance of our model.

As described in Sec. 4.5, the decoder’s parameters  $\Theta$  are initialized by the labeled dataset using real counts and updated in training.

**Varying Unlabeled Data Proportion** We first experimented with varying the proportion of unlabeled data, while fixing the amount of labeled data. We conducted these experiments over two languages, English and low-resource language Croatian. We fixed the proportion of labeled data at 20%, and gradually added more unlabeled data from 0% to 20% (from full supervision to semi-supervision). The unlabeled data was sampled from the same dataset (without overlapping with the labeled data), with the labels removed. The results are shown in Figure 4.4.

The left most point of both sub-figures is the accuracy of fully supervised learning with 20% of the whole data. As we can observe, the tagging accuracy increased as the proportion of unlabeled data increased.

**Semi-supervised POS Tagging on Multiple Languages** We compared our NCRFAE model with other semi-supervised learning models, including the HMM-EM algorithm and the hard-EM version of our NCRFAE. The hard-EM version of our model can be considered as a variant of self-training, as it infers the missing labels using the current model in the E-step, and uses the real counts of these labels to update the model in the M-step. To contextualize the results, we also provide the results of the NCRF model and the supervised version our NCRFAE model trained on 20% of the data. We set the proportion of labeled data to 20% for each language and set the

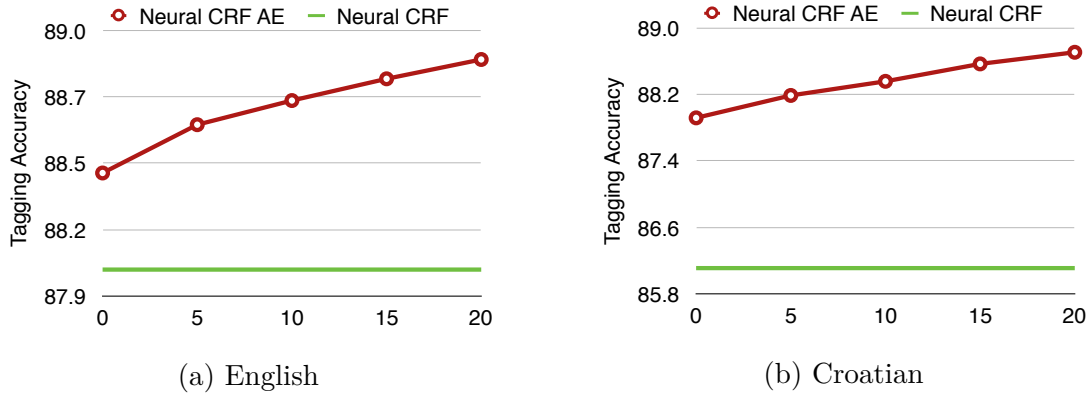


Figure 4.4.: UD English and Croatian POS tagging accuracy versus increasing proportion of unlabeled sequences using 20% labeled data. The green straight line is the performance of the neural CRF, trained over the labeled data.

proportion of unlabeled data to 50% of the dataset. There was no overlap between labeled and unlabeled data.

The results are summarized in Table 4.3. Similar to the supervised experiments, the supervised version of our NCRFAE, trained over 20% of the labeled data, outperforms the NCRF model. Our model was able to successfully use the unlabeled data, leading to improved performance in all languages, over both the supervised version of our model, as well as the HMM-EM and Hard-EM models that were also trained over both the labeled and unlabeled data.

**Varying Sizes of Labeled Data on English** As is known to all, semi-supervised approaches tend to work well when given a small size of labeled training data. But with the increase of labeled training data size, we might get diminishing effectiveness. To verify this conjecture, we conducted additional experiments to show how varying sizes of labeled training data affect the effectiveness of our NCRFAE model. In these experiments, we gradually increased the proportion of labeled data, and in accordance decreased the proportion of unlabeled data.

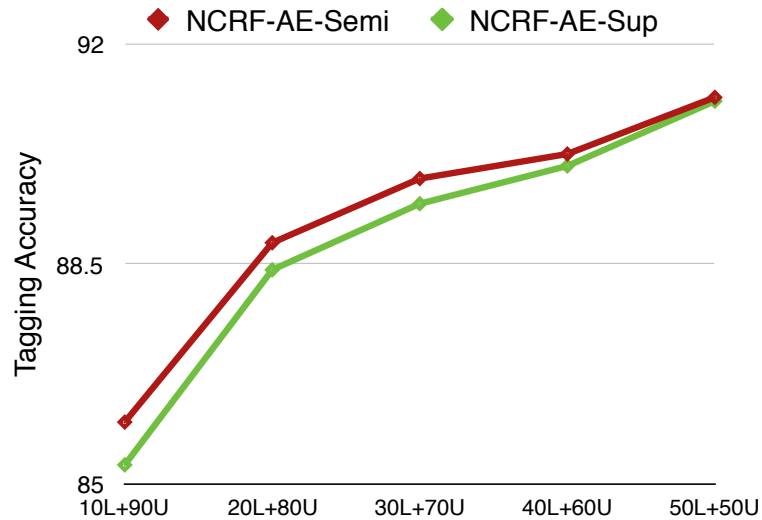


Figure 4.5.: This figure shows the performance of the NCRFAE model on different proportion of labeled and unlabeled data. The green line shows the results on only labeled data, and the red line on both labeled and unlabeled data. The difference between the red line and the green line are gradually vanishing.

The results of these experiments are demonstrated in Figure 4.5. As we speculated, we observed diminishing effectiveness when increasing the proportion of labeled data in training.

#### 4.7 Conclusion

We proposed an end-to-end neural CRF autoencoder (NCRFAE) model for semi-supervised sequence labeling. Our NCRFAE combines a discriminative component and a generative component, extending the generalized autoencoder by using a neural CRF model as its encoder and a generative decoder built on top of it. We also suggested a variant of the EM algorithm to learn the parameters of our NCRFAE model.

We evaluated our model in both supervised and semi-supervised scenarios over multiple languages, and show it can outperform other supervised and semi-supervised

methods. Additional experiments suggest how varying sizes of labeled training data affects the effectiveness of our model.

These results demonstrate the strength of our model, as it was able to utilize the small amount of labeled data and exploit the hidden information from the large amount of unlabeled data, without additional feature engineering which is often needed in order to get semi-supervised and weakly-supervised systems to perform well. The superior performance on the low-resource language also suggests its potential in practical use.



## 5 SEMI-SUPERVISED AUTOENCODING DEPENDENCY PARSING

From the perspective of linguistics, instead of as sequences, sentences can also be represented as dependency trees using dependency grammar to present enriched syntactic relationships between tokens, especially the predicate-argument structure that is frequently used in downstream tasks. Constructing tree banks for different languages is known for its high cost of hiring linguists, training annotators and compiling corpus. Semi-supervised parsing with fewer dependency tree annotation is a desirable way for training parsers to parse unseen text, especially for low-resource languages, in which the annotation is even more difficult.

In this chapter, we introduce two models for graph-based semi-supervised projective dependency parsing, namely locally autoencoding parser (LAP) and globally autoencoding parser (GAP). The LAP model assumes latent variables exists for the sentence which is used as intermediate representation to build the dependency tree; while the GAP model treats the dependency tree itself as a latent variable. During the training process, the LAP maximize the approximation of its Evidence Lower Bound (ELBO) via contrast-divergence [Carreira-Perpinan and Hinton, 2005] as that in VAE. In GAP, instead, We derived an algorithm which is a variant of the Eisner’s algorithm [Eisner, 1996] extending the EM algorithm for spanning trees, able to compute the ELBO exactly thus we maximize the ELBO directly.

By comparing the performance and running complexity with other baselines, we show LAP and GAP trade-off on speed and accuracy. The experimental results on several languages also show both LAP and GAP can use unlabeled data to assist training with labeled data, to improve on the performance using labeled data alone. In addition, GAP also outperforms several current competitors.

## 5.1 Introduction

Dependency parsing captures bi-lexical relationships by constructing directional arcs between words, defining a head-modifier syntactic structure for sentences, as shown in Figure 5.1. Dependency trees are fundamental for many downstream tasks such as semantic parsing [Reddy et al., 2016, Marcheggiani and Titov, 2017], machine translation [Bastings et al., 2017, Ding and Palmer, 2007], information extraction [Culotta and Sorensen, 2004, Liu et al., 2015] and question answering [Cui et al., 2005]. As a result, efficient parsers [Kiperwasser and Goldberg, 2016, Dozat and Manning, 2017, Dozat et al., 2017, Ma et al., 2018b] have been developed using various neural architectures.

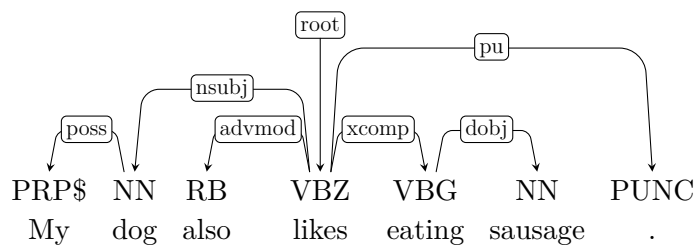


Figure 5.1.: This figure shows a dependency tree: directional arcs represent head-modifier relation between tokens.

While supervised approaches have been very successful, they require large amounts of labeled data, particularly when neural architectures are used. Syntactic annotation is notoriously difficult and requires specialized linguistic expertise, posing a serious challenge for low-resource languages. Semi-supervised parsing aims to alleviate this problem by combining a small amount of labeled data and a large amount of unlabeled data, to improve on the performance using labeled data alone. Traditional semi-supervised parsers use unlabeled data to generate additional features to assist the learning process [Koo et al., 2008], together with different variants of self-training [Søgaard, 2010]. However, these approaches are usually pipe-lined with manually crafted features, therefore error-propagation may occur.

In this study, we propose two end-to-end semi-supervised parsers, illustrated in Figure 5.2, namely Locally Autoencoding Parser (LAP) and Globally Autoencoding Parser (GAP). In LAP, the autoencoder model uses unlabeled examples to learn continuous latent variables of the sentence, which then are used to support tree inference by providing an enriched representation. Unlike LAP which does not perform tree inference when learning from unlabeled examples, in GAP, the dependency trees corresponding to the input sentences are treated as latent variables, and as a result the information learned from unlabeled data aligns better with the final tree prediction task.

Unfortunately, regarding trees as latent variables may cause the computation to be intractable, as the number of possible dependency trees to enumerate is exponentially large. A recent work [Corro and Titov, 2018], dealt with this difficulty by regenerating a particular tree of high possibility through sampling, which is Monte-Carlo estimation. In this study, we suggest a tractable algorithm for GAP, to directly compute all the possible dependency trees in an arc-decomposed manner, providing a tighter bound compared to [Corro and Titov, 2018]’s with lower time complexity. We demonstrate these advantages empirically by evaluating our model on two dependency parsing datasets. We summarize our contributions as follows:

1. We proposed two autoencoding parsers for semi-supervised dependency parsing, with complementary strengths, trading off speed vs. accuracy;
2. We propose a tractable inference algorithm to compute the expectation of the latent dependency tree analytically for GAP, which is naturally extendable to other tree-structured graphical models;
3. We show improved performance of both LAP and GAP with unlabeled data on WSJ and UD datasets on using labeled data alone, and on a recently proposed semi-supervised parser [Corro and Titov, 2018].

## 5.2 Related Work

Most dependency parsing studies fall into two major groups: graph-based and transition-based [Kubler et al., 2009]. Graph-based parsers [McDonald, 2006] regard parsing as a structured prediction problem to find the most probable tree, while transition-based parsers [Nivre, 2004, 2008] treat parsing as a sequence of actions at different stages leading to a dependency tree.

While earlier works relied on manual feature engineering, in recent years the hand-crafted features were replaced by embeddings and deep neural network architectures were used to learn representation for scoring structural decisions, leading to improved performance in both graph-based and transition-based parsing [Nivre, 2014, Pei et al., 2015, Chen and Manning, 2014, Dyer et al., 2015, Weiss et al., 2015, Andor et al., 2016, Kiperwasser and Goldberg, 2016, Wiseman and Rush, 2016].

The annotation difficulty for this task, has also motivated several works on unsupervised (grammar induction) and semi-supervised approaches to parsing [Tu and Honavar, 2012, Jiang et al., 2016, Koo et al., 2008, Li et al., 2014, Kiperwasser and Goldberg, 2015, Cai et al., 2017, Corro and Titov, 2018]. It also leads to advances in using unlabeled data for constituent grammar [Shen et al., 2018b,a] as well.

Similar to other structured prediction tasks, directly optimizing the objective is difficult when the underlying probabilistic model requires marginalizing over the dependency trees. Variational approaches are a natural way to alleviate this difficulty, as they try to improve the lower bound of the original objective, and have been applied in several recent NLP works [Stratos, 2019, Chen et al., 2018, Kim et al., 2019b,a]. Variational Autoencoder (VAE) [Kingma and Welling, 2014] is particularly useful for latent representation learning, and has been studied in semi-supervised context as the Conditional VAE (CVAE) [Sohn et al., 2015]. Note our work differs from VAE as VAE is designed for tabular data but not for structured prediction problems. Our work is more related to structured VAEs [Johnson et al., 2016].

The work mostly related to ours is Corro and Titov [2018]’s as they consider the dependency tree as the latent variable, but their work takes a second approximation to the variational lower bound by an extra sampling step to sample from the latent dependency tree, without identifying a tractable inference. We show that with the given structure, exact inference on the lower bound is achievable without approximation by sampling, which tightens the lower bound.

### 5.3 Graph-based Dependency Parsing

A dependency graph of a sentence can be regarded as a directed tree spanning all the words of the sentence, including a special “word”—the ROOT—to originate out. Assuming a sentence of length  $l$ , a dependency tree can be denoted as  $\mathcal{T} = (\langle h_1, m_1 \rangle, \dots, \langle h_{l-1}, m_{l-1} \rangle)$ , where  $h_t$  is the index in the sequence of the head word of the dependency connecting the  $t$ th word  $m_t$  as a modifier.

Our graph-based parsers are constructed by following the standard structured prediction paradigm [McDonald et al., 2005, Taskar et al., 2005]. In inference, based on the parameterized scoring function  $\mathcal{S}_\Lambda$  with parameter  $\Lambda$ , the parsing problem is formulated as finding the most probable directed spanning tree for a given sentence  $\mathbf{x}$ :

$$\mathcal{T}^* = \arg \max_{\tilde{\mathcal{T}} \in \mathbb{T}} \mathcal{S}_\Lambda(\mathbf{x}, \tilde{\mathcal{T}}),$$

where  $\mathcal{T}^*$  is the highest scoring parse tree and  $\mathbb{T}$  is the set of all valid trees for the sentence  $\mathbf{x}$ .

It is common to factorize the score of the entire graph into the summation of its substructures: the individual arc scores [McDonald et al., 2005]:

$$\mathcal{S}_\Lambda(\mathbf{x}, \tilde{\mathcal{T}}) = \sum_{(h,m) \in \tilde{\mathcal{T}}} s_\Lambda(h, m) = \sum_{t=1}^l s_\Lambda(h_t, m_t),$$

where  $\tilde{\mathcal{T}}$  represents the candidate parse tree, and  $s_\Lambda$  is a function scoring individual arcs.  $s_\Lambda(h, m)$  describes the likelihood of an arc from the head  $h$  to its modifier  $m$

in the tree. Throughout this chapter, the scoring is based on individual arcs, as we focus on first-order parsing.

### 5.3.1 Scoring Function Using Neural Architecture

We used the same neural architecture as that in Kiperwasser and Goldberg [2016]’s study. We first use a bi-LSTM model to take as input  $\mathbf{u}_t = [\mathbf{p}_t; \mathbf{e}_t]$  at position  $t$  to incorporate contextual information, by feeding the word embedding  $\mathbf{e}_t$  concatenated with the POS (part of speech) tag embeddings  $\mathbf{p}_t$  of each word. The bi-LSTM then projects  $\mathbf{u}_t$  as  $\mathbf{o}_t$ .

Subsequently a nonlinear transformation is applied on these projections. Suppose the hidden states generated by the bi-LSTM are  $[\mathbf{o}_{root}, \mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_t, \dots, \mathbf{o}_l]$ , for a sentence of length  $l$ , we compute the arc scores by introducing parameters  $\mathbf{W}_h$ ,  $\mathbf{W}_m$ ,  $\mathbf{w}$  and  $\mathbf{b}$ , and transform them as follows:

$$\begin{aligned} \mathbf{r}_t^{h-arc} &= \mathbf{W}_h \mathbf{o}_t; & \mathbf{r}_t^{m-arc} &= \mathbf{W}_m \mathbf{o}_t, \\ s_{\Lambda}(h, m) &= \mathbf{w}^{\top} (\tanh(\mathbf{r}_h^{h-arc} + \mathbf{r}_m^{m-arc} + \mathbf{b})). \end{aligned}$$

In this formulation, we first use two parameters to extract two different representations that carry two different types of information: a head seeking for its modifier (h-arc); as well as a modifier seeking for its head (m-arc). Then a nonlinear function maps them to an arc score.

For a single sentence, we can form a scoring matrix as shown in Figure 5.3, by filling each entry in the matrix using the score we obtained. Therefore, the scoring matrix is used to represent the head-modifier arc scores for all the possible arcs connecting two tokens in a sentence [Zheng, 2017].

Using the scoring arc matrix, we build graph-based parsers. Since exploring neural architectures for scoring is not our focus, we did not try other complicates, however performance shall be further improved by using advanced neural architectures [Dozat and Manning, 2017, Dozat et al., 2017].

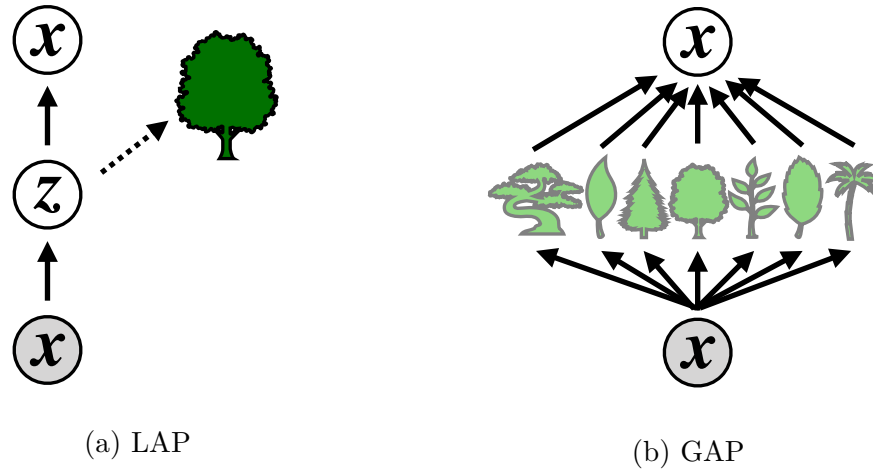


Figure 5.2.: This figure illustrates two different parsers. (a) LAP uses continuous latent variable to form the dependency tree (b) GAP treats the dependency tree as the latent variable.

	Root	$x_1$	...	$x_t$	...	$x_{L-1}$
Root	0	The score of the (t, t-1) right arc				
$x_1$		0				
$\vdots$			0			
$x_t$				0	$S_{(t, t-1)}$	
$\vdots$				$S_{(t-1, t)}$	0	
$\vdots$						0
$x_{L-1}$	The score of the (t-1, t) left arc					0

Figure 5.3.: This figure illustrates the arc scoring matrix, in which each entry represents the  $(h(head) \rightarrow m(modifier))$  score.

## 5.4 Locally Autoencoding Parser (LAP)

LAP is a fast, semi-supervised parser able to make use of unlabeled data in addition to labeled data. The illustration of this model is displayed in Figure 5.2a. LAP learns, using both labeled and unlabeled data, a continuous latent variables representation, designed to support the parsing task by creating contextualized token-representations that capture properties of the full sentence. Typically, each token in the sentence is represented by its latent variable  $z_t$ , which is a high-dimensional Gaussian variable, to be aggregated as a group of latent variables  $\mathbf{z}$ . This configuration ensures the continuous latent variable retains the contextual information from lower-level neural models to assist finding its head or its modifier; as well as forcing the representation of similar tokens to be closer. The latent variable group  $\mathbf{z}$  is modeled via  $P(\mathbf{z}|\mathbf{x})$ . In addition, we model the process of reconstructing the input sentence from the latent variable through a generative story  $P(\mathbf{x}|\mathbf{z})$ .

We adjust the original VAE setup in our semi-supervised task by considering examples with labels, similar to recent conditional variational formulations [Sohn et al., 2015, Miao and Blunsom, 2016, Zhou and Neubig, 2017]. We propose a full probabilistic model for a given sentence  $\mathbf{x}$ , with the unified objective to maximize for both supervised and unsupervised parsing as follows:

$$\mathcal{J} = \log P_{\theta}(\mathbf{x})P_{\omega}^{\epsilon}(\mathcal{T}|\mathbf{x}), \quad \epsilon = \begin{cases} 1, & \text{if } \mathcal{T} \text{ exists,} \\ 0, & \text{otherwise.} \end{cases}$$

This objective can be interpreted as follows: if the training example has a golden tree  $\mathcal{T}$  with it, then the objective is the log joint probability  $\log P_{\theta,\omega}(\mathcal{T}, \mathbf{x})$ ; if the golden tree is missing, then the objective is the log marginal probability  $\log P_{\theta}(\mathbf{x})$ . The probability of a certain tree is modeled by a tree-CRF in Eq. 2.2 with parameters  $\omega$  as  $P_{\omega}(\mathcal{T}|\mathbf{x})$ . Following the VAE inference framework, given the assumed generative



process  $P_\theta(\mathbf{x}|\mathbf{z})$ , directly optimizing this objective is intractable, instead optimize its Evidence Lower Bound (ELBO) :

$$\begin{aligned} \mathcal{J}_{lap} = & \mathbb{E}_{\mathbf{z} \sim Q_\phi(\mathbf{z}|\mathbf{x})} [\log P_\theta(\mathbf{x}|\mathbf{z})] - \mathbb{KL}(Q_\phi(\mathbf{z}|\mathbf{x})||P_\theta(\mathbf{z})) \\ & + \epsilon \mathbb{E}_{\mathbf{z} \sim Q_\phi(\mathbf{z}|\mathbf{x})} [\log P_\omega(\mathcal{T}|\mathbf{z})]. \end{aligned}$$

We show  $\mathcal{J}_{lap}$  is the ELBO of  $J$  in the appendix B.1. In practice, similar as VAE-style models,  $\mathbb{E}_{\mathbf{z} \sim Q_\phi(\mathbf{z}|\mathbf{x})} [\log P_\theta(\mathbf{x}|\mathbf{z})]$  is approximated by  $\frac{1}{N} \sum_{j=1}^N \log P_\theta(\mathbf{x}|\mathbf{z}_j)$  and  $\mathbb{E}_{\mathbf{z} \sim Q_\phi(\mathbf{z}|\mathbf{x})} [\log P_\omega(\mathcal{T}|\mathbf{z})]$  by  $\frac{1}{N} \sum_{j=1}^N \log P_\omega(\mathcal{T}|\mathbf{z}_j)$ , where  $\mathbf{z}_j$  is the  $j$ -th sample of  $N$  samples sampled from  $Q_\phi(\mathbf{z}|\mathbf{x})$ , which is a auxiliary distribution parameterized by neural architecture to approximate  $P(\mathbf{z}|\mathbf{x})$ . We describe the details of  $Q_\phi(\mathbf{z}|\mathbf{x})$  in the appendix B.2. At prediction stage, we simply use the mean  $\mu_{\mathbf{z}}$  rather than sampling  $\mathbf{z}$ .

#### 5.4.1 Incorporating POS and External Embeddings

In previous studies [Chen and Manning, 2014, Dozat and Manning, 2017, Dozat et al., 2017, Kiperwasser and Goldberg, 2016] exploring parsing using neural architectures, POS tags and external embeddings have been shown to contain important information characterizing the dependency relationship between a head and a child. Therefore, in addition to the variational autoencoding framework taking as input the randomly initialized word embeddings, optionally we can build the same structure for POS to reconstruct tags and for external embeddings to reconstruct words as well, whose variational objectives are  $\mathcal{U}_p$  and  $\mathcal{U}_e$  respectively. Hence, the final variational objective can be a combination of three:  $\mathcal{U} = \mathcal{U}_w$  (The original  $\mathcal{U}$  in Lemma B.1.1) +  $\mathcal{U}_p + \mathcal{U}_e$  (or just  $\mathcal{U} = \mathcal{U}_w + \mathcal{U}_p$  if external embeddings are not used).

#### 5.5 Globally Autoencoding Parser (GAP)

LAP autoencodes the input locally at the sequence level, focusing on the textual representation in isolation by connecting them to the parsing algorithm. A more

direct approach is to treat the entire dependency tree as a structured latent variable to reconstruct the input. This approach connects the sentence representation and the dependency tree in more immediately, thus the latent variable is handled in a straightforward manner.

Based on this motivation, we propose, globally autoencoding parser or GAP, by building a model containing both a discriminative component and a generative component to jointly learn the downstream representations and the dependency structure construction. The discriminative component builds a neural CRF model for dependency tree construction, and the generative model reconstructs the sentence from the factor graph as a Bayesian network, by assuming a generative process in which each head generates its modifier. Concretely, the latent variable in this model is the dependency tree structure.

### 5.5.1 Discriminative Component: the Encoder

We model the discriminative component in our model as  $P_{\Phi}(\mathcal{T}|\mathbf{x})$  parameterized by  $\Phi$ , taking the same form as in Eq. 2.2. Typically in our model,  $\Phi$  are the parameters of the underlying neural networks, whose architecture is described in Sec. 5.3.1.

### 5.5.2 Generative Component: the Decoder

We use a set of conditional categorical distributions to construct our Bayesian network decoder. More specifically, using the head  $h$  and modifier  $m$  notation, each head reconstructs its modifier with the probability  $P(m_t|h_t)$  for the  $t$ th word in the sentence (0th word is always the special “ROOT” word), which is parameterized by the set of parameters  $\Theta$ . Given  $\Theta$  as a matrix of  $|\mathcal{V}|$  by  $|\mathcal{V}|$ , where  $|\mathcal{V}|$  is the vocabulary size,  $\theta_{mh}$  is the item on row  $m$  column  $h$  denoting the probability that the head word  $h$  generates  $m$ . In addition, we have a simplex constraint  $\sum_{m \in \mathcal{V}} \theta_{mh} = 1$ .

The probability of reconstructing the input  $\mathbf{x}$  as modifiers  $\mathbf{m}$  in the generative process is

$$P_{\Theta}(\mathbf{m}|\mathcal{T}) = \prod_t^l P(m_t|h_t) = \prod_t^l \theta_{m_t h_t},$$

where  $l$  is the sentence length and  $P(m_t|h_t)$  represents the probability a head generates its modifier.

### 5.5.3 A Unified Supervised and Unsupervised Learning Framework

With the design of the discriminative component and the generative component of the proposed model, we have a unified learning framework for sentences with or without golden parse trees.

The complete data likelihood of a given sentence, if the golden tree is given, is

$$\begin{aligned} P_{\Theta, \Phi}(\mathbf{m}, \mathcal{T}|\mathbf{x}) &= P_{\Theta}(\mathbf{m}|\mathcal{T})P_{\Phi}(\mathcal{T}|\mathbf{x}) \\ &= \left[ \prod_{t=1}^l P(m_t|h_t) \right] \frac{\exp \mathcal{S}_{\Phi}(\mathbf{x}, \mathcal{T})}{Z(\mathbf{x})} \\ &= \frac{\exp \sum_{(h,m) \in \mathcal{T}} s'_{\Phi, \Theta}(h, m)}{Z(\mathbf{x})}, \end{aligned}$$

where  $s'_{\Phi, \Theta}(h, m) = s_{\Phi}(h, m) + \log \theta_{mh}$ , with  $\mathbf{m}$ ,  $\mathbf{x}$  and  $\mathcal{T}$  all observable.

For a unlabeled sentence, the complete data likelihood can be obtained by marginalizing over all the possible parse trees in the set  $\mathbb{T}(\mathbf{x})$ :

$$\begin{aligned} P_{\Theta, \Phi}(\mathbf{m}|\mathbf{x}) &= \sum_{\mathcal{T} \in \mathbb{T}(\mathbf{x})} P_{\Theta, \Phi}(\mathbf{m}, \mathcal{T}|\mathbf{x}) \\ &= \frac{U(\mathbf{x})}{Z(\mathbf{x})}, \end{aligned}$$

where  $U(\mathbf{x}) = \sum_{\mathcal{T} \in \mathbb{T}(\mathbf{x})} \exp \sum_{(h,m) \in \mathcal{T}} s'_{\Phi, \Theta}(h, m)$ .

We adapted a variant of Eisner [1996]’s algorithm to marginalize over all possible trees to compute both  $Z$  and  $U$ , as  $U$  has the same structure as  $Z$ , assuming a projective tree.

We use log-likelihood as our objective function. The objective for a sentence with golden tree is:

$$\begin{aligned}\mathcal{J}_l &= \log P_{\Theta, \Phi}(\mathbf{m}, \mathcal{T}|\mathbf{x}) \\ &= \sum_{(h,m) \in \mathcal{T}} s'_{\Phi, \Theta}(h, m) - \log Z(\mathbf{x})\end{aligned}$$

If the input sentence does not have an annotated golden tree, then the objective is:

$$\begin{aligned}\mathcal{J}_u &= \log P_{\Theta, \Phi}(\mathbf{m}|\mathbf{x}) \\ &= \log U(\mathbf{x}) - \log Z(\mathbf{x}).\end{aligned}\tag{5.1}$$

Thus, during training, the objective function with shared parameters is chosen based on whether the sentence in the corpus has golden parse tree or not.

#### 5.5.4 Learning

Directly optimizing the loss in Eq.5.1 is difficult when using the unlabeled data, and may lead to undesirable shallow local optima. Instead, we derive the evidence lower bound (ELBO) of  $\log P_{\Theta, \Phi}(\mathbf{m}|\mathbf{x})$  as follows, by denoting  $Q(\mathcal{T}) = P_{\Theta, \Phi}(\mathcal{T}|\mathbf{m}, \mathbf{x})$  as the posterior:

$$\begin{aligned}\log P_{\Theta, \Phi}(\mathbf{m}|\mathbf{x}) &= \log \sum_{\mathcal{T}} Q(\mathcal{T}) \frac{P_{\Theta, \Phi}(\mathbf{m}, \mathcal{T}|\mathbf{x})}{Q(\mathcal{T})} \\ &= \log \mathbb{E}_{\mathcal{T} \sim Q(\mathcal{T})} \frac{P_{\Theta, \Phi}(\mathbf{m}, \mathcal{T}|\mathbf{x})}{Q(\mathcal{T})} \\ &\geq \mathbb{E}_{\mathcal{T} \sim Q(\mathcal{T})} \log \frac{P_{\Theta, \Phi}(\mathbf{m}, \mathcal{T}|\mathbf{x})}{Q(\mathcal{T})} \\ &= \mathbb{E}_{\mathcal{T} \sim Q(\mathcal{T})} [\log P_{\Theta}(\mathbf{m}|\mathcal{T})] - \mathbb{KL} [Q(\mathcal{T})||P_{\Phi}(\mathcal{T}|\mathbf{x})].\end{aligned}$$

Therefore, instead of maximizing the log-likelihood directly, we alternatively maximize the ELBO, so our new objective function for unlabeled data becomes

$$\max_{\Theta, \Phi} \mathbb{E}_{\mathcal{T} \sim Q(\mathcal{T})} [\log P_{\Theta}(\mathbf{m}|\mathcal{T})] - \mathbb{KL} [Q(\mathcal{T})||P_{\Phi}(\mathcal{T}|\mathbf{x})].$$

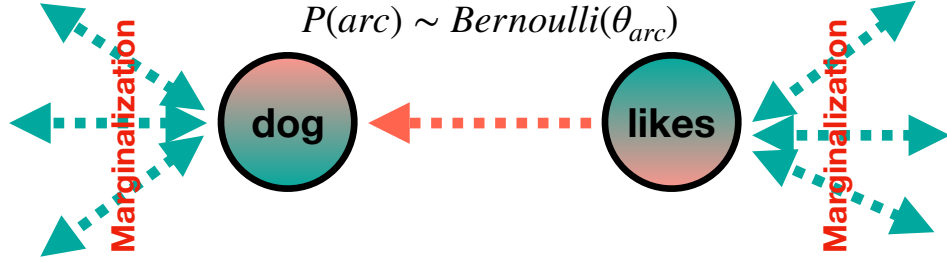


Figure 5.4.: This figure illustrates how tractable inference can be done by marginalization in a arc-decomposed manner.

Note instead of using the Monte Carlo sampling to approximate the ELBO above as done in [Corro and Titov, 2018]’s work:  $\mathbb{E}_{\mathcal{T} \sim Q(\mathcal{T})} [\log P_{\Theta}(\mathbf{m}|\mathcal{T})] - \mathbb{KL} [Q(\mathcal{T})||P_{\Phi}(\mathcal{T}|\mathbf{x})] \approx \frac{1}{N} \sum_{j=1}^N [\log P_{\Theta}(\mathbf{m}|\mathcal{T}_j)] - \mathbb{KL} [Q(\mathcal{T}_j)||P_{\Phi}(\mathcal{T}_j|\mathbf{x})]$ , we identify a tractable algorithm to calculate it directly, making the bound tighter.

In addition, to account for the unambiguity in the posterior, we incorporate entropy regularization [Tu and Honavar, 2012] when applying our algorithm, by adding an entropy term  $-\sum_{\mathcal{T}} Q(\mathcal{T}) \log Q(\mathcal{T})$  with a non-negative factor  $\sigma$  when the input sentence does not have a golden tree. Adding this regularization term is equivalent as raising the expectation of  $Q(\mathcal{T})$  to the power of  $\frac{1}{1-\sigma}$ . We annealed  $\sigma$  from the beginning of training to the end, as in the beginning, the generative model is well initialized by sentences with golden trees that resolve disambiguity.

In practice, we found the model benefits more by fixing the parameter  $\Phi$  when the data is unlabeled and optimizing the ELBO solely w.r.t. the parameter  $\Theta$ . We attribute this to the strict convexity of the ELBO w.r.t.  $\Theta$ , by sketching the proof in Appendix B.3. The details of the entire training procedure are shown in Alg. 3.

### 5.5.5 Tractable Inference

The common approach to approximate the expectation of the latent variables from the posterior distribution  $Q(\mathcal{T})$  is via sampling in VAE-type models [Kingma

---

**Algorithm 3 Learning Algorithm for GAP**


---

- 1: Initialize the parameter  $\Theta$  in the decoder with the labeled dataset  $\{\mathbf{x}, \mathcal{T}\}^l$ .
  - 2: Initialize  $\Lambda$  in the encoder randomly.
  - 3: for  $t$  in *epochs* do
  - 4: for sentence  $\mathbf{x}_i^l$  with golden parse tree  $\mathcal{T}_i^l$  in the labeled dataset  $\{\mathbf{x}, \mathcal{T}\}^l$  do
  - 5: Stochastically update the parameter  $\Lambda$  in the encoder using Adam while fixing the decoder.
  - 6: end for
  - 7: Initialize a Counting Buffer  $\mathcal{B}$
  - 8: for unlabeled sentence  $\mathbf{x}_i^u$  in the unlabeled dataset  $\{\mathbf{x}\}^u$  do
  - 9: Compute the posterior  $Q(\mathcal{T})$  in an arc factored manner for  $\mathbf{x}_i^u$  tractably.
  - 10: Compute the expectation of all possible  $(h(\text{head}) \rightarrow m(\text{modifier}))$  occurrence in the sentence  $\mathbf{x}$  based on  $Q(\mathcal{T})$ .
  - 11: Update buffer  $\mathcal{B}$  using the expectation to the power for  $\frac{1}{1-\sigma}$  of all possible  $(h \rightarrow m)$ .
  - 12: end for
  - 13: Obtain  $\Theta$  globally and analytically based on the buffer  $\mathcal{B}$  and renew the decoder.
  - 14: end for
- 

and Welling, 2014]. In a significant contrast to that, we argue in this GAP model the expectation of the latent variable (which is the dependency tree structure) is analytically tractable by designing a variant of the inside-outside algorithm [Eisner, 1996, Paskin, 2001] in an arc decomposed manner. This can be done by regarding each directed arc as an indicator variable from a Bernoulli distribution, showing whether the arc exists or not. We argue that assuming the dependency tree is projective, specialized belief propagation algorithm exists to compute the marginalization over all valid dependency trees. This algorithm also computes the expectation of each arc analytically, making inference tractable. This idea is illustrated in Figure 5.4. We

show the details of this algorithm in the appendix B.5. We also show how to calculate the expectation w.r.t the posterior distribution  $Q(\mathcal{T})$  in appendix B.4.

## 5.6 Experiments

### 5.6.1 Experimental Settings

**Datasets** First we compared our models’ performance with strong baselines on the WSJ dataset, which is the Stanford Dependency conversion [De Marneffe and Manning, 2008] of the Penn Treebank [Marcus et al., 1993]. We used the standard section split: 2-21 for training, 22 for development and 23 for testing. Second we evaluated our models on multiple languages, using datasets from UD (Universal Dependency) 2.3 [McDonald et al., 2013]. Since semi-supervised learning is particularly useful for low-resource languages, we believe those languages in UD can benefit from our approaches. The statistics of the data used in our experiments are described in Table 5.1.

To simulate the low-resource language environment, we used 10% of the whole training set as the annotated, and the rest 90% as the unlabeled.

Table 5.1.: This table shows statistics of multiple languages we used in our experiments: the numbers of sentences in the training, development and testing set.

Language	WSJ	Dutch	Spanish	English	French	Croatian	German	Italian	Russian	Japanese
Training	39832	12269	14187	2914	14450	6983	13814	13121	3850	7133
Development	1700	718	1400	707	1476	849	799	564	579	511
Testing	2416	596	426	769	416	1057	977	482	601	551

**Input Representation and Architecture** Since we use the same neural architecture in all of our models, we specify the details of the architecture once, as follows: The internal word embeddings have dimension 100 and the POS embeddings have dimension 25. The hidden layer of the bi-LSTM layer is of dimension 125. The nonlinear

Table 5.2.: In this table we compare different models on multiple languages from UD. Models were trained in a fully supervised fashion with labeled data only (noted as “L”) or semi-supervised (notes as “L+U”). “ST” stands for self-training.

Model	Dutch	Spanish	English	French	Croatian	German	Italian	Russian	Japanese
NMP (L)	76.11	82.00	75.51	83.07	77.44	74.07	82.85	75.18	93.46
NTP (L)	76.20	82.09	75.57	83.12	77.51	74.13	82.99	75.23	93.54
LAP (L)	76.15	81.93	75.36	83.09	77.45	74.14	83.07	74.84	93.38
GAP (L)	76.23	81.97	75.75	83.11	77.49	74.16	83.14	75.17	93.52
CRFAE (L+U)	71.32	74.67	68.52	77.35	69.89	68.44	76.37	68.64	87.26
ST (L+U)	75.37	80.86	72.76	81.38	76.10	73.45	82.74	72.57	91.43
LAP (L+U)	76.29	82.48	75.48	83.23	77.78	74.48	83.34	75.22	93.65
GAP (L+U)	76.54	82.56	76.21	83.26	77.83	74.63	83.54	75.69	93.92

layers used to form the head and the modifier representation both have 100 dimension. For LAP, we use separate bi-LSTMs for words and POSs. In GAP, using the “POS generating POS” decoder yields the best performance comparing with other decoder designs. This echos the finding that complicated decoders may cause “posterior collapse” [van den Oord et al., 2017, Kim et al., 2018].

**Training** In the training phase, we use Adam [Kingma and Ba, 2014] to learn all the parameters in both LAP and GAP, except the parameters in the decoder in GAP, which are learned using their global optima in each epoch. In GAP, We annealed  $\sigma$  from 1 to 0.3. We did not take efforts to tune models’ hyper-parameters and they remained the same across all the experiments. To preventing over-fitting, we applied the “early stop” strategy by using the development set.

### 5.6.2 Semi-Supervised Dependency Parsing on WSJ Dataset

We first evaluate our models on the WSJ dataset and compared the model performance with other semi-supervised parsing models, including CRFAE [Cai et al.,



2017], which is originally designed for dependency grammar induction but can be adopted for semi-supervised parsing, and “differentiable Perturb-and-Parse” parser (DPPP) [Corro and Titov, 2018]. To contextualize the results, we also experiment with the supervised neural margin-based parser (NMP) [Kiperwasser and Goldberg, 2016], neural tree-CRF parser (NTP) and the supervised version of LAP and GAP, with only the labeled data. To ensure a fair comparison, our experimental set up on the WSJ is identical as that in DPPP, using the same 100 dimension skip-gram word embeddings employed in an earlier transition-based system [Dyer et al., 2015]. We show our experimental results in Table 5.3.

Table 5.3.: This table compares the model performance on the WSJ dataset with 10% labeled data. “L” means only 10% labeled data is used, while “L+U” means both 10% labeled and 90% unlabeled data are used.

Model	UAS
DPPP[Corro and Titov, 2018](L)	88.79
DPPP[Corro and Titov, 2018](L+U)	89.50
CRFAE[Cai et al., 2017](L+U)	82.34
NMP[Kiperwasser and Goldberg, 2016](L)	89.64
NTP (L)	89.63
Self-training (L+U)	87.81
LAP (L)	89.37
LAP (L+U)	89.49
GAP (L)	89.65
GAP (L+U)	89.96

As shown in this table, both of our LAP and GAP model are able to utilize the unlabeled data to improve the overall performance on with only using labeled data. Our LAP model performs slightly worse than the NMP model, which we attribute to the increased model complexity by incorporating extra encoder and decoders to deal with the latent variable. However, our LAP model achieved comparable results

on semi-supervised parsing as the DPPP model, while our LAP model is simple and straightforward without additional inference procedure. Instead, the DPPP model has to sample from the posterior of the structure by using a “GUMBEL-MAX trick” to approximate the categorical distribution at each step, which is intensively computationally expensive. Further, our GAP model achieved the best results among all these methods, by successfully leveraging the the unlabeled data in an appropriate manner. We owe this success to such a fact: GAP is able to calculate the exact expectation of the arc-decomposed latent variable, the dependency tree structure, in the ELBO for the complete data likelihood when the data is unlabeled, rather than using Monte Carlo sampling to approximate the true expectation. Self-training using NMP with both labeled and unlabeled data is also included as a base-line, where the performance is deteriorated without appropriately using the unlabeled data.

### 5.6.3 Semi-supervised Dependency Parsing on the UD Dataset

We also evaluated our models on multiple languages from the UD data and compared the model performance with the semi-supervised version of CRFAE and the fully supervised NMP and NTP. To fully simulate the low-resource scenario, we did not use any external word embeddings while initialized them randomly.

We summarize the results in Table 5.2. First, when using labeled data only, LAP and GAP have similar performance as NMP and NTP. Second, we note that our LAP and GAP models do benefit from the unlabeled data, compared to using labeled data only. Both our LAP and GAP model are able to exploit the hidden information in the unlabeled data to improve the performance. Comparing between LAP and GAP, we notice GAP in general has better performance than LAP, and can better leverage the information in the unlabeled data to boost the performance. These results validate that GAP is especially useful for low-resource languages that are difficult to annotate. We also experimented using self-training on the labeled and unlabeled data with the

NMP model. As results show, self-training deteriorate the performance especially when the size of the training data is small.

### 5.7 Comparison of Complexity

We briefly compare the complexity of LAP, GAP and their competitors in Table 5.4. As can be seen, though all the algorithms are of  $O(l^3)$  complexity, the constant differs significantly. Our LAP model is 6 times faster than the DPPP algorithm while our GAP model is 2 times faster. Considering the model performance, our models are favored.

Table 5.4.: This table compares the complexity of different models in this table, in which  $l$  is the length of the sentence.

Model	Complexity (Train)	Complexity (Eval)
DPPP	$24l^3$	$4l^3$
CRFAE	$12l^3$	$4l^3$
NMP*	$4l^3$	$4l^3$
NTP	$4l^3$	$4l^3$
LAP	$4l^3$	$4l^3$
GAP	$12l^3$	$4l^3$

### 5.8 Conclusion

In this chapter, we present two semi-supervised parsers, which are locally autoencoding parser (LAP) and globally autoencoding parser (GAP). Both of them are end-to-end learning systems enhanced with neural architecture, capable of utilizing the latent information within the unlabeled data together with labeled data to improve the parsing performance, without using external resources. More importantly, our GAP model outperforms the previous published [Corro and Titov, 2018] semi-

---

\*Although it is the same as NTP and LAP, in fact the max operation is faster than the sum operation.

supervised parsing system on the WSJ dataset. We attribute this success to two reasons: First, our GAP model consists both a discriminative component and a generative component. These two components are constraining and supplementing each other such that the final parsing choices are made under the checks and balances to avoid over-fitting. Second, instead of sampling from posterior of the latent variable (the dependency tree) [Corro and Titov, 2018], our model analytically computes the expectation and marginalization of the latent variable, such that the global optima can be found for the decoder, which leads to an improved performance.

## 6 CONCLUSIONS

### 6.1 Summary

Structured prediction aims to predict multiple outputs with inner coherence as structured objects given the input. Structured prediction prevails in NLP as natural languages are complex structured cognitive representation developed through the evolution and interaction of the human brain and the human civilization to carry and communicate information. Structured prediction have been used in many NLP tasks, including but not limited to chunking, POS tagging, named entity recognition, constituent parsing, dependency parsing, semantic role labeling, co-reference resolution and so on, most of which are syntactic processing or shallow semantic processing.

Machine learning offers a potentially powerful approach to automatically annotate corpora through computer algorithms, but the common supervised method requires large amount of annotated training data. The manual annotation process demands annotators with linguistic expertise or domain knowledge, but still can be intensively laborious and error-prone since the annotation task is of heavy cognitive load. This difficulty motivates researchers to look for methods with reduced supervision.

The common existing approaches to structured prediction in NLP with reduced supervision can be mostly divided into three categories. The first are multi-task and modular approaches trying to aggregate auxiliary tasks either at the top output level or at the bottom input level to constrain the model towards the right direction. The second approaches are sequence learning with latent variables, which assume the chain structure of the output and try to deal with unknown output as latent variables through different manners during learning. The third approaches are the tree induction with latent variables, which relax the structure to DAGs and treat unseen edges as latent variables to learn the probabilities of a prefixed grammar set.

In this dissertation we have focused on structured prediction in NLP with reduced supervision, particularly under two scenarios: sequence labeling and dependency parsing. These two types of structured prediction problems are typical representatives of the complicated real NLP applications. We have presented three novel approaches to tackle these problems.

## 6.2 Contributions

We summarize our contributions of this dissertation as follows:

- We identified and categorized the common existing approaches to structured predictions with reduce supervision in NLP, and discussed the relations among these methods. Particularly, we pointed out that learning with latent variables approaches are more general than multi-task and modular approaches since they can deal with unlabeled data. While the tree induction with latent variable approaches are more general than sequence learning with latent variables as they are able to handle more complicated structures in NLP.
- We suggested a general modular framework for sequence learning tasks, particularly, sentiment tagging and NER but extendable to other sequence labeling tasks such as chunking and SRL. This framework is a novel weakly supervised learning approach—learning with partial labels, which exploits the modular structure with label decomposition to reduce the supervision effort. We evaluated our proposed model, in both the fully-supervised and weakly supervised scenarios, on several benchmark datasets and show our approach not only outperforms previous supervised models but with partially labeled data, also improves on the performance using the fully labeled data alone.
- We proposed the NCRFAE model together with the development of a variation of the Expectation-Maximization (EM) algorithm used for optimizing the discriminative component and the generative component of our model simul-

taneously. This model has a unified architecture with shared parameters, accommodating both unlabeled and labeled data during training. Our empirical experiments on the well-known POS tagging problem in several different languages, including low resource languages, show that NCRFAE outperforms the highly optimized competitors in both the supervised and the semi-supervised setup, and is able to utilize unlabeled data to improve on the performance using the labeled data alone.

- We introduced two autoencoding parsers for semi-supervised dependency parsing, namely, LAP and GAP with complementary strengths, trading off speed vs. accuracy. LAP treats the representation of the sentence as the latent variable while GAP treats the dependency tree structure itself as the latent variable. In addition, we identified a tractable inference algorithm to compute the expectation of the latent dependency tree analytically for GAP, thus a tighter ELBO can be optimized. Our experiments show improved performance of both LAP and GAP with unlabeled data on the WSJ and the UD datasets, and GAP outperforms existing competitors.

### 6.3 Future Work

Based on the discussion in this dissertation, we point out several interesting possible directions for future research directions.

- In Chapter 3 we proposed a modular framework with task decomposition for learning with reduced supervision and run experiments on domain adaptation. We have shown that it is possible to reuse modules in a similar task. In the context of meta learning, it is naturally to conjecture the possibility of using the learning parameters, or the distribution of them, of one task as the initialization of another task.

- In Chapter 4 we built our NCRFAE model by using a discriminative component as the encoder and a generative component as the decoder. For the generative component, it is possible to add some sort of prior distribution to inject inductive bias to help learning. A choice here is to use Dirichlet prior or modified Dirichlet prior [Tu, 2016] to induce the desired sparsity property of the grammar probabilities in the decoder.
- Posterior regularization can help constrain the direction towards which the model is converged to. In both Chapter 4 and Chapter 5 we did not explicitly apply any additional regularization on the model. It is possible to employ posterior regularization [Ganchev et al., 2010] to improve the model performance.
- With the surge of DNN in recent machine learning, traditional Bayesian statistics approaches are no more as popular as before, especially due to its unscalability. However, recent advances in Bayesian neural networks have started to connect these two methods by bringing the strengths of both. With SGLD [Welling and Teh, 2011] and its variant [Chen et al., 2014] SGHMC, which both are stochastic gradient MCMC based algorithms, Bayesian neural networks can be computed under a scalable framework with asymptotic convergence guarantee. A recent study on image recognition shows that by using a proper prior with a suitable empirical Bayesian algorithm, not only the model achieves state-of-the-art performance, but also the uncertainty can be well quantified [Deng et al., 2019]. It is possible to introduce this approach into the structured prediction in NLP with latent variables, and a well-designed prior can carry inductive bias to guide the model to learn the desirable language properties.
- Another approach we did not discuss in this dissertation is incremental learning. Active learning can be categorized into this approach, which gradually pick up examples for annotators to label but not the whole corpus. A recent study combining active learning with deep neural networks has shown to be effective for sequence labeling [Yanyao Shen et al., 2018]. It is even possible to combine



this approach with Bayesian statistics since Bayesian methods are known to be good at quantifying uncertainty and the quantified uncertainty is extremely useful in active learning.

- In Chapter 5 we described the method under the framework of projective parsing, which assumes no crossing edges exist in the dependency tree. However, some languages are known to be convoluted with non-projectivity, e.g., Arabic, Czech, Danish, Dutch and so on. Kirchhoff's Matrix-Tree Theorem [Tutte, 1984] sheds light on marginalizing all the possible dependency trees (including both projective and non-projective) in polynomial time. A possible direction is to extend our research into the scenario of non-projective parsing for better generalization.
- Another important syntactic tree is the constituent tree, which consists of terminal and non-terminals. PCFG is the grammar usually used in constituent tree constructions. DNN as effective feature extractors, can also take into account the contextual information. This breaks the context-free constraint and brings additional expressivity. It is possible to extend our research in Chapter 5 to constituent grammars with necessary modification and adjustment.
- In this dissertation we focus on structured prediction with reduced supervision in NLP. Considering the wide applications of structured prediction, it is possible to extend our approaches to other types of data, e.g., DNA/RNA sequence data, speech data and image data. These data have different characteristics than natural language data thus additional procedures or modifications are required for adaptation.

## REFERENCES

- Omri Abend, Tom Kwiatkowski, Nathaniel J. Smith, Sharon Goldwater, and Mark Steedman. Bootstrapping language acquisition. *Cognition*, 164:116–143, jul 2017.
- Waleed Ammar, Chris Dyer, and Noah A Smith. Conditional random field autoencoders for unsupervised structured prediction. In *Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2014.
- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, 2016.
- Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, 2016.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. of the International Conference on Learning Representation (ICLR)*, 2015.
- J K Baker. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132, 1979.
- Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. Graph convolutional encoders for syntax-aware neural machine translation. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2017.

- Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics*, 37(6):1554–1563, 12 1966.
- Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proc. of the International Conference on Learning Representation (ICLR)*, 2016.
- Eric Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proc. of the Third Workshop on Very Large Corpora*, 1995.
- Jiong Cai, Yong Jiang, and Kewei Tu. Crf autoencoder for unsupervised dependency parsing. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2017.
- Miguel. Carreira-Perpinan and Geoffrey. Hinton. On contrastive divergence learning. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTAT)*, 2005.
- Xavier Carreras, Lluís Màrquez, and Lluís Padró. Named entity extraction using adaboost. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, 2002.
- Rich Caruana. Multitask Learning. *Machine Learning*, 28(1):41–75, July 1997.
- Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.
- Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2014.
- Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. Variational sequential labelers for semi-supervised learning. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2018.

- Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In Proc. of the International Conference on Machine Learning (ICML), 2014.
- Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-criteria learning for chinese word segmentation. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2017.
- Yong Cheng, Yang Liu, and Wei Xu. Maximum reconstruction estimation for generative latent-variable models. In Proc. of the National Conference on Artificial Intelligence (AAAI), 2017.
- Noam Chomsky. Aspects of the Theory of Syntax. The MIT Press, 50 edition, 1965.
- Noam Chomsky. Approaching ug from below. Interfaces + recursion= language, 01 2007.
- Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text. In Proc. of the Conference on Applied Natural Language Processing (ANLC), 1988.
- Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. Semi-supervised sequence modeling with cross-view training. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2018.
- Shay Cohen and Noah A. Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2009.
- Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. Logistic normal priors for unsupervised probabilistic grammar induction. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 2008.

- Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2002.
- Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.
- Caio Corro and Ivan Titov. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. In Proc. of the International Conference on Learning Representation (ICLR), 2018.
- Timothee Cour, Ben Sapp, and Ben Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12(May), 2011.
- Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question Answering Passage Retrieval Using Dependency Relations. In Proc. of the International Conference on Research and Development in Information Retrieval (SIGIR), 2005.
- Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2004.
- Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning Journal (MLJ)*, 2009.
- Marie-Catherine De Marneffe and Christopher D Manning. The Stanford typed dependencies representation. In Proc. of the International Conference on Computational Linguistics (COLING), 2008.

- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38, 1977.
- Wei Deng, Xiao Zhang, Faming Liang, and Guang Lin. An Adaptive Empirical Bayesian Method for Sparse Deep Learning. In *Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS)*, 2019.
- Steven J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1), 1988.
- Yuan Ding and Martha Palmer. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, 2007.
- Cícero Nogueira dos Santos and Victor Guimarães. Boosting named entity recognition with neural character embeddings. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, 2015.
- Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. In *Proc. of the International Conference on Learning Representation (ICLR)*, 2017.
- Timothy Dozat, Peng Qi, and Christopher D. Manning. Stanford’s graph-based neural dependency parser at the conll 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 2017.
- Greg Durrett and Dan Klein. Neural crf parsing. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, 2015.
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, 2015.

- Jason Eisner. Three new probabilistic models for dependency parsing: An exploration. In Proc. of the International Conference on Computational Linguistics (COLING), 1996.
- David Elworthy. Does baum-welch re-estimation help taggers? In Proc. of the Conference on Applied Natural Language Processing (ANLC), 1994.
- Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. Learning to parse and translate improves neural machine translation. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2017.
- M.W. Eysenck and M.T. Keane. Cognitive Psychology: A Student's Handbook. Psychology Press, 2005.
- Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. The infinite tree. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2007.
- Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. J. Mach. Learn. Res., 11: 2001–2049, August 2010. ISSN 1532-4435.
- Norman. Geschwind and Albert M. Galaburda. Cerebral lateralization : biological mechanisms, associations, and pathology. MIT Press, 1987.
- Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar. Sparsity in dependency grammar induction. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2010.
- D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya. Multilingual Language Processing From Bytes. ArXiv e-prints, 2015.
- Fernand Gobet. Entrenchment and the Psychology of Language Learning: How We Reorganize and Adapt Linguistic Knowledge, chapter Entrenchment, gestalt formation, and chunking, pages 245–267. De Gruyter Mouton, December 2016.

- Sharon Goldwater. Nonparametric Bayesian Models of Lexical Acquisition. PhD thesis, Brown University, 2006.
- Sharon Goldwater and Thomas L. Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2007.
- I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. ArXiv e-prints, 2014.
- Stephen Grossberg. Adaptive Resonance Theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural Networks*, 37:1–47, 2013.
- Wenjuan Han, Yong Jiang, and Kewei Tu. Enhancing unsupervised generative dependency parser with contextual information. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2019.
- Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Unsupervised learning of syntactic structure with invertible neural projections. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2018.
- Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length and helmholtz free energy. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 1994.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In Proc. of the International Conference on Computer Vision (ICCV), 2017.
- Robert A. Jacobs, Michael I. Jordan, and Andrew G. Barto. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 15(2):219–250, April 1991.



- Yong Jiang, Wenjuan Han, and Kewei Tu. Unsupervised Neural Dependency Parsing. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2016.
- Mark Johnson, Thomas Griffiths, and Sharon Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2007.
- Matthew J Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. Composing graphical models with neural networks for structured representations and fast inference. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 2016.
- Arzoo Katiyar and Claire Cardie. Investigating lstms for joint extraction of opinion entities and relations. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2016.
- Yoon Kim. Convolutional neural networks for sentence classification. Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2014.
- Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In Proc. of the International Conference on Machine Learning (ICML), 2018.
- Yoon Kim, Chris Dyer, and Alexander Rush. Compound probabilistic context-free grammars for grammar induction. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2019a.
- Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. Unsupervised recurrent neural network grammars. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2019b.

- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. ArXiv e-prints, 2014.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In Proc. of the International Conference on Learning Representation (ICLR), 2014.
- Eliyahu Kiperwasser and Yoav Goldberg. Semi-supervised dependency parsing using bilexical contextual features from auto-parsed data. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2015.
- Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. Transactions of the Association for Computational Linguistics (TACL), 4:313–327, 2016.
- Dan Klein and Christopher Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2004.
- Tomás Kociský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. Semantic parsing with semi-supervised sequential autoencoders. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2016.
- Terry Koo, Xavier Carreras, and Michael Collins. Simple semi-supervised dependency parsing. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2008.
- Stephen M. Kosslyn. Seeing and Imagining in the Cerebral Hemispheres: A Computational Approach. *Psychological Review*, 94(2):148–175, 1987.
- Sandra Kubler, Ryan McDonald, Joakim Nivre, and Graeme Hirst. *Dependency Parsing*. Morgan and Claypool Publishers, 2009.

- Kenichi Kurihara and Taisuke Sato. Variational bayesian grammar induction for natural language. In Proc. of the International Conference on Grammatical Inference: Algorithms and Applications, 2006.
- John D Lafferty, Andrew McCallum, and Fernando C N Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. of the International Conference on Machine Learning (ICML), 2001.
- Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. Neural architectures for named entity recognition. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2016.
- K Lari and S J Young. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech & Language*, 4(1):35–56, 1990.
- Hao Li and Wei Lu. Learning latent sentiment scopes for entity-level sentiment analysis. In Proc. of the National Conference on Artificial Intelligence (AAAI), 2017.
- Zhenghua Li, Min Zhang, and Wenliang Chen. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2014.
- Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. The infinite PCFG using hierarchical Dirichlet processes. In Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007.
- Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. Unsupervised pos induction with word embeddings. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2015.

- Liyuan Liu, Jingbo Shang, Frank F. Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. Empower sequence labeling with task-aware neural language model. In Proc. of the National Conference on Artificial Intelligence (AAAI), 2018.
- Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. A Dependency-Based Neural Network for Relation Classification. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2015.
- Minh-Thang Luong. Multi-Task Sequence To Sequence Learning. In Proc. of the International Conference on Learning Representation (ICLR), 2016.
- Dehong Ma, Sujian Li, and Houfeng Wang. Joint learning for targeted sentiment analysis. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2018a.
- Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2016.
- Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. Stack-Pointer Networks for Dependency Parsing. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2018b.
- Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2017.
- Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. Computational Linguistics, 1993.
- Z. Marinho, A. F. T. Martins, S. B. Cohen, and N. A. Smith. Semi-supervised learning of sequence models with the method of moments. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2016.

- Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy markov models for information extraction and segmentation. In Proc. of the International Conference on Machine Learning (ICML), 2001.
- David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2006.
- Ryan McDonald. Discriminative learning and spanning tree algorithms for dependency parsing. PhD thesis, University of Pennsylvania, 2006.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2005.
- Ryan McDonald, Joakim Nivre, Yvonne Quirnbach-brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu, and Castelló Jungmee Lee. Universal dependency annotation for multilingual parsing. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2013.
- Bernard Merialdo. Tagging english text with a probabilistic model. *Comput. Linguist.*, 20(2):155–171, June 1994. ISSN 0891-2017.
- Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and et al. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 23(3):530–539, March 2015.
- Yishu Miao and Phil Blunsom. Language as a latent variable: Discrete generative models for sentence compression. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), November 2016.

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 2013.
- Mortimer Mishkin, Leslie G. Ungerleider, and Kathleen A. Macko. Object vision and spatial vision: two cortical pathways. *Trends in Neurosciences*, 6(C):414–417, January 1983.
- Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. Open domain targeted sentiment. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2013.
- T. Miyato, A. M. Dai, and I. Goodfellow. Adversarial Training Methods for Semi-Supervised Text Classification. ArXiv e-prints, 2016.
- Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. SemEval-2013 task 2: Sentiment analysis in twitter. In Second Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), 2013.
- Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2):103–134, May 2000.
- Joakim Nivre. Incrementality in deterministic dependency parsing. In Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together, 2004.
- Joakim Nivre. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 2008.

- Joakim Nivre. The inside-outside recursive neural network model for dependency parsing. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2014.
- Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. Learning multilingual named entity recognition from wikipedia. *Artif. Intell.*, 194: 151–175, January 2013.
- Mark A. Paskin. Cubic-time parsing and learning algorithms for grammatical bigram models. Technical report, EECS Department, University of California, Berkeley, 2001.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 2019.
- Lisa Pearl and Sharon Goldwater. Statistical learning, inductive bias, and bayesian inference in language acquisition. In Jeffrey Lidz, William Snyder, and Joe Pater, editors, *Oxford Handbook of Developmental Linguistics*, chapter 28, page 664–695. Oxford University Press, 2016.
- Lisa Pearl, Sharon Goldwater, and Mark Steyvers. Online learning mechanisms for bayesian models of word segmentation. *Research on Language and Computation*, 8(2):107–132, 2010.
- Wenzhe Pei, Tao Ge, and Baobao Chang. An effective neural network model for graph-based dependency parsing. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2015.

- Nanyun Peng and Mark Dredze. Improving named entity recognition for chinese social media with word segmentation representation learning. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2016.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2014.
- Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. Semeval-2016 task 5: Aspect based sentiment analysis. In Proc. of the international workshop on semantic evaluation (SemEval), 2016.
- Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Michael Collins, and Trevor Darrell. Hidden conditional random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1848–1852, October 2007. ISSN 0162-8828.
- L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL), 2009.
- Siva Reddy, Oscar Tackstrom, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics (TACL)*, pages 127–140, 2016.
- Ellen Riloff, Janyce Wiebe, and Theresa Wilson. Learning subjective nouns using extraction pattern bootstrapping. In Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL), 2003.



- Sebastian Ruder and Barbara Plank. Strong baselines for neural semi-supervised learning under domain shift. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2018.
- Jay G. Rueckl, Kyle R. Cave, and Stephen M. Kosslyn. Why are "What" and "Where" Processed by Separate Cortical Visual Systems? A Computational Investigation. *Journal of cognitive neuroscience*, 1(2):171–86, April 1989.
- Tjong Kim Sang and Erik F. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL), 2002.
- Tjong Kim Sang, Erik F., and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL), 2003.
- Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. In Proc. of the International Conference on Learning Representation (ICLR), 2018a.
- Yikang Shen, Shawn Tan, Alessandro Sordani, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In Proc. of the International Conference on Learning Representation (ICLR), 2018b.
- Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2011.
- Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. Parsing with compositional vector grammars. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2013.

- Anders Søgaard. Simple semi-supervised training of part-of-speech taggers. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2010.
- Kihyuk Sohn, Xinchun Yan, and Honglak Lee. Learning structured output representation using deep conditional generative models. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 2015.
- Valentin I Spitkovsky, Hiyun Alshawi, and Daniel Jurafsky. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2010a.
- Valentin I. Spitkovsky, Hiyun Alshawi, Daniel Jurafsky, and Christopher D. Manning. Viterbi training improves unsupervised dependency parsing. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2010b.
- Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway Networks. ArXiv e-prints, May 2015.
- Karl Stratos. Mutual Information Maximization for Simple and Accurate Part-Of-Speech Induction. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2019.
- Karl Stratos and Michael Collins. Simple semi-supervised pos tagging. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2015a.
- Karl Stratos and Michael Collins. Simple semi-supervised POS tagging. In Proceedings of the Workshop on Vector Space Modeling for Natural Language Processing, 2015b.
- Amarnag Subramanya, Slav Petrov, and Fernando Pereira. Efficient graph-based semi-supervised learning of structured tagging models. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2010.

- Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In Proc. of the International Conference on Machine Learning (ICML), 2011.
- Toshiyuki Tanaka. A theory of mean field approximation. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 1999.
- Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In Proc. of the International Conference on Machine Learning (ICML), 2005.
- Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. In Proc. of the Annual Conference of the International Speech Communication Association (INTERSPEECH), 2017.
- Kristina Toutanova and Mark Johnson. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 2007.
- Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. Un-supervised neural hidden markov models. In Proc. of the Workshop on Structured Prediction for NLP, 2016.
- I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In Proc. of the International Conference on Machine Learning (ICML), 2004.
- Kewei Tu. Modified dirichlet distribution: Allowing negative parameters to induce stronger sparsity. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2016.

- Kewei Tu and Vasant Honavar. Unambiguity regularization for unsupervised learning of probabilistic grammars. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2012.
- William Thomas Tutte. Graph Theory, volume 11. Addison-Wesley, Menlo Park, 1984.
- Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 2017.
- Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. Supertagging with lstms. In Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL), 2016.
- David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. Structured training for neural network transition-based parsing. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2015.
- Lloyd R Welch. Hidden markov models and the baum-welch algorithm. IEEE Information Theory Society Newsletter, 53(4):1,10–13, 2003.
- Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In Proc. of the International Conference on Machine Learning (ICML), 2011.
- Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), November 2016.
- Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep Active Learning for Named Entity Recognition. In Proc. of the International Conference on Learning Representation (ICLR), 2018.

- David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 1995.
- Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2018.
- Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. ArXiv e-prints, 2018.
- Matthew D. Zeiler. Adadelta: An adaptive learning rate method. CoRR, abs/1212.5701, 2012.
- Meishan Zhang, Yue Zhang, and Duy Tin Vo. Neural networks for open domain targeted sentiment. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2015.
- Xiao Zhang and Dan Goldwasser. Sentiment Tagging with Partial Labels using Modular Architectures. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2019.
- Xiao Zhang, Yong Jiang, Hao Peng, Kewei Tu, and Dan Goldwasser. Semi-supervised structured prediction with neural crf autoencoder. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2017.
- Xiaoqing Zheng. Incremental graph-based neural dependency parsing. In Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP), 2017.
- Chunting Zhou and Graham Neubig. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2017.

Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2015.

## A ADDITIONAL EXPERIMENTAL RESULTS FOR THE MODULAR NEURAL ARCHITECTURE

### A.1 Examples of Task Decomposition

In Figure A.1, we show an example of task decomposition for standard NER.

<b>Text</b>	Brush Wellman comments on beryllium lawsuits .							
<b>Tag</b>	B-ORG	I-ORG	O	O	O	O	O	O
<b>Seg</b>	B	I	O	O	O	O	O	O
<b>Ent</b>	ORG	ORG	O	O	O	O	O	O

Figure A.1.: This figure shows an example of NER decomposition.

In Figure A.2, we show another example of task decomposition for targeted sentiment, in addition to the one in the main text.

<b>Text</b>	KCConcepcionRogue Magazine Photos Continue to Get Praised by Fans onTwitter												
<b>Tag</b>	B-pos	B-pos	B-neu	E-neu	O	O	O	O	O	O	O	O	S-neu
<b>Seg</b>	B	E	B	E	O	O	O	O	O	O	O	O	S
<b>Senti</b>	pos	pos	neu	neu	O	O	O	O	O	O	O	O	neu

Figure A.2.: This figure illustrates an extra example of target sentiment decomposition.

## A.2 Full Experimental Results on Targeted Sentiment

The complete results of our experiments on the targeted sentiment task are summarized in Tab. A.1. Our LSTM-CRF-TI(g) model outperforms all the other competing models in Precision, Recall and the F1 score.

Table A.1.: This table compares the performance on the targeted sentiment task

System	Architecture	English			Spanish		
		Pre	Rec	F1	Pre	Rec	F1
Zhang, Zhang and Vo (2015)	Pipeline	43.71	37.12	40.06	45.99	40.57	43.04
	Joint	44.62	35.84	39.67	46.67	39.99	43.02
	Collapsed	46.32	32.84	38.36	47.69	34.53	40.00
Li and Lu (2017)	SS	44.57	36.48	40.11	46.06	39.89	42.75
	+embeddings	47.30	40.36	43.55	47.14	41.48	44.13
	+POS tags	45.96	39.04	42.21	45.92	40.25	42.89
	+semiMarkov	44.49	37.93	40.94	44.12	40.34	42.14
Base Line	LSTM-CRF	53.29	46.90	49.89	51.17	46.71	48.84
This work	LSTM-CRF-T	54.21	48.77	51.34	51.77	47.37	49.47
	LSTM-CRF-Ti	54.58	49.01	51.64	52.14	47.56	49.74
	LSTM-CRF-Ti(g)	55.31	49.36	52.15	52.82	48.41	50.50

## A.3 Experiments on NER

**NER datasets** We evaluated our models on three NER datasets, i.e., the English, Dutch and Spanish parts of the 2002 and 2003 CoNLL shared tasks [Sang and F., 2002, Sang et al., 2003]. We used the original division of training, validation and test sets. The task is defined over four different entity types: PERSON, LOCATION, ORGANIZATION, MISC. We used the BIOES tagging scheme during the training, and convert them back to original tagging scheme in testing, as the previous studies show that using this tagging scheme instead of BIOES can help improve performance [Ratinov and Roth, 2009, Lample et al., 2016, Ma and Hovy, 2016, Liu et al., 2018]. As



a result, the segmentation module has 5 output labels, and the entity module has 4. The final decision task, consisted of the Cartesian product of the segmentation set (BIES) and the entity set, plus the “O” tag, resulting in 17 labels totally.

**Results on NER** We compared our models with the state-of-the-art systems on English\*, Dutch and Spanish. For Dutch and Spanish, we used cross-lingual embedding as a way to exploit lexical information. The results are shown in Tab. A.2 and Tab. A.3. Our best-performing model outperforms all the competing systems.

Table A.2.: This table compares our models with several state-of-the-art systems on the CoNLL 2003 English NER dataset.

Model	English
LSTM-CRF [Lample et al., 2016]	90.94
LSTM-CNN-CRF [Ma and Hovy, 2016]	91.21
LM-LSTM-CRF [Liu et al., 2018]	91.06
LSTM-CRF-T	90.8
LSTM-CRF-TI	91.16
LSTM-CRF-TI(g)	91.68

#### A.4 Additional Experiments on Knowledge Integration

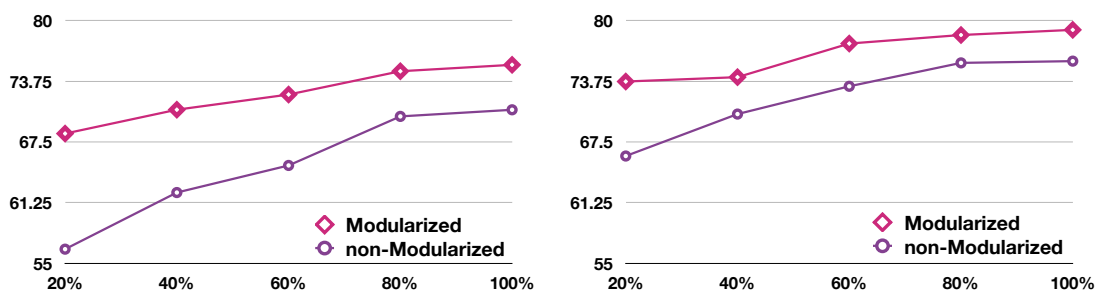
We conducted additional experiments on knowledge integration in the same setting as in the main text to investigate the properties of the modules. Figure A.3 shows the results for Dutch and Spanish NER datasets, while Figure A.4 shows the results for the Subjective Polarity Disambiguation Datasets using the in-domain data.

---

\*[Liu et al., 2018]’s results are different since their implementation did not convert the predicted BIOES tags back to BIO2 during evaluation. For fair comparison, we only report the results of the standard evaluation.

Table A.3.: This table compares our models with recent results on the 2002 CoNLL Dutch and Spanish NER datasets.

Model	Dutch	Spanish
[Carreras et al., 2002]	77.05	81.39
[Nothman et al., 2013]	78.60	N/A
[dos Santos and Guimarães, 2015]	N/A	82.21
[Gillick et al., 2015]	82.84	82.95
[Lample et al., 2016]	81.74	85.75
LSTM-CRF-T	83.91	84.89
LSTM-CRF-TI	84.12	85.28
LSTM-CRF-TI(g)	84.51	85.92



(a) Dutch NER

(b) Spanish NER

Figure A.3.: This figure shows experimental results on modular knowledge integration on the Dutch and Spanish NER datasets.

## A.5 Convergence Analysis

The proposed twofold modular infusion model (with guided gating as an option) breaks the complex learning problem into several sub-problems and then integrate

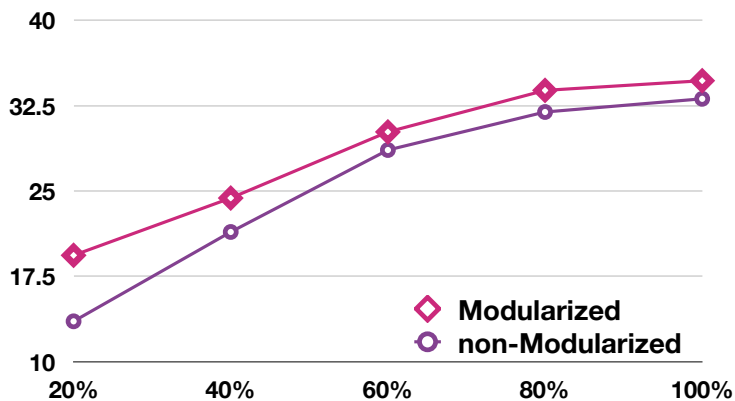


Figure A.4.: This figure shows experimental results on modular knowledge integration on the Subjective Polarity Disambiguation datasets.

them using joint training. The process defined by this formulation has more parameters and requires learning multiple objectives jointly. Our convergence analysis intends to evaluate whether the added complexity leads to a harder learning problem (i.e., slower to converge) or whether the tasks constrain each other and as a result can be efficiently learned.

We compare between our LSTM-CRF-TI(g) model and recent published top models on the English NER dataset in Figure A.5 and on the subjective polarity disambiguation datasets in Figure A.6. The curve compares convergence speed in terms of learning epochs. Our LSTM-CRF-TI(g) model has a much faster convergence rate compared to the other models.

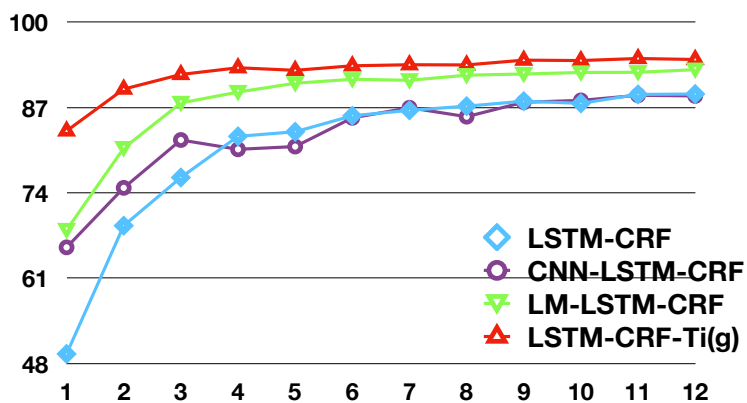


Figure A.5.: This figure compares the convergence over the development set on the English NER dataset. The x-axis is number of epochs and the y-axis is the F1-score.

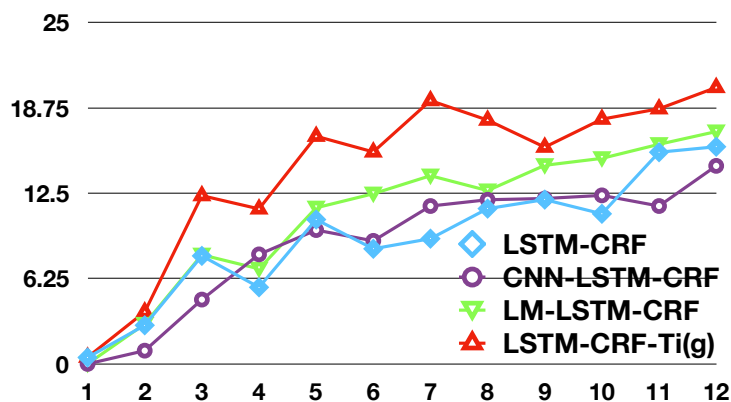


Figure A.6.: This figure compares the convergence over the development set on the subjective polarity disambiguation datasets. The x-axis is number of epochs and the y-axis is the F1-score.

## B SUPPLEMENTARY MATERIALS FOR LAP AND GAP

### B.1 ELBO of LAP's Original Objective

Lemma B.1.1  $\mathcal{J}_{lap}$  is the ELBO (evidence lower bound) of the original objective  $\mathcal{J}$ , with an input sequence  $\mathbf{x}$ .

Denote the encoder  $Q$  is a distribution used to approximate the true posterior distribution  $P_\phi(\mathbf{z}|\mathbf{x})$ , parameterized by  $\phi$  such that  $Q$  encoding the input into the latent space  $\mathbf{z}$ .

Proof

$$\begin{aligned}
\log P_\theta(\mathbf{x})P_\omega^c(\mathcal{T}|\mathbf{x}) &= \underbrace{\log P_\theta(\mathbf{x})}_{\mathcal{U}} + \underbrace{\epsilon \log P_\omega(\mathcal{T}|\mathbf{x})}_{\mathcal{L}} \\
\mathcal{U} &= \log \int_{\mathbf{z}} Q_\phi(\mathbf{z}|\mathbf{x}) \frac{P_\theta(\mathbf{x})}{Q_\phi(\mathbf{z}|\mathbf{x})} d_{\mathbf{z}} \\
&\geq \mathbb{E}_{\mathbf{z} \sim Q_\phi(\mathbf{z}|\mathbf{x})} [\log P_\theta(\mathbf{x}|\mathbf{z})] \\
&\quad - \mathbb{E}_{\mathbf{z} \sim Q_\phi(\mathbf{z}|\mathbf{x})} \left[ \log \frac{Q_\phi(\mathbf{z}|\mathbf{x})}{P_\theta(\mathbf{x})} \right] \\
&= \mathbb{E}_{\mathbf{z} \sim Q_\phi(\mathbf{z}|\mathbf{x})} [\log P_\theta(\mathbf{x}|\mathbf{z})] \\
&\quad - \text{KL}(Q_\phi(\mathbf{z}|\mathbf{x}) || P_\theta(\mathbf{z})), \\
&\quad \text{(ELBO of traditional VAE)} \\
\mathcal{L} &= \epsilon \log P_\omega(\mathcal{T}|\mathbf{x}) \\
&= \epsilon \log \int_{\mathbf{z}} P_\omega(\mathcal{T}|\mathbf{z}) Q_\phi(\mathbf{z}|\mathbf{x}) d_{\mathbf{z}} \\
&= \epsilon \log \mathbb{E}_{\mathbf{z} \sim Q_\phi(\mathbf{z}|\mathbf{x})} [P_\omega(\mathcal{T}|\mathbf{z})] \\
&\geq \epsilon \mathbb{E}_{\mathbf{z} \sim Q_\phi(\mathbf{z}|\mathbf{x})} [\log P_\omega(\mathcal{T}|\mathbf{z})].
\end{aligned}$$

Combining  $\mathcal{U}$  and  $\mathcal{L}$  leads to the fact:

$$\begin{aligned} \mathcal{U} + \mathcal{L} &\geq \mathbb{E}_{z \sim Q_\phi(z|\mathbf{x})} [\log P_\theta(\mathbf{x}|z)] - \mathbb{KL}(Q_\phi(z|\mathbf{x})||P_\theta(z)) \\ &+ \epsilon \mathbb{E}_{z \sim Q_\phi(z|\mathbf{x})} [\log P_\omega(\mathcal{T}|z)] = \mathcal{J}_{lap} \end{aligned}$$

■

## B.2 Details of the Auxiliary Posterior

### B.2.1 Mean Field Approximation and Annealing

Directly calculating the the auxiliary posterior  $Q_\phi(z|\mathbf{x})$  is difficult due to the complex model structure with deep neural networks, thus we used a mean field approximation [Tanaka, 1999] together with the conditional independence assumption by assuming  $Q_\phi(z|\mathbf{x}) = \prod_{t=1}^l Q_\phi(z_t|x_t)$  to compute it. Similarly, the generative model  $P_\theta(\mathbf{x}|z)$ , acting as a decoder parameterized by  $\theta$ , tries to regenerate the input  $x_t$  at time step  $t$  given the latent variable  $z_t$ , assuming  $P_\theta(\mathbf{x}|z) = \prod_{t=1}^l P_\theta(x_t|z_t)$ . The encoder and the decoder are trained jointly using the traditional variational autoencoder framework, by minimizing the KL divergence between the approximated posterior and the true posterior.

We parameterize the encoder  $Q_\phi(z_t|x_t)$  in two steps: First a bi-LSTM is used to obtain a non-linear transformation  $h_t$  of the original  $x_t$ ; then two separate MLPs are used to compute the mean  $\mu_{z_t}$  and the variance  $\sigma_{z_t}^2$ . The generative story  $P_\theta(x_t|z_t)$  follows such parameterization: we used a MLP of two hidden layers in-between to take  $z_t$  as the input, and predict the word (or POS tag) over the vocabulary, such that the reconstruction probability can be measured.

Following traditional VAE training paradigms, we also apply the “re-parameterization” trick [Kingma and Welling, 2014] to circumvent the non-differentiable sampling procedure to sample  $z_t$  from the  $Q_\phi(z_t|x_t)$ . Instead of directly sample from  $\mathcal{N}(\mu_{z_t}, \sigma_{z_t}^2)$ , we form  $z_t = \mu_{z_t} + \epsilon \odot \sigma_{z_t}$  by sampling  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . In addition, to avoid hindering learning during the initial training phases, following previous works [Chen et al.,

2018, Bowman et al., 2016], we anneal the temperature on the KL divergence term from a small value to 1.

### B.2.2 Empirical Bayesian Treatment

From an empirical Bayesian perspective, it is beneficial to estimate the prior distribution directly from the data by treating prior’s parameters part of the model parameters, rather than fixing the prior using some certain distributions. Similar to the approach used in the previous study [Chen et al., 2018], LAP also learns the priors from the data by updating them iteratively. We initialize the priors from a standard Gaussian distribution  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ , where  $\mathbf{I}$  is an identity matrix. During the training, the current priors are updated using the last optimized posterior, following the rule:

$$\pi_{\theta}^k(\mathbf{z}) = \sum_{\mathbf{x}} Q_{\phi}^{k-1}(\mathbf{z}|\mathbf{x})P(\mathbf{x}),$$

where  $P(\mathbf{x})$  represents the empirical data distribution, and  $k$  the iteration step. Empirical Bayesian is also named as “maximum marginal likelihood”, such that our approach here is to marginalize over the missing observation as a random variable.

### B.3 Proof of the Convexity of GAP’s ELBO

Since we only care about the term containing  $\Theta$ , the KL divergence term degenerates to a constant. For sentence  $i$ ,  $Q(\mathcal{T}_i)$  has been derived in the main text. We denote  $\mathbb{1}$  as the indication function.

$$\begin{aligned}
& \max_{\Theta} \sum_i \mathbb{E}_{\mathcal{T}_i \sim Q(\mathcal{T}_i)} [\log P_{\Theta}(\mathbf{m}_i | \mathcal{T}_i)] \\
& \quad - \mathbb{KL}[Q(\mathcal{T}_i) || P_{\Phi}(\mathcal{T}_i | \mathbf{x}_i)] \\
& \max_{\Theta} \sum_i \sum_{\mathcal{T}_i \in \mathbb{T}(\mathbf{x}_i)} Q(\mathcal{T}_i) \log P_{\Theta}(\mathbf{m}_i | \mathcal{T}_i) + Const \\
& \max_{\Theta} \sum_{(h \rightarrow m)} \log \theta_{mh} \mathbb{E}_{(h \rightarrow m) \sim Q} \mathbb{1}(h \rightarrow m) \\
& \max_{\Theta} \sum_{(h \rightarrow m)} Q(\mathbb{1}(h \rightarrow m)) \log \theta_{mh}, \tag{B.1}
\end{aligned}$$

$Q(\mathbb{1}(h \rightarrow m))$  is a Bernoulli distribution,  
indicating whether the arc  $(h \rightarrow m)$  exists.

$$s.t. \sum_m \theta_{mh} = 1 \quad \forall h. \tag{B.2}$$

#### B.4 Marginalization and Expectation of Latent Parse Trees

Light modification is needed in our study to calculate the expectation w.r.t. the posterior distribution  $Q(\mathcal{T}) = P_{\Theta, \Phi}(\mathcal{T} | \mathbf{m}, \mathbf{x})$ , as we have

$$\begin{aligned}
P_{\Theta, \Phi}(\mathcal{T} | \mathbf{m}, \mathbf{x}) &= \frac{P_{\Theta, \Phi}(\mathcal{T}, \mathbf{m} | \mathbf{x})}{P_{\Theta, \Phi}(\mathbf{m} | \mathbf{x})} \\
&= \frac{\exp^{\sum_{(h,m) \in \mathcal{T}} s'_{\Phi, \Theta}(h,m)}}{Z(\mathbf{x})} / \sum_{\mathcal{T} \in \mathbb{T}} \left[ \frac{\exp^{\sum_{(h,m) \in \mathcal{T}} s'_{\Phi, \Theta}(h,m)}}{Z(\mathbf{x})} \right] \\
&= \frac{\exp^{\sum_{(h,m) \in \mathcal{T}} s'_{\Phi, \Theta}(h,m)}}{Z'(\mathbf{x})},
\end{aligned}$$

where  $Z'(\mathbf{x}) = \sum_{\mathcal{T} \in \mathbb{T}} \exp^{\sum_{(h,m) \in \mathcal{T}} s'_{\Phi, \Theta}(h,m)}$  is the real marginal we need to calculate using the transformed scoring matrix  $\mathbf{S}'$  as input in the inside algorithm. Each entry in this transformed scoring matrix is defined in the text as  $s'_{\Phi, \Theta}(h, m)$ .



## B.5 Details of the Tractable Algorithm for GAP

Assuming the sentence is of length  $l$ , we could obtain an arc decomposed scoring matrix  $\mathbf{S}$  of size  $l \times l$ , with the entry  $\mathbf{S}[i, j]_{i \neq j, j \neq 0}$  stands for the arc score where  $i$ th word is the head and  $j$ th word the modifier.

We first describe the inside algorithm to compute the marginalization of all possible projective trees in Algo. 4.

We then describe the outside algorithm to compute the outside tables in Algo. 5. In this algorithm,  $\oplus$  stands for the logaddexp operation.

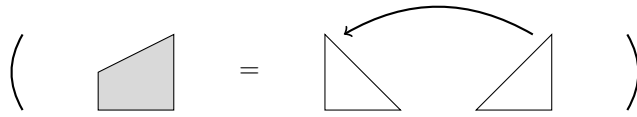
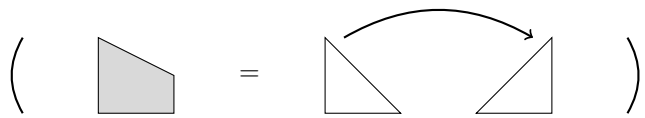
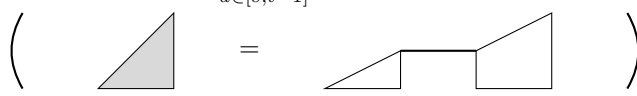
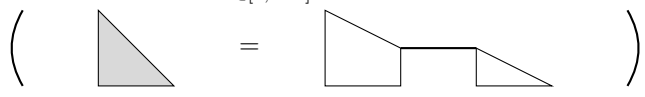
Finally, with the inside table  $\alpha$ , outside table  $\beta$  and the marginalization  $Z$  of all possible latent trees, we can compute the expectation of latent tree in an arc-decomposed manner. Algo. 6 describes the procedure. It results the matrix  $\mathbf{P}$  containing the expectation of all individual arcs by marginalize over all other arcs except itself.

---

**Algorithm 4 Inside Algorithm**


---

Input:  $\mathbf{S}$ Output:  $\alpha, Z$ 

- 1:  $\alpha \leftarrow -\infty$
  - 2: for  $s \in 0 \dots l - 1$  do
  - 3:   if  $s > 0$  then
  - 4:      $\alpha[s, s, L, C] \leftarrow 0$
  - 5:   end if
  - 6:    $\alpha[s, s, R, C] \leftarrow 0$
  - 7: end for
  - 8: for  $k \in 1 \dots l - 1$  do
  - 9:   for  $s \in 0 \dots l - k$  do
  - 10:      $t = s + k$
  - 11:     if  $s > 0$  then
  - 12:        $\alpha[s, t, L, I] \leftarrow \log \sum_{u \in [s, t-1]} \exp(\alpha[s, u, R, C] + \alpha[u + 1, t, L, C]) + \mathbf{S}[t, s]$
  - 13:       
  - 14:     end if
  - 15:      $\alpha[s, t, R, I] \leftarrow \log \sum_{u \in [s, t-1]} \exp(\alpha[s, u, R, C] + \alpha[u + 1, t, L, C]) + \mathbf{S}[s, t]$
  - 16:     
  - 17:     if  $s > 0$  then
  - 18:        $\alpha[s, t, L, C] \leftarrow \log \sum_{u \in [s, t-1]} \exp(\alpha[s, u, L, C] + \alpha[u, t, L, I])$
  - 19:       
  - 20:     end if
  - 21:      $\alpha[s, t, R, C] \leftarrow \log \sum_{u \in [s, t-1]} \exp(\alpha[s, u + 1, R, I] + \alpha[u + 1, t, R, C])$
  - 22:     
  - 23:     end for
  - 24: end for
  - 25:  $Z \leftarrow \alpha[0, l - 1, R, C]$
-




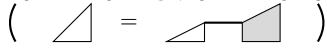




---

**Algorithm 5** Outside Algorithm
 

---

Input:  $S, \alpha$ Output:  $\beta$ 

```

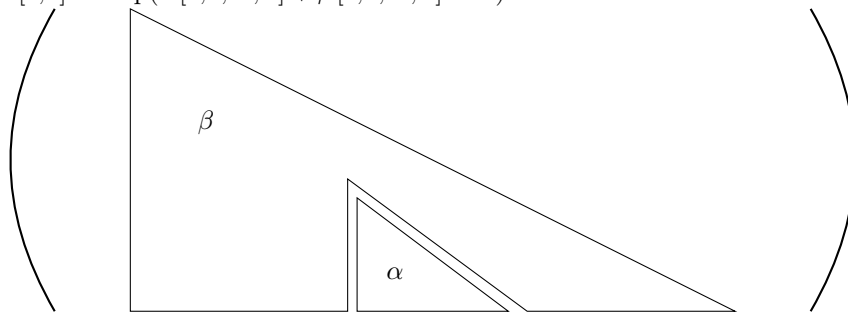
1:  $\beta \leftarrow -\infty$ 
2:  $\beta[0, l-1, R, C] \leftarrow 0$ 
3: for  $k \in l-1 \dots 1$  do
4:   for  $s \in 0 \dots l-k$  do
5:      $t = s+k$ 
6:     for  $u \in s \dots t-1$  do
7:        $\beta[s, u+1, R, I] \leftarrow \oplus(\beta[s, u+1, R, I], \beta[s, t, R, C] + \alpha[u+1, t, R, C])$ 
8:       (  )
9:     end for
10:    for  $u \in s \dots t-1$  do
11:       $\beta[u+1, t, R, C] \leftarrow \oplus(\beta[u+1, t, R, C], \beta[s, t, R, C] + \alpha[s, u+1, R, I])$ 
12:      (  )
13:    end for
14:    if  $s > 0$  then
15:      for  $u \in s \dots t-1$  do
16:         $\beta[s, u, L, C] \leftarrow \oplus(\beta[s, u, L, C], \beta[s, t, L, C] + \alpha[u, t, L, I])$ 
17:        (  )
18:      end for
19:      for  $u \in s \dots t-1$  do
20:         $\beta[u, t, L, I] \leftarrow \oplus(\beta[u, t, L, I], \beta[s, t, L, C] + \alpha[s, u, L, C])$ 
21:        (  )
22:      end for
23:    end if
24:    for  $u \in s \dots t-1$  do
25:       $\beta[s, u, R, C] \leftarrow \oplus(\beta[s, u, R, C], \beta[s, t, R, I] + \alpha[u+1, t, L, C] + S[s, t])$ 
26:      (  )
27:    end for
28:    for  $u \in s \dots t-1$  do
29:       $\beta[u+1, t, L, C] \leftarrow \oplus(\beta[u+1, t, L, C], \beta[s, t, R, I] + \alpha[s, u, R, C] + S[s, t])$ 
30:      (  )
31:    end for
32:    if  $s > 0$  then
33:      for  $u \in s \dots t-1$  do
34:         $\beta[s, u, R, C] \leftarrow \oplus(\beta[s, u, R, C], \beta[s, t, L, I] + \alpha[u+1, t, L, C] + S[t, s])$ 
35:        (  )
36:      end for
37:      for  $u \in s \dots t-1$  do
38:         $\beta[u+1, t, L, C] \leftarrow \oplus(\beta[u+1, t, L, C], \beta[s, t, L, I] + \alpha[s, u, R, C] + S[t, s])$ 
39:        (  )
40:      end for
41:    end if
42:  end for
43: end for

```

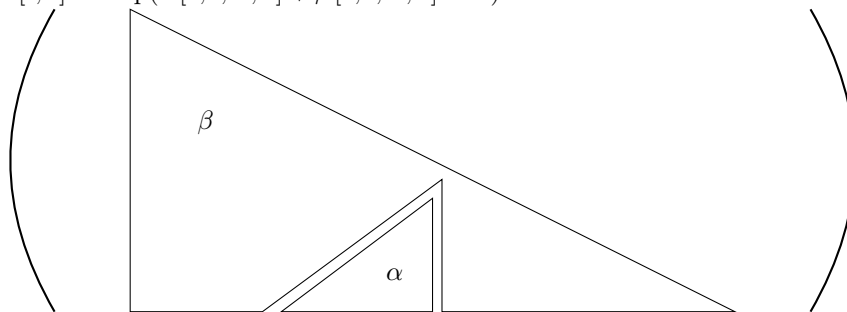
---

**Algorithm 6 Arc Decomposed Expectation**


---

Input:  $\alpha, \beta, Z$ Output:  $\mathbf{P}$ 1:  $\mathbf{P} \leftarrow 0$ 2: for  $s \in 0 \dots l - 2$  do3:   for  $t \in s + 1 \dots l - 1$  do4:     if  $s \neq t$  then5:        $\mathbf{P}[s, t] \leftarrow \exp(\alpha[s, t, R, I] + \beta[s, t, R, I] - Z)$ 

6:

7:     if  $s > 0$  then8:        $\mathbf{P}[t, s] \leftarrow \exp(\alpha[s, t, L, I] + \beta[s, t, L, I] - Z)$ 

9:

10:     end if

11:   end if

12: end for

13: end for

## VITA

Xiao Zhang is a Ph.D. student in Computer Science at Purdue University starting from 2015. Before that, he obtained a bachelor in Linguistics, a bachelor in Psychology, a master in Psychology and a joint master in Statistics and Computer Science. He has been working with Professor Dan Goldwasser in the Purdue Natural Language Processing (NLP) group during his Ph.D. studies.

His research interests are connectionism inspired machine learning methods for natural language processing, more specifically, neural networks and probabilistic models. He is particularly interested in structural models that try to understand language as a structured cognitive representation. During his Ph.D. studies, he majorly focuses on structured prediction in NLP with reduced supervision.

He has also spent time at A9.com, HP Labs and MISO.

He is joining A9.com as an applied scientist II upon his completion of his Ph.D.

## PUBLICATIONS

- Wei Deng, Xiao Zhang, Faming Liang, and Guang Lin. An Adaptive Empirical Bayesian Method for Sparse Deep Learning. In Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS), 2019.
- Dan Goldwasser and Xiao Zhang. Understanding satirical articles using common-sense. Transactions of the Association for Computational Linguistics, 4:537–549, 2016.
- I-Ta Lee, Mahak Goindani, Chang Li, Di Jin, Kristen Marie Johnson, Xiao Zhang, Maria Leonor Pacheco, and Dan Goldwasser. PurdueNLP at SemEval-2017 task 1: Predicting semantic textual similarity with paraphrase and event embeddings. In Proc. of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017.
- Maria Leonor Pacheco, I-Ta Lee, Xiao Zhang, Abdullah Khan Zehady, Pranjal Daga, Di Jin, Ayush Parolia, and Dan Goldwasser. Adapting event embedding for implicit discourse relation recognition. In Proc. of the CoNLL-16 shared task, 2016.
- Hao Peng, Shandian Zhe, Xiao Zhang, and Yuan Qi. Asynchronous distributed variational gaussian process for regression. In Proc. of the International Conference on Machine Learning (ICML), 2017.
- Xiao Zhang. Rademacher Complexity of the Restricted Boltzmann Machine. arXiv e-prints, December 2015.
- Xiao Zhang and Dan Goldwasser. Sentiment Tagging with Partial Labels using Modular Architectures. In Proc. of the Annual Meeting of the Association Computational Linguistics (ACL), 2019.
- Xiao Zhang, Kan Fang, and Gregory Francis. How to optimize switch virtual keyboards to trade off speed and accuracy. Cognitive Research: Principles and Implications, 1(1):6, 2016a.

- Xiao Zhang, Maria Leonor Pacheco, Chang Li, and Dan Goldwasser. Introducing DRAIL – a step towards declarative deep relational learning. In *In Proc. of the Workshop on Structured Prediction for NLP*, 2016b.
- Xiao Zhang, Yong Jiang, Hao Peng, Kewei Tu, and Dan Goldwasser. Semi-supervised structured prediction with neural crf autoencoder. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2017.
- Xiao Zhang, Manish Marwah, I ta Lee, Martin Arlitt, and Dan Goldwasser. Ace – an anomaly contribution explainer for cyber-security applications. In *Proc. of the IEEE International Conference on Big Data (Big Data)*, 2019.
- Xiao (Cosmo) Zhang, Kan Fang, and Gregory Francis. Optimization of switch keyboards. In *Proc. of the International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, 2013.