

Adversarial Prediction Framework
for Information Retrieval and Natural Language Processing Metrics

BY

HONG WANG

B.E., Nanjing University of Posts and Telecommunications, 2008

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2017

Chicago, Illinois

Defense Committee:

Brian D. Ziebart, Chair and Advisor

Barbara Di Eugenio

Bing Liu

Xinhua Zhang

Dan Roth, University of Illinois at Urbana-Champaign

Copyright by

HONG WANG

2017

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my Ph.D. advisor Professor Brian D. Ziebart. Professor Ziebart is a wise, patient, and kind person, like a big brother in the family that gives his genuine guidances, advices, and supports to me and all the other students in the lab. He led me into the field of Machine Learning, and encourages me to work on the topic of this dissertation. Without his guidances, this dissertation would not have been possible.

I also want to thank my former advisor Professor Clement T. Yu. Before his retirement, I got the chance to learn the knowledges in Information Retrieval and its related areas from him. Those knowledges motivate the study of this dissertation. And most of all, I learned a rigorous scholarship attitude from him that will last a lifetime.

I thank Professor Barbara Di Eugenio for her consistent kind help during my Ph.D. study. We knew each other when I was a student in her Natural Language Processing class. And I feel very fortunate to have been her Teaching Assistant of that class. Those experiences enriched my background in the field of Natural Language Processing. I appreciate the chat and encouragement she gave me during the difficult time when switching my research focus to the area of Machine Learning.

I also thank the rest of my thesis committee: Professor Bing Liu, Professor Xinhua Zhang, and Professor Dan Roth for their valuable advices to my dissertation.

I want to thank all my colleagues and friends here in Chicago. It is my pleasure to know them and spend the enjoyable six years with them together.

ACKNOWLEDGMENTS (Continued)

Lastly, I would like to thank my family. I thank my parents for bringing me to this world, giving their endless loves, providing their bests to support me for pursuing more and more achievements in my life. I thank my aunt and uncle who live in the Chicago area. It is because of them, I could enjoy a very comfortable and convenient life in this foreign land. Qi, my cousin, I appreciate not only everything he helped me during these years in the United States, but also his cares as my big brother ever since I remember. I want to thank my girlfriend, Xinyi. It is my great fortune to know her in this country far from our homeland. My life becomes more enjoyable and more exciting after I met her. I thank for her understanding, encouragement, support, and love to me.

HW

CONTRIBUTION OF AUTHORS

This thesis contains materials from a published work (1), for which I am the primary author. Wei Xing contributed the best response algorithm for discounted cumulative gain, and conducted its corresponding experiments. The Linear Programming for solving zero-sum games was revised and reimplemented based on Kaiser Asif's MATLAB version that he used in his publication (2). My advisor Professor Brian D. Ziebart contributed to discussions with respect to the work, and assisted me in writing the manuscript.

TABLE OF CONTENTS

<u>CHAPTER</u>		<u>PAGE</u>
1	INTRODUCTION	1
1.1	Notations	4
2	BACKGROUND AND RELATED WORK	5
2.1	Empirical Risk Minimization	6
2.2	Logistic Regression	7
2.2.1	Modeling	7
2.2.2	Parameter Estimation	8
2.3	Conditional Random Fields	9
2.3.1	Linear-chain CRFs Modeling	10
2.3.2	Relation to Logistic Regression	11
2.3.3	Parameter Estimation	11
2.3.4	Inference	13
2.4	Support Vector Machines	14
2.4.1	Separable Case Modeling	14
2.4.2	Non-separable Case Modeling	15
2.4.3	Parameter Estimation	16
2.5	Structured Support Vector Machines	19
2.5.1	Separable Case Modeling	19
2.5.2	Non-separable Case Modeling	20
2.5.3	Parameter Estimation	21
2.6	Two-player Zero-sum Game	24
2.6.1	Definition	24
2.6.2	Zero-sum Game Solving	26
2.7	Fisher Consistency	28
3	MULTIVARIATE PREDICTION GAME	29
3.1	Introduction	30
3.2	Architecture	32
3.3	Modeling	35
3.4	Example Multivariate Prediction Games and Small-scale So- lutions	38
3.4.1	Example Multivariate Metrics	38
3.4.2	Example Game Payoff Matrices	40
3.5	Large-scale Strategy Inference	42
3.6	Parameter Estimation	45
3.7	Finding The Best Response	46

TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
4	METRICS, BEST RESPONSE ALGORITHMS	48
4.1	Metrics in Information Retrieval and Natural Language Pro- cessing	49
4.1.1	Hamming Loss, Accuracy	49
4.1.2	F-score	50
4.1.3	Precision at k	52
4.1.4	Discounted Cumulative Gain	53
4.1.5	Alignment Error Rate	55
4.2	Best Response Algorithms	58
4.2.1	Best Response for F_1 Score	59
4.2.2	Best Response for F_1 Score with Multi-class Linear-chain Struc- ture	62
4.2.3	Best Response for Precision at k	66
4.2.4	Best Response for Discounted Cumulative Gain	68
4.2.5	Best Response for Alignment Error Rate	69
4.2.6	Best Response for Context-free Grammar Parsing	72
5	APPLICATIONS AND EXPERIMENTS	76
5.1	Binary Classification with F_1 Score	77
5.2	Named-entity Recognition with F_1 Score	79
5.3	Ranking with Precision at k and Discounted Cumulative Gain	81
5.4	Machine Translation with Alignment Error Rate	84
5.5	Syntactic Context-free Grammar Parsing with Hamming loss	87
6	CONCLUSION AND FUTURE PROSPECTS	89
6.1	Conclusion	89
6.2	Future Prospects	91
	APPENDICES	92
	CITED LITERATURE	94
	VITA	101

LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
I	The game payoff matrix between player \tilde{Y} and player \hat{Y} . Each player has three actions, where \tilde{y}_i is an action that player \tilde{Y} plays, and \hat{y}_i is an action that player \hat{Y} plays.	24
II	The game payoff matrix between player \tilde{Y} and player \hat{Y} , where player \tilde{Y} has four actions, while player \hat{Y} has two.	25
III	The payoff matrix for the zero-sum game between player \tilde{Y} choosing columns and player \hat{Y} choosing rows with three variables for <i>precision at k</i>	40
IV	The payoff matrix for the zero-sum game between player \tilde{Y} choosing columns and player \hat{Y} choosing rows with three variables for F_1	41
V	The payoff matrix for the zero-sum game between player \tilde{Y} choosing columns and player \hat{Y} choosing rows with three variables for DCG with binary relevance values $\tilde{y}_i \in \{0, 1\}$, accumulated at rank 3, and we let $\lg 3 \triangleq \log_2 3$	42
VI	Confusion matrix of a predictor.	50
VII	Sequences of the gold standard (left) and the proposed (right) source-target word alignments of the example in Figure 4.	57
VIII	F_1 score performances on OPTDIGITS and ADULT datasets. . .	78
IX	The statistics of three linear-chain datasets.	80
X	F_1 scores of CRF and MPG on ‘testa’ and ‘testb’.	81
XI	Precision@k score performances on OPTDIGITS and ADULT datasets. . .	82
XII	MQ2007 NDCG results.	83
XIII	AER of different models.	85
XIV	Results for CFG parsing on the Penn Treebank.	87

LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1	Loss functions for binary classification.	7
2	Hinge losses with slack rescaling method (red) and margin rescaling method (blue).	22
3	The architecture of Multivariate Prediction Game (MPG) framework.	32
4	The gold standard sequence alignment \mathbf{y} (top) between English words e_i and French words f_j , and the proposed sequence alignment \mathbf{a} (bottom) with AER of 5/13. Note that N tags are omitted in the figure for clarity.	56
5	An example of linear-chain structure, with $k = 2$, and the target class $C = LOC$. Green links represent the chains with the target class C , the others are links without it.	65
6	The parse tree (left) for the sentence “Margin debt was at a record high,” and its corresponding binarized parse tree (right), where ‘@NP’ is a synthetic node added to make the tree binary.	72
7	The binarized parse table for the sentence “Margin debt was at a record high.” The row number of a non-terminal symbol indicates the rule’s start index of its span of words, while the column number indicates its end index.	73
8	NDCG@k as k increases.	84

SUMMARY

Many Information Retrieval (IR) and Natural Language Processing (NLP) tasks require predicting structured objects (e.g., sequences, rankings, matchings, parse trees) that are evaluated using F-score (i.e., the harmonic mean of precision and recall), precision at k ($P@k$, which limits the number of positive predictions to k), discounted cumulative gain (DCG), alignment error rate (AER), Hamming loss (i.e., accuracy) or other multivariate performance measures. Due to the non-convexity of most of the multivariate performance metrics, and the computational intractability of optimizing empirical risk over those metrics (3), traditional Machine Learning algorithms use convex surrogates (e.g., log-loss for Logistic Regression, hinge-loss for Support Vector Machine) as the approximations for empirical risk optimization (4; 5; 6; 7; 8). However, these approximations introduce a mismatch between the learner’s objective and the desired application performance (9).

How can Machine Learning algorithms’ predictions be more closely aligned with application performance measures in Information Retrieval and Natural Language Processing? In this thesis, we focus on answering this question by building an adversarial prediction framework—*Multivariate Prediction Game (MPG)*—for the metrics that are widely used in Information Retrieval and Natural Language Processing areas. MPG treats the multivariate prediction as an adversarial zero-sum game between a loss-minimizing prediction player and a loss-maximizing evaluation player constrained to match specified properties of training data. By solving the problem of effectively finding the best responses to the opponent’s strategies, and applying

SUMMARY (Continued)

the double oracle constraint generation method, the framework avoids the non-convexity of empirical risk minimization, and more directly optimizes the metrics.

In this thesis, we first introduce the background of our research with its related works. Then, the Multivariate Prediction Game framework is explained in detail. For each metric of predicting structure, we give the corresponding algorithm for effectively finding the best responses. Finally, the MPGs are evaluated on several widely used datasets in Information Retrieval and Natural Language Processing areas to demonstrate their effectiveness.

CHAPTER 1

INTRODUCTION

Statistical supervised machine learning methods are prevalently used in a wide range of Information Retrieval (IR) and Natural Language Processing (NLP) tasks (10; 11; 12; 13; 14; 15; 16; 17; 18; 19). Those tasks require predicting structured objects (e.g., sequences, rankings, matchings, parse trees) that are commonly evaluated using F-score (i.e., the harmonic mean of precision and recall), precision at k ($P@k$, which limits the number of positive predictions to k), discounted cumulative gain (DCG), alignment error rate (AER), Hamming loss (i.e., accuracy). Unfortunately, it is computationally difficult to choose the model parameters that inductively optimize these evaluation measures for many machine learning methods. This is due to the non-concavity of evaluation measures for which maximization is desired or the non-convexity of loss functions for which the empirical risk minimizer (ERM) is sought. Indeed, even the simple zero-one loss (the classification accuracy) suffers from these computational difficulties; optimal parameter optimization is generally NP-hard (3).

Existing approaches bypass the computational difficulties arising from non-convex loss functions by replacing them with convex surrogates for which optimization is tractable. This can be viewed as *approximating the loss function and employing the exact training data*. Many common machine learning methods result from the empirical risk minimization of these convex (surrogate) loss functions. Minimizing the logarithmic loss yields logistic regression and conditional random fields (CRFs) (12). Minimizing the hinge loss of the training data yields support

vector machines (20) and structured support vector machines (21). The latter method extends SVMs to multivariate settings with the added benefit of integrating different multivariate performance measures into the hinge loss. Unfortunately, the mismatch that using a hinge loss approximation introduces degrades predictive performance in both theory (i.e., inconsistency (22)) and practice.

This thesis develops a method for taking the opposite approach to making predictions by *adversarially approximating the training data and optimizing the exact evaluation metrics* that commonly used in IR/NLP tasks: F-score for sequence prediction with/without linear-chain constraints; precision at k (P@k), and discounted cumulative gain (DCG) for rankings; alignment error rate (AER) for word alignment quality evaluation in machine translation; and Hamming loss for syntactic context-free grammar (CFG) parsing. Different from the existing work (23) that explicitly constructs and then solves a game between the classifier and the adversary with 0-1 loss, our work iteratively finds a Nash equilibrium, and hence optimizes over much larger spaces of structured objects with more complicated losses.

To demonstrate the computational feasibility, theoretical benefits, and empirical advantages of this approach, this thesis is organized as follows:

- Chapter 2 first reviews the empirical risk minimization (ERM) learning methods: logistic-regression, support vector machine (SVM), and their corresponding extensions to structure prediction: conditional random fields (CRFs), structure support vector machine (SSVM). Then it gives a brief review on *two-player zero-sum game*, and shows how to

solve such zero-sum games using Linear Programming. Fisher consistency is briefly discussed at the end of this chapter.

- Chapter 3 introduces *Multivariate Prediction Game (MPG)* framework that we proposed. Its architecture is demonstrated first, then each component is discussed in depth.
- Chapter 4 focuses on the details of implementations of *best response finding* algorithms for IR/NLP the commonly evaluation metrics.
- Chapter 5 shows the IR/NLP tasks that are benefit from directly optimizing those evaluation metrics which are discussed in this thesis. Experiments are conducted on several widely used datasets with the comparisons to ERM Machine Learning algorithms.
- Chapter 6 summarizes the contributions of this work, and discusses several possible future directions.

1.1 Notations

We consider the general task of making a multivariate prediction for variables $\mathbf{y} = \{y_1, y_2, \dots, y_n\} \in \mathbb{Y}$ (with random variables denoted as $\mathbf{Y} = \{Y_1, Y_2, \dots, Y_n\}$) given some contextual information $\mathbf{x} = \{x_1, x_2, \dots, x_n\} \in \mathbb{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n\}$ (with random variable, \mathbf{X}). Each x_i is the information relevant to predicted variable y_i . We denote the estimator's predicted values as $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n\}$. The multivariate performance measure when predicting $\hat{\mathbf{y}}$ when the true multivariate value is actually \mathbf{y} is represented as a scoring function: $\text{score}(\hat{\mathbf{y}}, \mathbf{y})$. Equivalently, a complementary loss function for any score function based on the maximal score can be defined as: $\text{loss}(\hat{\mathbf{y}}, \mathbf{y}) = \max_{\mathbf{y}', \mathbf{y}''} \text{score}(\mathbf{y}', \mathbf{y}'') - \text{score}(\hat{\mathbf{y}}, \mathbf{y})$.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we review the empirical risk minimization (ERM) learning methods: logistic-regression (with logarithmic loss), support vector machine (SVM, with hinge loss), and their corresponding extensions to structure prediction: conditional random fields (CRFs), structure support vector machine (SSVM). Then we give a brief review on two-player zero-sum game, and show how to solve such a game using Linear Programming. Fisher consistency is briefly discussed at the end of this chapter.

2.1 Empirical Risk Minimization

Empirical risk minimization (24) is a common approach for constructing predictors in supervised learning settings. This approach seeks a predictor $\hat{P}(\hat{\mathbf{y}}|\mathbf{x})$ (from, e.g., a set of predictors $\mathbf{\Gamma}$) that minimizes the loss under the empirical distribution of training data, denoted as $\tilde{P}(\mathbf{y}, \mathbf{x})$:

$$\min_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x}) \in \mathbf{\Gamma}} \mathbb{E}_{\tilde{P}(\mathbf{y}, \mathbf{x})} \left[\text{loss}(\hat{\mathbf{Y}}, \mathbf{Y}) \right]. \quad (2.1)$$

Multivariate losses are often not convex and finding the optimal solution is computationally intractable for expressive classes of predictors $\mathbf{\Gamma}$ typically specified by some set of parameters $\boldsymbol{\theta}$ (e.g., linear discriminant functions: $\hat{P}(\hat{\mathbf{y}}|\mathbf{x}) = 1$ if $\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}, \hat{\mathbf{y}}) > \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}') \forall \mathbf{y}' \neq \hat{\mathbf{y}}$).

Given these difficulties, convex surrogates to the multivariate loss are instead employed that are additive over \hat{y}_i and y_i (i.e., $\text{loss}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_i \text{loss}(\hat{y}_i, y_i)$). Employing the logarithmic loss yields the logistic regression model (25). Using the hinge loss yields support vector machines (20). Conditional random fields (26) extend logistic regression to accommodate structure information into a model. Structured support vector machines (21) employ a convex approximation (i.e., a hinge loss) that upper bounds the multivariate structured loss (See Figure 1).

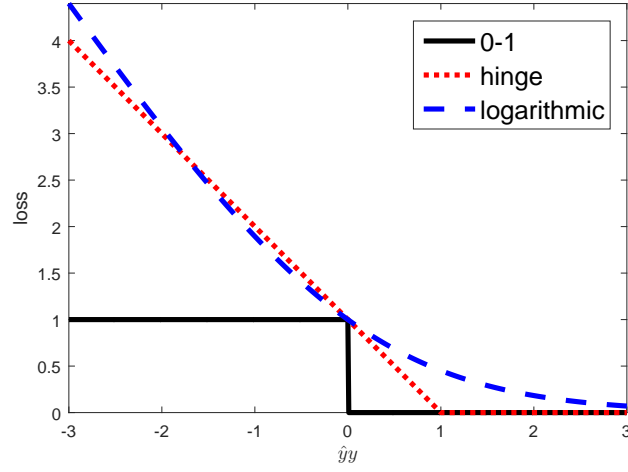


Figure 1: Loss functions for binary classification.

2.2 Logistic Regression

2.2.1 Modeling

For a binary dataset $\{y_i, \phi(\mathbf{x}_i)\}$ with K features, $y_i \in \{0, 1\}$ and $i = 1, \dots, N$, logistic regression models the likelihood probability of \mathbf{Y} under a Bernoulli distribution:

$$p(\mathbf{Y}|\boldsymbol{\theta}) = \prod_{i=1}^N \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}, \text{ where} \quad (2.2)$$

$$\hat{y}_i = \frac{1}{1 + \exp\{-\sum_{k=1}^K \theta_k \phi_k(\mathbf{x}_i)\}} \quad (2.3)$$

is a sigmoid function of $\sum_{k=1}^K \theta_k \phi_k(x_i, y_i)$, whose derivative is:

$$\frac{\partial}{\partial \theta_k} \hat{y}_i = \hat{y}_i(1 - \hat{y}_i)\phi_k(\mathbf{x}_i). \quad (2.4)$$

2.2.2 Parameter Estimation

Employing the *logarithmic loss*, which is the *negative log-likelihood* of \mathbf{Y} , gives the convex loss function of logistic regression:

$$\begin{aligned} \text{loss}(\hat{\mathbf{Y}}, \mathbf{Y}, \boldsymbol{\theta}) &= -\log p(\mathbf{Y}|\boldsymbol{\theta}) \\ &= -\sum_{i=1}^N \log \{\hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}\} \\ &= -\sum_{i=1}^N \{y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)\}. \end{aligned} \quad (2.5)$$

$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \text{loss}(\hat{\mathbf{Y}}, \mathbf{Y}, \boldsymbol{\theta})$ is computed using convex optimization methods, like gradient descent, L-BFGS, etc. (27) by plugging-in the loss's gradient whose partial derivative is:

$$\begin{aligned} \frac{\partial}{\partial \theta_k} \text{loss}(\hat{\mathbf{Y}}, \mathbf{Y}, \boldsymbol{\theta}) &= \frac{\partial \text{loss}(\hat{\mathbf{Y}}, \mathbf{Y}, \boldsymbol{\theta})}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial \theta_k} \\ &= \sum_{i=1}^N \left\{ -\frac{y_i}{\hat{y}_i} + \frac{1 - y_i}{1 - \hat{y}_i} \right\} \hat{y}_i(1 - \hat{y}_i)\phi_k(\mathbf{x}_i) \\ &= \sum_{i=1}^N (\hat{y}_i - y_i)\phi_k(\mathbf{x}_i). \end{aligned} \quad (2.6)$$

After learned $\boldsymbol{\theta}^*$ from the training data, given a new instance, Equation 2.3 can be applied directly to predict its $\hat{\mathbf{y}}$.

2.3 Conditional Random Fields

In natural language processing tasks, the content information usually helps improving the model performance. Named-entity recognition (NER) is a very representative example task. It finds and classifies named entities in text into pre-defined classes, such as PERSON, ORGANIZATION, LOCATION, and OTHER (i.e., not an entity). The variables to predict in the task is each word in a sentence, the prediction of each variable is one of the named-entity classes. Given the sentence *“Barack Obama is the president of the United States who was living in Chicago”* as an example, we want to find that *“Barack Obama”* is a PERSON, *“United States”* is an ORGANIZATION, *“Chicago”* is a LOCATION, and the rests are OTHER. Classifiers like logistic regression we discussed in the previous section, predict only one single variable. However, named-entity classes of surrounding words are sometimes dependent. For example, *“University of Illinois at Chicago”* is an ORGANIZATION, while *“Chicago”* alone is a LOCATION. Conditional random fields (CRFs) (12) are extended from logistic regression to model such dependencies.

2.3.1 Linear-chain CRFs Modeling

The simplest CRF is the linear-chain conditional random fields¹ that model the dependency between two consecutive state variables $y^{(t-1)} \in \mathbb{Y}$ and $y^{(t)} \in \mathbb{Y}$, where \mathbb{Y} is the set of all possible states:

$$p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^N \frac{1}{Z(\mathbf{x}_i, \boldsymbol{\theta})} \prod_{t=1}^T \exp\left\{\sum_{k=1}^K \theta_k \phi_k(\mathbf{x}_i, y_i^{(t-1)}, y_i^{(t)})\right\}, \quad (2.7)$$

where for each instance i in a dataset of size N , $\phi(\mathbf{x}_i, y_i^{(t-1)}, y_i^{(t)})$ is the feature vector with length K that captures the dependency between $y_i^{(t-1)}$ and $y_i^{(t)}$, T is the total number of states in a liner-chain structure, and $Z(\mathbf{x}_i, \boldsymbol{\theta})$ is an instance-specified normalization constant that sums over all possible \mathbf{y} sequences:

$$\begin{aligned} Z(\mathbf{x}_i, \boldsymbol{\theta}) &= \sum_{\mathbf{y}} \prod_{t=1}^T \exp\left\{\sum_{k=1}^K \theta_k \phi_k(\mathbf{x}_i, y^{(t-1)}, y^{(t)})\right\} \\ &= \sum_{\mathbf{y}} \exp\left\{\sum_{t=1}^T \sum_{k=1}^K \theta_k \phi_k(\mathbf{x}_i, y^{(t-1)}, y^{(t)})\right\}. \end{aligned} \quad (2.8)$$

¹The discussion of general CRF models is beyond the scope of this thesis, and can be found in (26).

2.3.2 Relation to Logistic Regression

Note that if we consider binary variable $y^{(t)} \in \{+1, -1\}$, each has its own feature vector $\boldsymbol{\theta}^{y^{(t)}}$ (i.e., $\boldsymbol{\theta}^+$, and $\boldsymbol{\theta}^-$), we discard the liner-chain relation between $y^{(t-1)}$ and $y^{(t)}$, and inspect $y^{(t)}$ individually, then for each instance i , at state t , Equation 2.7 can be rewritten as:

$$\begin{aligned} p(y_i^{(t)} = +1 | \mathbf{x}_i, \boldsymbol{\theta}) &= \frac{\exp\{\sum_{k=1}^K \theta_k^+ \phi_k(\mathbf{x}_i, y_i^{(t)})\}}{\exp\{\sum_{k=1}^K \theta_k^+ \phi_k(\mathbf{x}_i)\} + \exp\{\sum_{k=1}^K \theta_k^- \phi_k(\mathbf{x}_i)\}} \\ &= \frac{1}{1 + \exp\{\sum_{k=1}^K (\theta_k^- - \theta_k^+) \phi_k(\mathbf{x}_i)\}}, \end{aligned} \quad (2.9)$$

which is the logistic regression Equation 2.3 if let $\phi_k(\mathbf{x}_i, y_i^{(t)}) = \phi_k(\mathbf{x}_i)$ and $(\theta_k^- - \theta_k^+) = \theta_k$.

2.3.3 Parameter Estimation

The partial derivative of $\log Z(\mathbf{x}_i, \boldsymbol{\theta})$ with respect to θ_k is:

$$\begin{aligned} \frac{\partial}{\partial \theta_k} \log Z(\mathbf{x}_i, \boldsymbol{\theta}) &= \frac{1}{Z(\mathbf{x}_i, \boldsymbol{\theta})} \sum_{\mathbf{y}} \left\{ \sum_{t=1}^T \phi_k(\mathbf{x}_i, y^{(t-1)}, y^{(t)}) \right\} \exp\left\{ \sum_{t=1}^T \sum_{k=1}^K \theta_k \phi_k(\mathbf{x}_i, y^{(t-1)}, y^{(t)}) \right\} \\ &= \mathbb{E}_{\mathbf{y}} \left[\sum_{t=1}^T \phi_k(\mathbf{x}_i, y^{(t-1)}, y^{(t)}) \right] \\ &= \sum_{t=1}^T \mathbb{E}_{\mathbf{y}} \left[\phi_k(\mathbf{x}_i, y^{(t-1)}, y^{(t)}) \right] \\ &= \sum_{t=1}^T \sum_{\mathbf{y}} p(\mathbf{y} | \mathbf{x}_i) \phi_k(\mathbf{x}_i, y^{(t-1)}, y^{(t)}) \end{aligned} \quad (2.10)$$

Like logistic regression, logarithmic loss (i.e., negative log-likelihood) is defined as:

$$\begin{aligned}
\text{loss}(\mathbf{Y}, \boldsymbol{\theta}) &= -\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}) \\
&= -\sum_{i=1}^N \log \left\{ \frac{1}{Z(\mathbf{x}_i, \boldsymbol{\theta})} \prod_{t=1}^T \exp \sum_{k=1}^K \theta_k \phi_k(\mathbf{x}_i, y_i^{(t-1)}, y_i^{(t)}) \right\} \\
&= -\sum_{i=1}^N \sum_{t=1}^T \sum_{k=1}^K \theta_k \phi_k(\mathbf{x}_i, y_i^{(t-1)}, y_i^{(t)}) + \sum_{i=1}^N \log Z(\mathbf{x}_i, \boldsymbol{\theta}), \tag{2.11}
\end{aligned}$$

Then, after plugging in Equation 2.10 and Equation 2.11, $\text{loss}(\mathbf{y}, \boldsymbol{\theta})$'s partial derivative with respect to θ_k can be expressed as:

$$\frac{\partial}{\partial \theta_k} \text{loss}(\mathbf{Y}, \boldsymbol{\theta}) = -\sum_{i=1}^N \sum_{t=1}^T \phi_k(\mathbf{x}_i, y_i^{(t-1)}, y_i^{(t)}) + \underbrace{\sum_{i=1}^N \sum_{t=1}^T \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}_i) \phi_k(\mathbf{x}_i, y^{(t-1)}, y^{(t)})}_{\zeta}. \tag{2.12}$$

The first term in Equation 2.12 is the expected feature value under empirical distribution. The second term, ζ , sums over all possible \mathbf{y} , can be reformulated as:

$$\begin{aligned}
\zeta &= \sum_{t=1}^T \sum_{a,b \in \mathcal{Y}} \sum_{\mathbf{y}: y^{(t-1)}=a, y^{(t)}=b} p(\mathbf{y}|\mathbf{x}_i) \phi_k(\mathbf{x}_i, y^{(t-1)}, y^{(t)}) \\
&= \sum_{t=1}^T \sum_{a,b \in \mathbb{Y}} \phi_k(\mathbf{x}_i, y^{(t-1)}=a, y^{(t)}=b) \sum_{\mathbf{y}: y^{(t-1)}=a, y^{(t)}=b} p(\mathbf{y}|\mathbf{x}_i) \\
&= \sum_{t=1}^T \sum_{a,b \in \mathbb{Y}} \phi_k(\mathbf{x}_i, y^{(t-1)}=a, y^{(t)}=b) p_{\mathbf{y}_i}(a, b), \tag{2.13}
\end{aligned}$$

where for i -th training instance, $p_{\mathbf{y}_i}(a, b)$ is the marginal probability of having a at state $(t-1)$, and b at state t . It can be computed together with $Z(\mathbf{x}_i, \boldsymbol{\theta})$ in Equation 2.11 using *forward*-

backward algorithm in $\mathcal{O}(T|\mathbb{Y}|^2)$ time (26). Convex optimization methods can be applied to minimize loss function (Equation 2.11) to get $\boldsymbol{\theta}^*$.

2.3.4 Inference

When predicting a new instance $\phi(\mathbf{x})$ with the learned $\boldsymbol{\theta}^*$, we want to find the \mathbf{y}^* that maximizes $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}^*)$

$$\begin{aligned}
 \mathbf{y}^* &= \arg \max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}^*) \\
 &= \arg \max_{\mathbf{y}} \frac{1}{Z(\mathbf{x}, \boldsymbol{\theta}^*)} \prod_{t=1}^T \exp\left\{\sum_{k=1}^K \theta_k^* \phi_k(\mathbf{x}, y^{(t-1)}, y^{(t)})\right\} \\
 &= \arg \max_{\mathbf{y}} \prod_{t=1}^T \exp\left\{\sum_{k=1}^K \theta_k^* \phi_k(\mathbf{x}, y^{(t-1)}, y^{(t)})\right\}.
 \end{aligned} \tag{2.14}$$

It can be solved by Viterbi algorithm, which is a dynamic programming algorithm that can be found in (26).

2.4 Support Vector Machines

2.4.1 Separable Case Modeling

Support vector machines learn a decision boundary that maximizes the margin, which in turn is the smallest distance between the decision boundary and any of the training examples. For a size N linear-separable dataset $\{y_i, \phi(\mathbf{x}_i)\}$, $y_i \in \{-1, 1\}$, a linear model is defined as:

$$\hat{y}_i = \boldsymbol{\theta}^\top \phi(\mathbf{x}_i, y_i) = \sum_{k=1}^K \theta_k \phi_k(\mathbf{x}_i), \quad (2.15)$$

where $\boldsymbol{\theta}$ are the parameters to learn. If the boundary is defined to be $\hat{y}_i = 0$, all positive data points (i.e., $y_i = 1$) have $\hat{y}_i > 0$, and all negative data points (i.e., $y_i = -1$) have $\hat{y}_i < 0$, hence $y_i \hat{y}_i > 0$ for all training examples. $\boldsymbol{\theta}$ is perpendicular to the decision boundary because $\hat{y}_i = \boldsymbol{\theta}^\top \phi(\mathbf{x}_i) = 0$, for any two data points $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$, $\boldsymbol{\theta}^\top (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) = 0$. So, the distance for a data point to the decision boundary is expressed as $|\hat{y}_i|/||\boldsymbol{\theta}||$, and it can be rewritten as $y_i \hat{y}_i / ||\boldsymbol{\theta}|| = y_i \boldsymbol{\theta}^\top \phi(\mathbf{x}_i) / ||\boldsymbol{\theta}||$, since $y_i \in \{-1, 1\}$. If we set the constraint $y_i \boldsymbol{\theta}^\top \phi(\mathbf{x}_i) \geq 1$, which doesn't change the value of distance, then the problem of finding the maximum margin (distance) can be formulated as:

$$\max_{\boldsymbol{\theta}} \frac{1}{||\boldsymbol{\theta}||} \quad (2.16)$$

$$\text{subject to: } y_i \boldsymbol{\theta}^\top \phi(\mathbf{x}_i) \geq 1,$$

which is equivalent to:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \frac{||\boldsymbol{\theta}||^2}{2} \\ \text{subject to:} \quad & y_i \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_i) \geq 1. \end{aligned} \tag{2.17}$$

2.4.2 Non-separable Case Modeling

If the dataset is not linearly separable, we introduce a penalty for each data point i , to allow it being incorrectly separated by a decision boundary:

$$(1 - y_i \hat{y}_i)_+ = \max(0, 1 - y_i \hat{y}_i). \tag{2.18}$$

It upper bounds the 0-1 loss (the number of misclassified instances), and is called hinge loss due to its shape (see Figure 1). Then the optimization of SVM model becomes:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^N (1 - y_i \hat{y}_i)_+ + \frac{\lambda}{2} ||\boldsymbol{\theta}||^2, \tag{2.19}$$

where λ is the regularization parameter that controls the trade-off between the penalty (i.e., hinge loss) and the margin width. $||\boldsymbol{\theta}||^2$ is also called the ℓ_2 regularizer if we consider Equation 2.19 as a minimization of a regularized hinge loss (28).

2.4.3 Parameter Estimation

Due to the max function in hinge loss is not differentiable, we rewrite Equation 2.19 as following with a *slack variable* ξ_i for each instance:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \sum_{i=1}^N \xi_i + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \quad (2.20)$$

$$\text{subject to: } \forall i : \xi_i \geq 0, \xi_i \geq 1 - y_i \hat{y}_i.$$

Then its corresponding primal Lagrangian is written as:

$$L_P(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^N \xi_i + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 - \sum_{i=1}^N \alpha_i \{y_i \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_i) - 1 + \xi_i\} - \sum_{i=1}^N \beta_i \xi_i, \quad (2.21)$$

where α, β are Lagrange multipliers. Then the Lagrange dual problem is:

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \inf_{\boldsymbol{\theta}, \boldsymbol{\xi}} L_P(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \quad (2.22)$$

$$\text{subject to: } \forall i : \alpha_i \geq 0, \beta_i \geq 0.$$

Because Karush-Kuhn-Tucker (KKT) conditions hold for the optimization problem in Equation 2.20 (29), we have the followings:

$$\frac{\partial}{\partial \theta_k} L_P(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \lambda \theta_k - \sum_{i=1}^N \alpha_i y_i \phi_k(\mathbf{x}_i) = 0, \quad (2.23)$$

$$\frac{\partial}{\partial \xi_i} L_P(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = 1 - \alpha_i - \beta_i = 0, \quad (2.24)$$

$$y_i \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_i) - 1 + \xi_i \geq 0, \quad (2.25)$$

$$\xi_i \geq 0, \quad (2.26)$$

$$\alpha_i \geq 0, \quad (2.27)$$

$$\beta_i \geq 0, \quad (2.28)$$

$$\alpha_i \{y_i \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_i) - 1 + \xi_i\} = 0, \quad (2.29)$$

$$\beta_i \xi_i = 0. \quad (2.30)$$

Substitute Equation 2.23 and Equation 2.24 into the Lagrangian (Equation 2.21), we get:

$$\begin{aligned} L_D(\boldsymbol{\alpha}) &= \inf_{\boldsymbol{\theta}, \boldsymbol{\xi}} L_P(\boldsymbol{\theta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\ &= \sum_{i=1}^N \xi_i + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 - \sum_{i=1}^N \alpha_i y_i \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_i) + \sum_{i=1}^N \alpha_i - \sum_{i=1}^N \alpha_i \xi_i - \sum_{i=1}^N (1 - \alpha_i) \xi_i \\ &= \sum_{i=1}^N \alpha_i + \frac{\lambda}{2} \left[\frac{\sum_{i=1}^N \alpha_i y_i \phi_k(\mathbf{x}_i)}{\lambda} \right]^2 - \sum_{i=1}^N \alpha_i y_i \frac{\sum_{j=1}^N \alpha_j y_j \phi_k^\top(\mathbf{x}_j)}{\lambda} \boldsymbol{\phi}(\mathbf{x}_i) \end{aligned} \quad (2.31)$$

$$= \sum_{i=1}^N \alpha_i - \frac{1}{2\lambda} \sum_{i=1, j=1}^N \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}^\top(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_j). \quad (2.32)$$

Then, the Lagrange dual problem becomes:

$$\max_{\alpha} \sum_{i=1}^N \alpha_i - \frac{1}{2\lambda} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \phi^{\top}(\mathbf{x}_i) \phi(\mathbf{x}_j) \quad (2.33)$$

subject to: $\forall i : 0 \leq \alpha_i \leq 1$.

Lagrange multipliers $\boldsymbol{\alpha}^*$ in this dual problem can be obtained by solving the quadratic programming (29), hence parameters $\boldsymbol{\theta}^*$ can be obtained through Equation 2.23.

After learning $\boldsymbol{\theta}^*$ from the training data, given a new instance \mathbf{x} , it can be classified using the following equation:

$$\hat{y} = \text{sign}(\boldsymbol{\theta}^{*\top} \phi(\mathbf{x})). \quad (2.34)$$

2.5 Structured Support Vector Machines

2.5.1 Separable Case Modeling

Extending the SVM model described in the previous section, structured support vector machines (SSVMs) aim to model structured variables $\mathbf{y} \in \mathbf{Y}$ from a size N dataset $\{\mathbf{y}_i, \phi(\mathbf{x}_i, \mathbf{y}_i)\}$ with the large margin approach. For a given \mathbf{y} , we define $f(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$ to be a linear combination of its corresponding K features:

$$f(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \phi(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^K \theta_k \phi_k(\mathbf{x}, \mathbf{y}). \quad (2.35)$$

For each instance i , we want to use this $f(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$ to distinguish \mathbf{y}_i from any other $\mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i$ as much as possible. We define the margin for instance i as following:

$$\Delta f_i(\boldsymbol{\theta}) = f(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta}) - \max_{\mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i} f(\mathbf{x}_i, \mathbf{y}, \boldsymbol{\theta}) \quad (2.36)$$

$$= \boldsymbol{\theta}^\top \phi(\mathbf{x}_i, \mathbf{y}_i) - \max_{\mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i} \boldsymbol{\theta}^\top \phi(\mathbf{x}_i, \mathbf{y}). \quad (2.37)$$

The margin width in the direction $\boldsymbol{\theta}/\|\boldsymbol{\theta}\|$ is $|\Delta f_i(\boldsymbol{\theta})|/\|\boldsymbol{\theta}\|$. Note that rescaling $\boldsymbol{\theta}$ can give arbitrary value of $\Delta f_i(\boldsymbol{\theta})$. Like SVMs, we restrict $\Delta f_i(\boldsymbol{\theta}) \geq 1$. Then we get the following optimization problem:

$$\max_{\boldsymbol{\theta}} \frac{1}{\|\boldsymbol{\theta}\|} \quad (2.38)$$

subject to: $\forall i : \Delta f_i(\boldsymbol{\theta}) \geq 1$

which is equivalent to:

$$\min_{\boldsymbol{\theta}} \frac{\|\boldsymbol{\theta}\|^2}{2} \quad (2.39)$$

$$\text{subject to: } \forall i, \forall \mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i : \boldsymbol{\theta}^\top [\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y})] \geq 1.$$

2.5.2 Non-separable Case Modeling

Similar to SVMs, a penalty (which is a hinge loss) is introduced for each data point i , to accommodate the violation of the margin width constraint and combine the structure loss $\Delta(\mathbf{y}_i, \mathbf{y})$ together:

$$\Delta(\mathbf{y}_i, \mathbf{y}) [1 - \Delta f_i(\boldsymbol{\theta})]_+ = \max\{0, \Delta(\mathbf{y}_i, \mathbf{y}) [1 - \Delta f_i(\boldsymbol{\theta})]\}. \quad (2.40)$$

This is called *slack rescaling* method (21), where the optimization problem becomes:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \sum_{i=1}^N \xi_i + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \quad (2.41)$$

$$\text{subject to: } \forall i, \forall \mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i : \xi_i \geq 0$$

$$\xi_i \geq \Delta(\mathbf{y}_i, \mathbf{y})(1 - \boldsymbol{\theta}^\top [\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y})])$$

There's another method to combine the structure loss with margin width constraint violation, and is called *margin rescaling* (30). Instead of restricting $\Delta f_i(\boldsymbol{\theta}) \geq 1$ in Equation 2.38, margin rescaling method restrict $\Delta f_i(\boldsymbol{\theta}) \geq \Delta(\mathbf{y}_i, \mathbf{y})$, therefore its hinge loss becomes:

$$[\Delta(\mathbf{y}_i, \mathbf{y}) - \Delta f_i(\boldsymbol{\theta})]_+ = \max\{0, \Delta(\mathbf{y}_i, \mathbf{y}) - \Delta f_i(\boldsymbol{\theta})\}, \quad (2.42)$$

and the optimization problem can be expressed as:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \sum_{i=1}^N \xi_i + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \quad (2.43)$$

$$\text{subject to: } \forall i, \forall \mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i : \xi_i \geq 0$$

$$\xi_i \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \boldsymbol{\theta}^\top [\boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y})].$$

The shapes of hinge losses of these two scaling methods can be found in Figure 2.

2.5.3 Parameter Estimation

Applying KKT conditions, one can show that the slack rescaling method's Lagrange dual problem becomes:

$$\max_{\alpha} \sum_{i=1}^N \sum_{\mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i} \alpha_{i\mathbf{y}} - \frac{1}{2\lambda} \sum_{i=1}^N \sum_{\substack{\mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i \\ \mathbf{y}' \in \mathbf{Y} \setminus \mathbf{y}_j}} \alpha_{i\mathbf{y}} \alpha_{j\mathbf{y}'} \Delta \boldsymbol{\phi}_i(\mathbf{y}) \Delta \boldsymbol{\phi}_j(\mathbf{y}') \quad (2.44)$$

$$\text{subject to: } \forall i, \mathbf{y} : 0 \leq \alpha_{i\mathbf{y}} \leq \Delta(\mathbf{y}_i, \mathbf{y}),$$

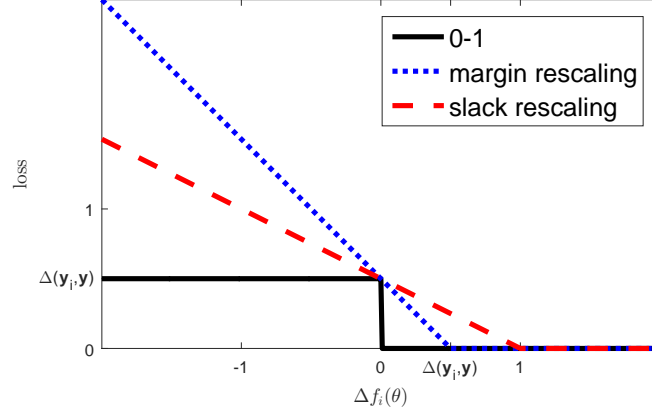


Figure 2: Hinge losses with slack rescaling method (red) and margin rescaling method (blue).

where $\Delta\phi_i(\mathbf{y}) = [\phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y})]$. The margin rescaling method's Lagrange dual problem becomes:

$$\max_{\alpha} \sum_{i=1}^N \sum_{\mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i} \alpha_{i\mathbf{y}} \Delta(\mathbf{y}_i, \mathbf{y}) - \frac{1}{2\lambda} \sum_{i=1}^N \sum_{\substack{\mathbf{y} \in \mathbf{Y} \setminus \mathbf{y}_i \\ \mathbf{y}' \in \mathbf{Y} \setminus \mathbf{y}_j}} \alpha_{i\mathbf{y}} \alpha_{j\mathbf{y}'} \Delta\phi_i(\mathbf{y}) \Delta\phi_j(\mathbf{y}') \quad (2.45)$$

subject to: $\forall i, \mathbf{y} : 0 \leq \alpha_{i\mathbf{y}} \leq 1$.

Due to the number of optimization variables $\alpha_{i\mathbf{y}}$ grows exponentially in size of \mathbf{y} , it is impossible to solve these two quadratic programmings above directly. In (21), the authors proposed a *cutting plan algorithm* that iteratively solves quadratic programs on a subset of \mathbf{Y} , which contains the constraint most violated \mathbf{y} that is added during each iteration. The algorithm stops

when no new violation can be found. The \mathbf{Y} is selected to maximizes hinge loss (Equation 2.40 for slack rescaling, Equation 2.42 for margin rescaling.)

At the test time, given a new instance \mathbf{x} , the predicted \mathbf{y} is given by:

$$\mathbf{y} = \arg \max_{\mathbf{y} \in \mathbf{Y}} f(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}^*) = \arg \max_{\mathbf{y} \in \mathbf{Y}} \boldsymbol{\theta}^{*\top} \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}). \quad (2.46)$$

2.6 Two-player Zero-sum Game

2.6.1 Definition

A two-player zero-sum game is a game played between two players, in which one player \hat{Y} wins what the other player \check{Y} loses. According to the Minimax Theorem (25), for each two-player zero-sum game, there exists an equilibrium with value v and a mixed strategy for each player, such that \hat{Y} 's best gain from the game is v , while \check{Y} 's gain is $-v$.

For example, for the game payoff matrix in Table I, the *mixed strategy* (i.e., probability distribution of actions) of player \check{Y} is $P(\check{\mathbf{y}}) = [\frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3}]$. The mixed strategy of player \hat{Y} is $P(\hat{\mathbf{y}}) = [\frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3}]$. The game value $v = -\frac{2}{15}$.

	\check{y}_1	\check{y}_2	\check{y}_3
\hat{y}_1	0.2	-0.3	-0.3
\hat{y}_2	-0.3	0.2	-0.3
\hat{y}_3	-0.3	-0.3	0.2

TABLE I: The game payoff matrix between player \check{Y} and player \hat{Y} . Each player has three actions, where \check{y}_i is an action that player \check{Y} plays, and \hat{y}_i is an action that player \hat{Y} plays.

For the game payoff matrix in Table II, the mixed strategy of player \check{Y} is $P(\check{\mathbf{y}}) = [\frac{1}{3} \quad \frac{2}{9} \quad \frac{2}{9} \quad \frac{2}{9}]$, and the mixed strategy of player \hat{Y} is $P(\hat{\mathbf{y}}) = [\frac{1}{3} \quad \frac{2}{3}]$. The game value $v = \frac{2}{3}$.

	\check{y}_1	\check{y}_2	\check{y}_3	\check{y}_4
\hat{y}_1	0	1	1	1
\hat{y}_2	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

TABLE II: The game payoff matrix between player \check{Y} and player \hat{Y} , where player \check{Y} has four actions, while player \hat{Y} has two.

2.6.2 Zero-sum Game Solving

The two-player zero-sum game can be solved using a pair of linear programs that have a constraint for each pure action (set of variable assignments) in the $m \times n$ game matrix (31; 32):

$$\begin{aligned}
 & \max_{v, p(\hat{\mathbf{y}}) \geq 0} v & (2.47) \\
 \text{subject to: } & v \leq \sum_{\hat{\mathbf{y}} \in \hat{\mathcal{Y}}} p(\hat{\mathbf{y}}) c_{\hat{\mathbf{y}}, 1} \\
 & \vdots \\
 & v \leq \sum_{\hat{\mathbf{y}} \in \hat{\mathcal{Y}}} p(\hat{\mathbf{y}}) c_{\hat{\mathbf{y}}, n} \\
 & \sum_{\hat{\mathbf{y}} \in \hat{\mathcal{Y}}} p(\hat{\mathbf{y}}) = 1;
 \end{aligned}$$

$$\begin{aligned}
 & \min_{v, p(\check{\mathbf{y}}) \geq 0} v & (2.48) \\
 \text{subject to: } & v \geq \sum_{\check{\mathbf{y}} \in \check{\mathcal{Y}}} p(\check{\mathbf{y}}) c_{1, \check{\mathbf{y}}} \\
 & \vdots \\
 \text{subject to: } & v \geq \sum_{\check{\mathbf{y}} \in \check{\mathcal{Y}}} p(\check{\mathbf{y}}) c_{m, \check{\mathbf{y}}} \\
 & \sum_{\check{\mathbf{y}} \in \check{\mathcal{Y}}} p(\check{\mathbf{y}}) = 1,
 \end{aligned}$$

where $c_{i,j}$ is each element of the game payoff matrix, v is the value of the game, $\hat{\mathcal{Y}}$ and $\check{\mathcal{Y}}$ are the pure actions of player \hat{Y} and player \check{Y} respectively.

When the value of the game is positive (i.e., $v > 0$), above linear programs can be rewritten to make the computation simpler. For Equation 2.47, we define $x_i = p(\hat{\mathbf{y}})/v$, then from $\sum_{\hat{\mathbf{y}} \in \hat{\mathcal{Y}}} p(\hat{\mathbf{y}}) = 1$, we have $\sum_{i=1}^m x_i = 1/v$. The original problem of maximizing v can be transformed to minimizing $\sum_{i=1}^m x_i$:

$$\begin{aligned}
 & \min_{x_i \geq 0} \sum_{i=1}^m x_i \\
 \text{subject to: } & 1 \leq \sum_{i=1}^m x_i c_{i,1} \\
 & \vdots \\
 & 1 \leq \sum_{i=1}^m x_i c_{i,n}
 \end{aligned} \tag{2.49}$$

Then game value $v = 1/\sum_{i=1}^m x_i$, and $p(\hat{\mathbf{y}}) = x_i v$. To guarantee the game value $v > 0$, we can compensate each element in the game matrix by setting $c_{i,j} = c_{i,j} - \min(c_{i,j}) + 1$, and remove the compensation from the solved game value, i.e., $v = v + \min(c_{i,j}) - 1$.

2.7 Fisher Consistency

Fisher consistency is a desired attribute of a classifier $\hat{f}(\mathbf{x})$ with surrogate loss function that when trained using the true distribution $P(\mathbf{y}, \mathbf{x})$ and an arbitrarily rich feature potential representation $\Psi(\mathbf{x}, \mathbf{y})$, predictions minimizing the multivariate loss are guaranteed, i.e., the classifier minimizes the expected loss under the true distribution, $\mathbb{E}_{P(\mathbf{y}, \mathbf{x})}[\text{loss}(\hat{f}(\mathbf{X}), \mathbf{Y})]$ (22; 33). It requires

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} \psi(\mathbf{x}, \mathbf{y}) \subseteq \arg \max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}), \quad (2.50)$$

which means the predicted $\hat{\mathbf{y}}$ by $\hat{f}(\mathbf{x})$ is always the most probable ones under the true distribution $P(\mathbf{y}, \mathbf{x})$. Logistic regression and CRF models that using logarithmic loss are known to be consistent for 0-1 loss/Hamming loss. Binary SVMs which using hinge-loss are Fisher consistent for 0-1 loss/Hamming loss also, but multi-class (i.e., non-binary) SVM and multi-class structured SVM (SSVM) are not (34; 35). One simple example is, suppose $\mathbf{Y} = \{y_1, y_2, y_3\}$, $P(y_1|\mathbf{x}) = 0.4, P(y_2|\mathbf{x}) = 0.4, P(y_3|\mathbf{x}) = 0.2$, then, $\arg \max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \{y_1, y_2\}$. For SSVM, from Equation 2.46, as long as prediction $\hat{\mathbf{y}} = \{y_1, y_2, y_3\}$ has the largest feature potential value, $\hat{\mathbf{y}} \subsetneq \{y_1, y_2\}$, i.e., not Fisher consistent.

CHAPTER 3

MULTIVARIATE PREDICTION GAME

(This chapter contains and expands on materials originally presented in Wang, H., Xing, W., Asif, K., and Ziebart, B.: Adversarial prediction games for multivariate losses. In *Advances in Neural Information Processing Systems*, pages 2728-2736, 2015.)

Multivariate loss functions are used to assess performance in many modern prediction tasks, including Information Retrieval and Natural Language Processing applications. Convex approximations are typically optimized in their place to avoid NP-hard empirical risk minimization problems. In this chapter, we propose a framework to approximate the training data instead of the loss function by posing multivariate prediction as an adversarial game between a performance-maximizing (i.e., loss-minimizing) prediction player and a performance-minimizing (i.e., loss-maximizing) evaluation player constrained to match specified properties of training data. This avoids the non-convexity of empirical risk minimization, but game sizes are exponential in the number of predicted variables. We overcome this intractability using the double oracle constraint generation method.

3.1 Introduction

For many problems in Information Retrieval (IR) and Natural Language Processing (NLP), the performance of a predictor is evaluated based on the combination of predictions it makes for multiple variables. Examples include the precision when limited to k positive predictions ($P@k$), the harmonic mean of precision and recall (F-score), the discounted cumulative gain (DCG) for assessing ranking quality, the alignment error rate (AER) for evaluating the quality of machine translation. These stand in contrast to measures like the accuracy and (log) likelihood, which are additive over independently predicted variables. Many multivariate performance measures are not concave functions of predictor parameters, so maximizing them over empirical training data (or, equivalently, empirical risk minimization over a corresponding non-convex multivariate loss function) is computationally intractable (3) and can only be accomplished approximately using local optimization methods (36). Instead, convex surrogates for the empirical risk are optimized using either an additive (4; 5; 6) or a multivariate approximation (7; 8) of the loss function. Though the ability to incorporate multivariate loss functions is attractive, the hinge loss approximation leads to theoretical shortcomings, including a lack of Fisher consistency (22), meaning even if trained using the true distribution $P(\mathbf{y}, \mathbf{x})$ and an arbitrarily rich feature representation Φ , predictions minimizing the multivariate loss are not guaranteed. For both types of approximations, the gap between the application performance measure and the surrogate loss measure can lead to substantial sub-optimality of the resulting predictions (9).

Rather than optimizing an approximation of the multivariate loss for available training data, we take an alternate approach (37; 25; 2) that robustly minimizes the exact multivariate loss function using approximations of the training data. We formalize this using a zero-sum game between a predictor player and an adversarial evaluator player. Learned weights parameterize this game’s payoffs and enable generalization from training data to new predictive settings. The key computational challenge this approach poses is that the size of multivariate prediction games grows exponentially in the number of variables. We leverage constraint generation methods developed for solving large zero-sum games (38) and efficient methods for computing best responses (39) to tame this complexity. In many cases, the structure of the multivariate loss function enables the zero-sum game’s Nash equilibrium to be efficiently computed. We formulate parameter estimation as a convex optimization problem and solve it using standard convex optimization methods. We present *Adversarial Multivariate Prediction Games (MPG)* framework in the following sections of this chapter, and will demonstrate the benefits from this approach on prediction tasks in IR and NLP areas in the next chapter.

3.2 Architecture

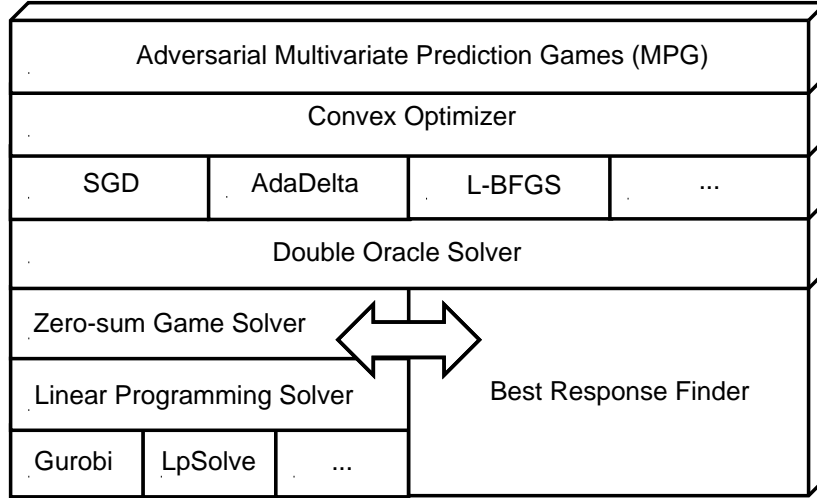


Figure 3: The architecture of Multivariate Prediction Game (MPG) framework.

The architecture of Adversarial Multivariate Prediction Game (MPG) framework ¹ is presented in Figure 3. It has several main components. We briefly introduce each of the components in this section, and give detailed discussions in the next few sections:

- *Convex Optimizer*: it encapsulates convex optimization algorithms since the framework models the learning procedure as solving a convex optimization problem. We discuss this modeling in Section 3.3. Multiple off-the-shelf implementations of numerical optimiza-

¹Code is available in https://github.com/hwang207/mpg_java

tion algorithms can be easily plugged in, including stochastic gradient descent (SGD), AdaDelta¹ (40), L-BFGS², etc.

- *Zero-sum Game Solver*: because MPG formulates a zero-sum game between two players: one is the performance-maximizer, the other is the performance-minimizer, to find the solution of a game (i.e., the equilibrium), the game payoff matrix needs to be solved. Recall we discussed in Section 2.6.2 that a zero-sum game can be solved by a *Linear Programming Solver*, which in turn solves Equation 2.47 and Equation 2.49. We build the game solver that encapsulates these logics above. Different LP solver implementations are available online: either open-sourced (e.g., LpSolve (41)), or commercial but academic free (e.g., Gurobi (42)).
- *Double Oracle Solver*: as we will see in the examples in Section 3.4, in real word problems, the prediction tasks usually have exponential-size game payoff matrices that can't be solved by brute-force. A double oracle constraint generation game solver is implemented to solve the game iteratively as described in Section 3.5. During each iteration, a game payoff matrix is solved by the *Zero-sum Game Solver*.
- *Best Response Finder*: the main idea of double oracle constraint generation algorithm (Algorithm 1) is iteratively solving a game payoff matrix by adding the best response to the opponent's current strategy. Depending on what the predicting structure is, and

¹Implementation at <https://github.com/mgormley/optimize>

²Implementation at <https://github.com/robert-dodier/riso>

what the evaluation metrics of interest are, the algorithms need to be designed carefully to guarantee their efficiencies. We give a general discussion on this topic in Section 3.7, and describe more detailed best response finding algorithms in Chapter 4.

3.3 Modeling

Following a recent adversarial formulation for classification (2), we view multivariate prediction as a two-player game between player $\hat{\mathbf{Y}}$ making predictions and player $\check{\mathbf{Y}}$ determining the evaluation distribution. Player $\hat{\mathbf{Y}}$ first stochastically chooses a predictive distribution of variable assignments, $\hat{P}(\hat{\mathbf{y}}|\mathbf{x})$, to maximize a multivariate performance measure, then player $\check{\mathbf{Y}}$ stochastically chooses an evaluation distribution, $\check{P}(\check{\mathbf{y}}|\mathbf{x})$, that minimizes the performance measure. Further, player $\check{\mathbf{Y}}$ must choose the relevant items in a way that (approximately) matches in expectation with a set of statistics, $\Phi(\mathbf{x}, \mathbf{y})$, measured from labeled training data. We denote this set as Ξ .

Definition 1. *The **multivariate prediction game (MPG)** for n predicted variables is:*

$$\max_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \min_{\check{P}(\check{\mathbf{y}}|\mathbf{x}) \in \Xi} \mathbb{E}_{\tilde{P}(\mathbf{x}) \hat{P}(\hat{\mathbf{y}}|\mathbf{x}) \check{P}(\check{\mathbf{y}}|\mathbf{x})} \left[\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) \right], \quad (3.1)$$

where $\hat{P}(\hat{\mathbf{y}}|\mathbf{x})$ and $\check{P}(\check{\mathbf{y}}|\mathbf{x})$ are distributions over combinations of labels for the n predicted variables and the set Ξ corresponds to the constraint: $\mathbb{E}_{\tilde{P}(\mathbf{x}) \hat{P}(\hat{\mathbf{y}}|\mathbf{x})} [\Phi(\mathbf{X}, \check{\mathbf{Y}})] = \mathbb{E}_{\tilde{P}(\mathbf{y}, \mathbf{x})} [\Phi(\mathbf{X}, \mathbf{Y})]$.

Since the set Ξ constrains the adversary's multivariate label distribution over the entire distribution of inputs $\tilde{P}(\mathbf{x})$, solving this game directly is impractical when the number of training examples is large. Instead, we employ the method of Lagrange multipliers in Theorem 1, which allows the set of games to be independently solved given Lagrange multipliers $\boldsymbol{\theta}$.

Theorem 1. *The multivariate prediction game's value (Definition 1) can be equivalently obtained by solving a set of unconstrained maximin games parameterized by Lagrange multipliers θ :*

$$\begin{aligned}
& \max_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \min_{\check{P}(\check{\mathbf{y}}|\mathbf{x}) \in \Xi} \mathbb{E}_{\tilde{P}(\mathbf{x}) \hat{P}(\hat{\mathbf{y}}|\mathbf{x}) \check{P}(\check{\mathbf{y}}|\mathbf{x})} \left[\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) \right] \\
&= \max_{\theta} \left(\mathbb{E}_{\tilde{P}(\mathbf{y}, \mathbf{x})} [\boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \mathbf{Y})] + \sum_{\mathbf{x} \in \mathcal{X}} \tilde{P}(\mathbf{x}) \min_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \max_{\check{P}(\check{\mathbf{y}}|\mathbf{x})} \mathbb{E}_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x}) \check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\underbrace{\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) - \boldsymbol{\theta} \cdot \Phi(\mathbf{x}, \check{\mathbf{Y}})}_{\mathbf{S}'_{\hat{\mathbf{y}}, \check{\mathbf{y}}}} \right) \right), \tag{3.2}
\end{aligned}$$

where: $\Phi(\mathbf{x}, \mathbf{y})$ is a vector of features characterizing the set of prediction variables $\{y_i\}$ and provided contextual variables $\{x_i\}$ each related to predicted variable y_i .

The resulting game's payoff matrix can be expressed as the original game scores of Equation 3.1 augmented with Lagrangian potentials. The combination defines a new payoff matrix with entries $\mathbf{S}'_{\hat{\mathbf{y}}, \check{\mathbf{y}}} = \text{score}(\hat{\mathbf{y}}, \check{\mathbf{y}}) - \boldsymbol{\theta} \cdot \Phi(\mathbf{x}, \check{\mathbf{y}})$, as shown in Equation 3.2. The Fisher consistency of adversarial MPG model is guaranteed (34).

Proof of Theorem 1.

$$\begin{aligned}
& \max_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \min_{\check{P}(\check{\mathbf{y}}|\mathbf{x}) \in \Xi} \mathbb{E}_{\hat{P}(\mathbf{x})\hat{P}(\hat{\mathbf{y}}|\mathbf{x})\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) \right) \\
& \stackrel{(a)}{=} \min_{\check{P}(\check{\mathbf{y}}|\mathbf{x}) \in \Xi} \max_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \mathbb{E}_{\hat{P}(\mathbf{x})\hat{P}(\hat{\mathbf{y}}|\mathbf{x})\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) \right) \\
& \stackrel{(b)}{=} \min_{\check{P}(\check{\mathbf{y}}|\mathbf{x}) \in \Xi} \mathbb{E}_{\hat{P}(\mathbf{x})} \left[\max_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \mathbb{E}_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) \right) \right] \\
& \stackrel{(c)}{=} \max_{\boldsymbol{\theta}} \min_{\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\mathbb{E}_{\hat{P}(\mathbf{x})} \left[\max_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \mathbb{E}_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) \right) \right] + \mathbb{E}_{\hat{P}(\mathbf{y}, \mathbf{x})} [\boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \mathbf{Y})] - \mathbb{E}_{\hat{P}(\mathbf{x})\check{P}(\check{\mathbf{y}}|\mathbf{x})} [\boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \check{\mathbf{Y}})] \right) \\
& \stackrel{(d)}{=} \max_{\boldsymbol{\theta}} \left(\mathbb{E}_{\hat{P}(\mathbf{y}, \mathbf{x})} [\boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \mathbf{Y})] + \min_{\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\mathbb{E}_{\hat{P}(\mathbf{x})} \left[\max_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \mathbb{E}_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) \right) - \mathbb{E}_{\check{P}(\check{\mathbf{y}}|\mathbf{x})} [\boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \check{\mathbf{Y}})] \right] \right) \right) \\
& \stackrel{(e)}{=} \max_{\boldsymbol{\theta}} \left(\mathbb{E}_{\hat{P}(\mathbf{y}, \mathbf{x})} [\boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \mathbf{Y})] + \mathbb{E}_{\hat{P}(\mathbf{x})} \left(\min_{\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left[\max_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \mathbb{E}_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) \right) - \mathbb{E}_{\check{P}(\check{\mathbf{y}}|\mathbf{x})} [\boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \check{\mathbf{Y}})] \right] \right) \right) \\
& \stackrel{(f)}{=} \max_{\boldsymbol{\theta}} \left(\mathbb{E}_{\hat{P}(\mathbf{y}, \mathbf{x})} [\boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \mathbf{Y})] + \mathbb{E}_{\hat{P}(\mathbf{x})} \left[\min_{\check{P}(\check{\mathbf{y}}|\mathbf{x})} \max_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})} \mathbb{E}_{\hat{P}(\hat{\mathbf{y}}|\mathbf{x})\check{P}(\check{\mathbf{y}}|\mathbf{x})} \left(\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) - \boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \check{\mathbf{Y}}) \right) \right] \right)
\end{aligned} \tag{3.3}$$

Equality (a) is a consequence of duality in zero-sum games (31). Equality (b) swaps the maximization and the expectation since each maximization can be performed independently. Equality (c) is obtained by writing the Lagrangian and taking the dual, and strong Lagrangian duality is guaranteed when a feasible solution exists on the relative interior of the convex constraint set Ξ (27). (A small amount of slack corresponds to regularization of the $\boldsymbol{\theta}$ parameter in the dual and guarantees the strong duality feasibility requirement is satisfied in practice.) Equality (d) holds since $\mathbb{E}_{\hat{P}(\mathbf{y}, \mathbf{x})} [\boldsymbol{\theta} \cdot \Phi(\mathbf{X}, \mathbf{Y})]$ is constant to $\check{P}(\check{\mathbf{y}}|\mathbf{x})$, and is moved out of *min*. The part insides $\max_{\boldsymbol{\theta}}$ is concave because the internal *minimization* is concavity preserving. Same as Equality (b), Equality (e) swaps minimization and expectation due to the independence of minimizations. \square

3.4 Example Multivariate Prediction Games and Small-scale Solutions

In this section, we first review three multivariate metrics that are commonly used in Information Retrieval and Natural Language Processing tasks. Then, we demonstrate examples of Lagrangian payoff matrices for the P@2, F-score, and DCG games for three variables. The more detailed description of each metric can be found in Section 4.1.

3.4.1 Example Multivariate Metrics

For both Information Retrieval and Natural Language Processing, a vector of retrieved (or classified as relevant) items from the pool of n items can be represented as $\hat{\mathbf{y}} \in \{0, 1\}^n$ and a vector of true relevant items as $\mathbf{y} \in \{0, 1\}^n$ with $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ denoting side contextual information (e.g., search terms and document contents). *Precision* (P) and *recall* (R) are important measures for the target systems, and are defined as followings:

$$P(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\hat{\mathbf{y}} \cdot \mathbf{y}}{\|\hat{\mathbf{y}}\|_1} \quad \text{and} \quad (3.4)$$

$$R(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\hat{\mathbf{y}} \cdot \mathbf{y}}{\|\mathbf{y}\|_1}, \quad (3.5)$$

where $\|\cdot\|_1$ denotes the number of ‘1’s in the vector.

However, maximizing either leads to degenerate solutions (predict all to maximize recall, or predict none to maximize precision). One popular metric is *F-score*, which is the harmonic

mean of the precision and recall. Using this notation, the F-score for a set of items can be simply represented as:

$$F_1(\hat{\mathbf{y}}, \mathbf{y}) = \frac{2\hat{\mathbf{y}} \cdot \mathbf{y}}{\|\hat{\mathbf{y}}\|_1 + \|\mathbf{y}\|_1} \text{ and } F_1(\mathbf{0}, \mathbf{0}) = 1. \quad (3.6)$$

Another metric, the *precision at k* ($P@k$), limits the predictor to exactly k positive predictions,

$$P@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\hat{\mathbf{y}} \cdot \mathbf{y}}{k} \text{ where } \|\hat{\mathbf{y}}\|_1 = k. \quad (3.7)$$

For some other tasks, a ranked list of retrieved items is desired. This can be represented as a permutation, σ , where $\sigma(i)$ denotes the i^{th} -ranked item (and $\sigma^{-1}(j)$ denotes the rank of the j^{th} item). Evaluation metrics that emphasize the top-ranked items are used, e.g., to produce search engine results attuned to actual usage. The *discounted cumulative gain* (DCG) measures the performance of item rankings, from 1 to n , with k relevancy scores, $y_i \in \{0, \dots, k-1\}$ as:

$$DCG(\sigma, \mathbf{y}) = \sum_{i=1}^n \frac{2^{y_{\sigma(i)}} - 1}{\log_2(i+1)}. \quad (3.8)$$

Since the $\log_2(i+1)$ function is monotonically increasing, DCG measures the sum of the influence of all items that decreases according to the predicted ranking.

3.4.2 Example Game Payoff Matrices

In this subsection, we show three example game payoff matrices with Lagrange potentials for P@2, F-score, and DCG with three variables in Table III, Table IV, Table V respectively. We employ additive feature functions, $\Phi(\mathbf{x}, \tilde{\mathbf{y}}) = \sum_{i=1}^n \phi(x_i) I(\tilde{y}_i = 1)$, in these examples (with indicator function $I(\cdot)$). The Lagrangian potential terms for each game with potential variables are compactly represented as: $\psi_i \triangleq \boldsymbol{\theta} \cdot \boldsymbol{\phi}(X_i = x_i)$ when $\tilde{Y}_i = 1$ (and 0 otherwise).

TABLE III: The payoff matrix for the zero-sum game between player \tilde{Y} choosing columns and player \hat{Y} choosing rows with three variables for *precision at k*.

P@2	000	001	010	011	100	101	110	111
011	0	$\frac{1}{2} - \psi_3$	$\frac{1}{2} - \psi_2$	$1 - \psi_2 - \psi_3$	$0 - \psi_1$	$\frac{1}{2} - \psi_1 - \psi_3$	$\frac{1}{2} - \psi_1 - \psi_2$	$1 - \psi_1 - \psi_2 - \psi_3$
101	0	$\frac{1}{2} - \psi_3$	$0 - \psi_2$	$\frac{1}{2} - \psi_2 - \psi_3$	$\frac{1}{2} - \psi_1$	$1 - \psi_1 - \psi_3$	$\frac{1}{2} - \psi_1 - \psi_2$	$1 - \psi_1 - \psi_2 - \psi_3$

Zero-sum games such as these can be solved using a pair of linear programs that we discussed in Section 2.6.2, where $\mathbf{S}'_{\tilde{\mathbf{y}}, \hat{\mathbf{y}}}$ is the Lagrangian-augmented payoff and v is the value of the game. The second player to act in a zero-sum game can maximize/minimize using a pure strategy (i.e., a single value assignment to all variables). Thus, these LPs consider only the set of pure strategies of the opponent to find the first player's mixed equilibrium strategy. The equilibrium strategy for the predictor is a distribution over rows and the equilibrium strategy for the adversary is a distribution over columns.

TABLE IV: The payoff matrix for the zero-sum game between player \tilde{Y} choosing columns and player \hat{Y} choosing rows with three variables for F_1 .

F_1	000	001	010	011	100	101	110	111
000	1	$0-\psi_3$	$0-\psi_2$	$0-\psi_2-\psi_3$	$0-\psi_1$	$0-\psi_1-\psi_3$	$0-\psi_1-\psi_2$	$0-\psi_1-\psi_2-\psi_3$
001	0	$1-\psi_3$	$0-\psi_2$	$\frac{2}{3}-\psi_2-\psi_3$	$0-\psi_1$	$\frac{2}{3}-\psi_1-\psi_3$	$0-\psi_1-\psi_2$	$\frac{1}{2}-\psi_1-\psi_2-\psi_3$
010	0	$0-\psi_3$	$1-\psi_2$	$\frac{2}{3}-\psi_2-\psi_3$	$0-\psi_1$	$0-\psi_1-\psi_3$	$\frac{2}{3}-\psi_1-\psi_2$	$\frac{1}{2}-\psi_1-\psi_2-\psi_3$
011	0	$\frac{2}{3}-\psi_3$	$\frac{2}{3}-\psi_2$	$1-\psi_2-\psi_3$	$0-\psi_1$	$\frac{1}{2}-\psi_1-\psi_3$	$\frac{1}{2}-\psi_1-\psi_2$	$\frac{4}{5}-\psi_1-\psi_2-\psi_3$
100	0	$0-\psi_3$	$0-\psi_2$	$0-\psi_2-\psi_3$	$1-\psi_1$	$\frac{2}{3}-\psi_1-\psi_3$	$\frac{2}{3}-\psi_1-\psi_2$	$\frac{1}{2}-\psi_1-\psi_2-\psi_3$
101	0	$\frac{2}{3}-\psi_3$	$0-\psi_2$	$\frac{1}{2}-\psi_2-\psi_3$	$\frac{2}{3}-\psi_1$	$1-\psi_1-\psi_3$	$\frac{1}{2}-\psi_1-\psi_2$	$\frac{4}{5}-\psi_1-\psi_2-\psi_3$
110	0	$0-\psi_3$	$\frac{2}{3}-\psi_2$	$\frac{1}{2}-\psi_2-\psi_3$	$\frac{2}{3}-\psi_1$	$\frac{1}{2}-\psi_1-\psi_3$	$1-\psi_1-\psi_2$	$\frac{4}{5}-\psi_1-\psi_2-\psi_3$
111	0	$\frac{1}{2}-\psi_3$	$\frac{1}{2}-\psi_2$	$\frac{4}{5}-\psi_2-\psi_3$	$\frac{1}{2}-\psi_1$	$\frac{4}{5}-\psi_1-\psi_3$	$\frac{4}{5}-\psi_1-\psi_2$	$1-\psi_1-\psi_2-\psi_3$

The size of each game's payoff matrix grows exponentially with the number of variables, n : $(2^n)\binom{n}{k}$ for the *precision at k* game; $(2^n)^2$ for the F-score game; and $(n! k^n)$ for the DCG game with k possible relevance levels. These sizes make explicit construction of the game matrix impractical for all but the smallest of problems.

TABLE V: The payoff matrix for the zero-sum game between player \tilde{Y} choosing columns and player \hat{Y} choosing rows with three variables for DCG with binary relevance values $\tilde{y}_i \in \{0, 1\}$, accumulated at rank 3, and we let $\lg 3 \triangleq \log_2 3$.

DCG	000	001	010	011
123	0	$\frac{1}{\lg 4} - \psi_3$	$\frac{1}{\lg 3} - \psi_2$	$\frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_2 - \psi_3$
132	0	$\frac{1}{\lg 3} - \psi_3$	$\frac{1}{\lg 4} - \psi_2$	$\frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_2 - \psi_3$
213	0	$\frac{1}{\lg 4} - \psi_3$	$\frac{1}{\lg 2} - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 4} - \psi_2 - \psi_3$
231	0	$\frac{1}{\lg 3} - \psi_3$	$\frac{1}{\lg 2} - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} - \psi_2 - \psi_3$
312	0	$\frac{1}{\lg 2} - \psi_3$	$\frac{1}{\lg 4} - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 4} - \psi_2 - \psi_3$
321	0	$\frac{1}{\lg 2} - \psi_3$	$\frac{1}{\lg 3} - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} - \psi_2 - \psi_3$
	100	101	110	111
123	$\frac{1}{\lg 2} - \psi_1$	$\frac{1}{\lg 2} + \frac{1}{\lg 4} - \psi_1 - \psi_3$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} - \psi_1 - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_2 - \psi_3$
132	$\frac{1}{\lg 2} - \psi_1$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} - \psi_1 - \psi_3$	$\frac{1}{\lg 2} + \frac{1}{\lg 4} - \psi_1 - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_2 - \psi_3$
213	$\frac{1}{\lg 3} - \psi_1$	$\frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_3$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} - \psi_1 - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_2 - \psi_3$
231	$\frac{1}{\lg 4} - \psi_1$	$\frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_3$	$\frac{1}{\lg 2} + \frac{1}{\lg 4} - \psi_1 - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_2 - \psi_3$
312	$\frac{1}{\lg 3} - \psi_1$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} - \psi_1 - \psi_3$	$\frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_2 - \psi_3$
321	$\frac{1}{\lg 4} - \psi_1$	$\frac{1}{\lg 2} + \frac{1}{\lg 4} - \psi_1 - \psi_3$	$\frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_2$	$\frac{1}{\lg 2} + \frac{1}{\lg 3} + \frac{1}{\lg 4} - \psi_1 - \psi_2 - \psi_3$

3.5 Large-scale Strategy Inference

More efficient methods for obtaining Nash equilibria are needed to scale the multivariate prediction game approach to large prediction tasks with exponentially-sized payoff matrices. Though much attention has focused on efficiently computing ϵ -Nash equilibria (e.g., in $\mathcal{O}(1/\epsilon)$ time or $\mathcal{O}(\ln(1/\epsilon))$ time (43)), which guarantee each player a payoff within ϵ of optimal, we employ the *double oracle algorithm* for iteratively finding an exact equilibrium that works well in practice, and guarantees to converge to a minimax equilibrium (38).

Algorithm 1 Double oracle constraint generation game solver

```

1: procedure DOUBLEORACLESOLVER( $\psi$ , any initial sequence sets  $\hat{Y}_0$  and  $\check{Y}_0$ )
2:   Initialize Player  $\hat{Y}$ 's sequence set  $\hat{Y} = \hat{Y}_0$ 
3:   Initialize Player  $\check{Y}$ 's sequence set  $\check{Y} = \check{Y}_0$ 
4:    $\mathbf{S}'_{\hat{\mathbf{y}}, \check{\mathbf{y}}} = \text{computeGameMatrix}(\hat{Y}, \check{Y}, \psi)$  ▷ Equation 3.2
5:   repeat
6:      $[P(\hat{\mathbf{y}}|\mathbf{x}), v_{\text{Nash}_1}] = \text{zeroSumGameSolver}_{\hat{Y}}(\mathbf{S}'_{\hat{\mathbf{y}}, \check{\mathbf{y}}})$  ▷ LP of Equation 2.47
7:      $[\check{\mathbf{y}}^*, \check{v}_{\text{BR}}] = \text{bestResponseFinder}(P(\hat{\mathbf{y}}|\mathbf{x}), \psi)$ 
8:     if ( $v_{\text{Nash}_1} \neq \check{v}_{\text{BR}}$ ) then ▷ Check if best response provides improvement
9:        $\check{Y} = \check{Y} \cup \check{\mathbf{y}}^*$ 
10:       $\mathbf{S}'_{\hat{\mathbf{y}}, \check{\mathbf{y}}} = \text{computeGameMatrix}(\hat{Y}, \check{Y}, \psi)$  ▷ Add new row to game matrix
11:    end if
12:     $[P(\check{\mathbf{y}}|\mathbf{x}), v_{\text{Nash}_2}] = \text{zeroSumGameSolver}_{\check{Y}}(\mathbf{S}'_{\hat{\mathbf{y}}, \check{\mathbf{y}}})$  ▷ LP of Equation 2.49
13:     $[\hat{\mathbf{y}}^*, \hat{v}_{\text{BR}}] = \text{bestResponseFinder}(P(\check{\mathbf{y}}|\mathbf{x}), \psi)$ 
14:    if ( $v_{\text{Nash}_2} \neq \hat{v}_{\text{BR}}$ ) then
15:       $\hat{Y} = \hat{Y} \cup \hat{\mathbf{y}}^*$ 
16:       $\mathbf{S}'_{\hat{\mathbf{y}}, \check{\mathbf{y}}} = \text{computeGameMatrix}(\hat{Y}, \check{Y}, \psi)$  ▷ Add new column to game matrix
17:    end if
18:  until ( $v_{\text{Nash}_1} = v_{\text{Nash}_2} = \hat{v}_{\text{BR}} = \check{v}_{\text{BR}}$ ) ▷ Stop if no more improvement
19:  return [ $v_{\text{Nash}}$ ,  $P(\hat{\mathbf{y}}|\mathbf{x})$ ,  $P(\check{\mathbf{y}}|\mathbf{x})$ ]
20: end procedure

```

Neither player can improve upon their strategy with additional actions when Algorithm 1 terminates, thus the strategies it returns are a Nash equilibrium pair (38). The algorithm is efficient in practice so long as each player's strategy is compact (i.e., the number of actions with non-zero probability is a polynomial subset of the label combinations) and best responses to opponents' strategies can be obtained efficiently (i.e., in polynomial time) for each player.

Approximate solutions can be obtained from the double oracle method to improve computational efficiency in two ways: by limiting the maximum number of best responses that are

added by the algorithm; and/or by allowing some tolerance ϵ when checking for game value and best response equality (i.e., $|v_{\text{Nash}_1} - \check{v}_{\text{BR}}| \geq \epsilon$ in place of $(v_{\text{Nash}_1} \neq \check{v}_{\text{BR}})$).

The double oracle game solver is a central component of the MPG approach. Its returned equilibrium distributions are used to compute the gradients needed for learning the model parameters, θ . The zero-sum game solver used as a sub-routine of the double oracle method is implemented using linear programming, which we discussed in Section 2.6.2.

3.6 Parameter Estimation

Predictive model parameters, θ , must be chosen to ensure that the adversarial distribution is similar to training data. Though adversarial prediction can be posed as a convex optimization problem (2), the objective function is not smooth. General subgradient methods require $\mathcal{O}(1/\epsilon^2)$ iterations to provide an ϵ approximation to the optima. Other method like L-BFGS (44) can also be employed, which has been empirically shown to converge at a faster rate in many cases despite lacking theoretical guarantees for non-smooth objectives (45). Recall in Section 3.3, we showed the objective function the MPG in Equation 3.2. We also employ ℓ_2 regularization, $-\frac{\lambda}{2}||\theta||^2$, to avoid overfitting to the training data sample. The addition of the smooth regularizer often helps to improve the rate of convergence:

$$\max_{\theta} \left(\mathbb{E}_{\tilde{P}(\mathbf{y}, \mathbf{x})} [\theta \cdot \Phi(\mathbf{X}, \mathbf{Y})] + \sum_{\mathbf{x} \in \mathcal{X}} \tilde{P}(\mathbf{x}) \min_{\tilde{P}(\tilde{\mathbf{y}}|\mathbf{x})} \max_{\tilde{P}(\tilde{\mathbf{y}}|\mathbf{x})} \mathbb{E}_{\tilde{P}(\tilde{\mathbf{y}}|\mathbf{x})} \left(\underbrace{\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{Y}}) - \theta \cdot \Phi(\mathbf{x}, \check{\mathbf{Y}})}_{\mathbf{s}'_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}} \right) - \frac{\lambda}{2} ||\theta||^2 \right), \quad (3.9)$$

where λ is the regularization parameter that controls the importance of the regularization value.

The gradient in these optimizations in Equation 3.9 for training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ is the difference between feature moments with additional regularization term:

$$\frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left(\Phi(\mathbf{x}_i, \mathbf{y}_i) - \sum_{\tilde{\mathbf{y}} \in \mathcal{Y}} \tilde{P}(\tilde{\mathbf{y}}|\mathbf{x}_i) \Phi(\mathbf{x}_i, \tilde{\mathbf{y}}) \right) - \lambda \theta. \quad (3.10)$$

The adversarial strategies $\tilde{P}(\tilde{\mathbf{y}}|\mathbf{x}_i)$ needed for calculating this gradient are parts of the return value of Algorithm 1.

3.7 Finding The Best Response

In Figure 3, we can see that on the bottom right, there is a critical component: *Best Response Finder*. It interacts with the *Zero-sum Game Solver* during each iteration of the double oracle algorithm (i.e., line 7, and 13 in Algorithm 1). It finds a solution $\hat{\mathbf{y}}^*$ for the maximizer $\hat{\mathbf{Y}}$ and a solution $\check{\mathbf{y}}^*$ for the minimizer $\check{\mathbf{Y}}$ as followings:

$$\begin{aligned}\hat{\mathbf{y}}^* &= \arg \max_{\hat{\mathbf{y}}} \mathbb{E}_{P(\check{\mathbf{y}}|\mathbf{x})} [\text{score}(\hat{\mathbf{y}}, \check{\mathbf{Y}}) - \psi(\check{\mathbf{y}})] \\ &\stackrel{(a)}{=} \arg \max_{\hat{\mathbf{y}}} \mathbb{E}_{P(\check{\mathbf{y}}|\mathbf{x})} [\text{score}(\hat{\mathbf{y}}, \check{\mathbf{Y}})],\end{aligned}\tag{3.11}$$

$$\check{\mathbf{y}}^* = \arg \min_{\check{\mathbf{y}}} \mathbb{E}_{P(\hat{\mathbf{y}}|\mathbf{x})} [\text{score}(\hat{\mathbf{Y}}, \check{\mathbf{y}}) - \psi(\check{\mathbf{y}})],\tag{3.12}$$

where the Lagrangian potential terms $\psi(\check{\mathbf{y}}) \triangleq \boldsymbol{\theta} \cdot \Phi(\mathbf{x}, \check{\mathbf{y}})$. Equality (a) holds due to the Lagrange potentials $\psi(\check{\mathbf{y}})$ are invariant to the choice of $\hat{\mathbf{y}}^*$.

Usually, efficiently finding the best responses is difficult. The difficulty depends on both the *score* function being considered and the constraints imposed on the adversary (i.e., the Lagrange potentials). An exact and efficient algorithm for F-score maximization (given a data distribution) has only been developed relatively recently (39) using dynamic programming techniques. We further develop from that technique to efficiently solve these best response problems for:

- F-score for binary classification;
- F-score for multi-class sequence prediction with linear-chain constraints;

- Precision at k ($P@k$), and discounted cumulative gain (DCG) for rankings;
- Alignment error rate (AER) for word alignment quality evaluation in machine translation;
- Hamming loss for syntactic context-free grammar (CFG) parsing.

All of these are discussed in the following Chapter 4.

CHAPTER 4

METRICS, BEST RESPONSE ALGORITHMS

(This chapter contains and expands on materials originally presented in Wang, H., Xing, W., Asif, K., and Ziebart, B.: Adversarial prediction games for multivariate losses. In *Advances in Neural Information Processing Systems*, pages 2728-2736, 2015.)

In this chapter, we first review several commonly used metrics in Information Retrieval (IR) and Natural Language Processing (NLP) areas that our Adversarial Multivariate Prediction Games (MPG) framework directly optimizes. As we discussed in the previous chapter, in order to apply MPG to those metrics, we need efficient algorithms that find the best responses to the opponent’s strategies. We propose the corresponding algorithms for finding the best response for each metric efficiently, after reviewing the evaluation metrics.

4.1 Metrics in Information Retrieval and Natural Language Processing

In this section, we review several commonly used metrics in Information Retrieval (IR) and Natural Language Processing (NLP) areas. These metrics are shown can be directly optimized by MPG framework, which overcomes the mismatches between the loss surrogates in traditional empirical risk minimization methods and the metrics themselves for evaluation.

4.1.1 Hamming Loss, Accuracy

Hamming loss, or 0-1 loss, measures the number of incorrectly predicted variables:

$$hammingLoss(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i=1}^N I(\hat{y}_i \neq y_i), \quad (4.1)$$

where $\hat{\mathbf{y}}$ and \mathbf{y} are two vectors of variables that denote prediction and gold standard (actual) of N variables respectively, $I(\cdot)$ is the indicator function.

Hamming loss is closely related to *accuracy*, which is number of correct predictions divided by the union of predicted and gold standard variables:

$$accuracy(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\sum_{i=1}^N I(\hat{y}_i = y_i)}{|\hat{\mathbf{y}} \cup \mathbf{y}|}. \quad (4.2)$$

Given the confusion matrix of a predictor in Table VI, which represents the information about the actual and predicted results by a predictor on a specified dataset, *accuracy* can be written as:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}. \quad (4.3)$$

TABLE VI: Confusion matrix of a predictor.

		predicted	
		positive	negative
actual	positive	TP	FN
	negative	FP	TN

4.1.2 F-score

F-score, or F-measure, is a very widely used binary evaluation metric in both IR and NLP.

Referring to the confusion matrix in Table VI, *precision* and *recall* are defined as follows:

$$precision = \frac{TP}{TP + FP}, \quad (4.4)$$

$$recall = \frac{TP}{TP + FN}. \quad (4.5)$$

Precision measures what is the proportion of predicted positive items that are actually true positive, while *recall* measures what is the proportion of true positive items that are correctly predicted as positive. When predicting all the examples as positive, $recall = 1$, $precision = 0$; and when predicting all examples as negative, $precision = 1$, $recall = 0$. F-score is defined to balance the precision and recall:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \quad (4.6)$$

$$= \frac{(1 + \beta^2) \cdot TP}{(1 + \beta^2) \cdot TP + \beta^2 \cdot FP + FN}, \quad (4.7)$$

where $\beta > 0$. When $\beta = 1$, F-score is the harmonic mean of the precision and recall, is also called F_1 score:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (4.8)$$

In this thesis, we use ‘F-score’ and ‘ F_1 ’ interchangeably, and all the algorithms we propose for F_1 can be extended to other β values.

Like we showed in Section 3.4.1, we can write *precision*, *recall*, and F_1 in the forms that are represented by vectors. For example, let a vector of predicted as positive items from the pool of n items be represented as $\hat{\mathbf{y}} \in \{0, 1\}^n$, and a vector of actual items as $\mathbf{y} \in \{0, 1\}^n$, where ‘1’ means positive, and ‘0’ means negative. Equation 4.4 and Equation 4.5 are rewritten as:

$$\text{precision}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\hat{\mathbf{y}} \cdot \mathbf{y}}{\|\hat{\mathbf{y}}\|_1}, \quad (4.9)$$

$$\text{recall}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\hat{\mathbf{y}} \cdot \mathbf{y}}{\|\mathbf{y}\|_1}, \quad (4.10)$$

where $\|\cdot\|_1$ denotes the number of positive items in the vector. F_1 in the vector form is then:

$$F_1(\hat{\mathbf{y}}, \mathbf{y}) = \frac{2\hat{\mathbf{y}} \cdot \mathbf{y}}{\|\hat{\mathbf{y}}\|_1 + \|\mathbf{y}\|_1} \text{ and } F_1(\mathbf{0}, \mathbf{0}) = 1. \quad (4.11)$$

Let's consider an example, suppose the gold standard $\mathbf{y} = 10110010$, and the prediction $\hat{\mathbf{y}} = 11100110$, then following their definitions, we get:

$$accuracy(\hat{\mathbf{y}}, \mathbf{y}) = \frac{3+2}{8} = \frac{5}{8},$$

$$precision(\hat{\mathbf{y}}, \mathbf{y}) = \frac{3}{5},$$

$$recall(\hat{\mathbf{y}}, \mathbf{y}) = \frac{3}{4},$$

$$F_1(\hat{\mathbf{y}}, \mathbf{y}) = \frac{2 \times 3}{4 + 5} = \frac{2}{3}.$$

4.1.3 Precision at k

In Information Retrieval, *precision at k*, or $P@k$ is commonly used for learning to rank tasks. It is because for a search engine, what matters most is not correctly retrieving all relevant web pages/documents, but how many retrieved web pages/documents *on the first one or two pages* are actually relevant (i.e., positive), for example, top 10 retrieved pages. So this gives the definition of $P@k$ in vector form:

$$P@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\hat{\mathbf{y}} \cdot \mathbf{y}}{k}, \text{ where } \|\hat{\mathbf{y}}\|_1 = k. \quad (4.12)$$

Considering $\mathbf{y} = 10110010$ as an example, suppose we let $k = 3$, and the prediction $\hat{\mathbf{y}} = 11000010$, then $P@3(\hat{\mathbf{y}}, \mathbf{y}) = \frac{2}{3}$.

4.1.4 Discounted Cumulative Gain

Discounted cumulative gain, or *DCG* (46) is another popular metric used in Information Retrieval. It assumes the higher relevant documents have higher ranked positions in the list of retrieved documents. If we consider the retrieved list of items (e.g., documents, web pages) as a permutation, $\hat{\sigma}$, where $\hat{\sigma}(i)$ denotes the i^{th} -ranked item. Let $y_{\hat{\sigma}(i)}$ be the relevance score (gain) for item $\hat{\sigma}(i)$. Then *DCG*, accumulated from position 1 to n , can be defined as following:

$$DCG(\hat{\sigma}, \mathbf{y}) = \sum_{i=1}^n \frac{2^{y_{\hat{\sigma}(i)}} - 1}{\log_2(i + 1)}. \quad (4.13)$$

Note that there's another version of the definition of *DCG*:

$$DCG'(\hat{\sigma}, \mathbf{y}) = y_{\hat{\sigma}(1)} + \sum_{i=2}^n \frac{y_{\hat{\sigma}(i)}}{\log_2 i}. \quad (4.14)$$

The difference between these two is that Equation 4.13 puts stronger emphasis on retrieving highly relevant items than Equation 4.14, i.e., the numerator in Equation 4.13 is $2^{y_{\hat{\sigma}(i)}}$ instead of $y_{\hat{\sigma}(i)}$ in Equation 4.14. Since the definition of *DCG* in Equation 4.13 is more widely used in the industry, and is the pseudo-standard (47; 48), we use this version in this thesis.

Consider the following ranking of eight items as an example, where each value is the relevance score $\in \{0, 1, 2\}$:

$$2, \quad 1, \quad 2, \quad 0, \quad 0, \quad 2, \quad 1, \quad 0.$$

Then, the discounted gains at each position are:

$$\begin{aligned} & \frac{2^2 - 1}{\log_2 2}, \quad \frac{2^1 - 1}{\log_2 3}, \quad \frac{2^2 - 1}{\log_2 4}, \quad \frac{2^0 - 1}{\log_2 5}, \quad \frac{2^0 - 1}{\log_2 6}, \quad \frac{2^2 - 1}{\log_2 7}, \quad \frac{2^1 - 1}{\log_2 8}, \quad \frac{2^0 - 1}{\log_2 9} \\ = & 3, \quad 0.631, \quad 1.5, \quad 0, \quad 0, \quad 1.069, \quad 0.333, \quad 0. \end{aligned}$$

The *DCGs* at rank $n = \{1, 2, \dots, 8\}$ are the accumulations of values above:

$$\begin{aligned} & 3, \quad 3 + 0.631, \quad 3.631 + 1.5, \quad 5.131 + 0, \quad 5.131 + 0, \quad 5.131 + 1.069, \quad 6.2 + 0.333, \quad 6.533 + 0 \\ = & 3, \quad 3.631, \quad 5.131, \quad 5.131, \quad 5.131, \quad 6.2, \quad 6.533, \quad 6.533. \end{aligned}$$

If the items are ordered in descending order of relevance scores, we can get an ideal ranking, and its corresponding *DCG* is called *ideal DCG*, or *IDCG*. Continue the example above, the ideal ranking is:

$$2, \quad 2, \quad 2, \quad 1, \quad 1, \quad 0, \quad 0, \quad 0,$$

thus its ideal discounted gains at each position are:

$$\begin{aligned} & \frac{2^2 - 1}{\log_2 2}, \quad \frac{2^2 - 1}{\log_2 3}, \quad \frac{2^2 - 1}{\log_2 4}, \quad \frac{2^1 - 1}{\log_2 5}, \quad \frac{2^1 - 1}{\log_2 6}, \quad \frac{2^0 - 1}{\log_2 7}, \quad \frac{2^0 - 1}{\log_2 8}, \quad \frac{2^0 - 1}{\log_2 9} \\ = & 3, \quad 1.893, \quad 1.5, \quad 0.431, \quad 0.387, \quad 0, \quad 0, \quad 0. \end{aligned}$$

and the *IDCGs* at rank $n = \{1, 2, \dots, 8\}$ are cumulated as followings:

$$\begin{aligned} & 3, 3 + 1.893, 4.893 + 1.5, 6.393 + 0.431, 6.824 + 0.387, 7.211 + 0, 7.211 + 0, 7.211 + 0 \\ = & 3, 4.893, 6.393, 6.824, 7.211, 7.211, 7.211, 7.211. \end{aligned}$$

The *normalized discounted cumulative gain*, or *NDCG* is defined as:

$$NDCG = \frac{DCG}{IDCG}. \quad (4.15)$$

The *NDCGs* at each position in the example are:

$$\begin{aligned} & \frac{3}{3}, \frac{3.631}{4.893}, \frac{5.131}{6.393}, \frac{5.131}{6.824}, \frac{5.131}{7.211}, \frac{6.2}{7.211}, \frac{6.533}{7.211}, \frac{6.533}{7.211} \\ = & 1, 0.742, 0.803, 0.752, 0.712, 0.860, 0.906, 0.906. \end{aligned}$$

4.1.5 Alignment Error Rate

The *alignment error rate* (*AER*) is a common evaluation measure for assessing word alignment quality in machine translation (49; 17; 50; 51). It generalizes the F-score for settings with more than binary-valued tags (52). For example, an alignment task may contain three different kinds of tags between each pair of source and target words: a sure tag (*S*) for unambiguous alignments, a possible tag (*P*) for alignments that might exist or not, and a negative tag (*N*)

for alignments that are neither S nor P . For a gold standard sequence of alignments \mathbf{y} , and a proposed sequence of alignments \mathbf{a} from the system under evaluation, AER is defined as:

$$AER(\mathbf{a}, \mathbf{y}) = 1 - \frac{|\mathbf{a}_A \cap \mathbf{y}_S| + |\mathbf{a}_A \cap \mathbf{y}_P|}{|\mathbf{a}_A| + |\mathbf{y}_S|}, \quad (4.16)$$

where A is the proposed positive tag. S alignments are also considered to be P alignments ($S \subseteq P$) in this measure and it is evaluated accordingly.

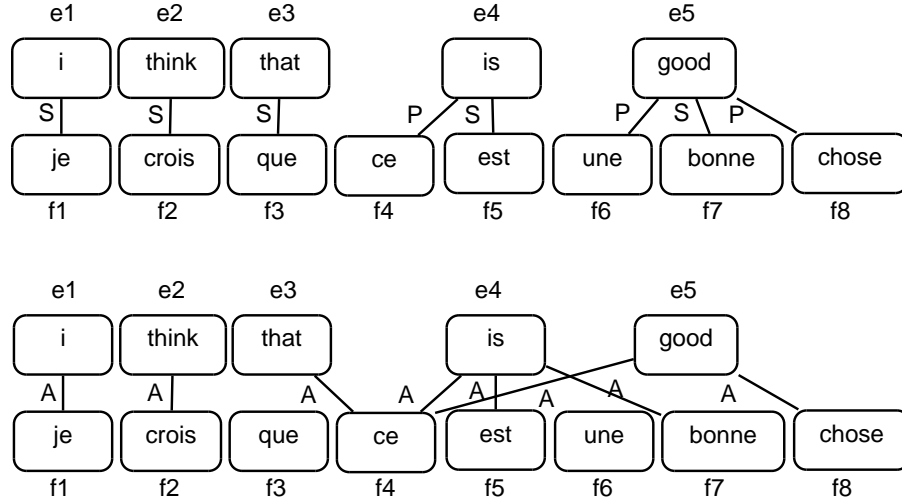


Figure 4: The gold standard sequence alignment \mathbf{y} (top) between English words e_i and French words f_j , and the proposed sequence alignment \mathbf{a} (bottom) with AER of 5/13. Note that N tags are omitted in the figure for clarity.

Figure 4 shows an example alignment task. Each complete alignment can be viewed as a variable for each pair of source and target words, as shown in Table VII, or represented in sequence form as $SNNNNNNNN\dots NPSP$ and $ANNNNNNNN\dots NNNA$, for the gold standard and proposed sequence alignments. The alignment error rate for this example is:

$$AER = 1 - \frac{3 + 5}{8 + 5} = \frac{5}{13}.$$

TABLE VII: Sequences of the gold standard (left) and the proposed (right) source-target word alignments of the example in Figure 4.

Gold	f1	f2	f3	f4	f5	f6	f7	f8	Prop.	f1	f2	f3	f4	f5	f6	f7	f8
e1	S	N	N	N	N	N	N	N	e1	A	N	N	N	N	N	N	N
e2	N	S	N	N	N	N	N	N	e2	N	A	N	N	N	N	N	N
e3	N	N	S	N	N	N	N	N	e3	N	N	N	A	N	N	N	N
e4	N	N	N	P	S	N	N	N	e4	N	N	N	A	A	N	A	N
e5	N	N	N	N	N	P	S	P	e5	N	N	N	A	N	N	N	A

4.2 Best Response Algorithms

As we show in Section 3.2 and Figure 3, there's one more major component left without discussion: the Best Response Finder. It encapsulates polynomial time algorithms for finding a best response action to the opponent strategy (i.e., possible actions and their probability distribution). To motivate the study of best response finding algorithms, let's first take a look at the following example of F_1 score, and see what is the best response.

Suppose we are given an opponent strategy $\mathbf{Y} : \{\mathbf{y}_1 = 101, \mathbf{y}_2 = 111, \mathbf{y}_3 = 010\}$, with probabilities $P(\mathbf{Y})$: $\{p(\mathbf{y}_1) = 0.5, p(\mathbf{y}_2) = 0.2, p(\mathbf{y}_3) = 0.3\}$. Consider the following tables as examples, if we have an action $\hat{\mathbf{y}} = 100$ as a response, we can get the expectation, $\mathbb{E}_{P(\mathbf{Y})}[F_1(\mathbf{y}, \hat{\mathbf{y}})] = 0.4333$. For another possible response $\hat{\mathbf{y}}' = 110$, we have $\mathbb{E}_{P(\mathbf{Y})}[F_1(\mathbf{y}, \hat{\mathbf{y}}')] = 0.61$:

	\mathbf{Y}			$F_1(\mathbf{y}, \hat{\mathbf{y}})$
$p(\mathbf{y}_1) = 0.5$	1	0	1	$2/3$
$p(\mathbf{y}_2) = 0.2$	1	1	1	$1/2$
$p(\mathbf{y}_3) = 0.3$	0	1	0	0
$\hat{\mathbf{y}}$	1	0	0	$\mathbb{E}_{P(\mathbf{Y})}[F_1(\mathbf{y}, \hat{\mathbf{y}})] = 0.5 \times 2/3 + 0.2 \times 1/2 + 0.3 \times 0 = 0.4333$

	\mathbf{Y}			$F_1(\mathbf{y}, \hat{\mathbf{y}}')$
$p(\mathbf{y}_1) = 0.5$	1	0	1	$1/2$
$p(\mathbf{y}_2) = 0.2$	1	1	1	$4/5$
$p(\mathbf{y}_3) = 0.3$	0	1	0	$2/3$
$\hat{\mathbf{y}}'$	1	1	0	$\mathbb{E}_{P(\mathbf{Y})}[F_1(\mathbf{y}, \hat{\mathbf{y}}')] = 0.5 \times 1/2 + 0.2 \times 4/5 + 0.3 \times 2/3 = 0.61$

Now, the question is which response $\hat{\mathbf{y}}$ is the best $\hat{\mathbf{y}}^*$ that has the largest expected F_1 score?

$$\hat{\mathbf{y}}^* = \arg \max_{\hat{\mathbf{y}}} \mathbb{E}_{P(\mathbf{Y})}[F_1(\mathbf{y}, \hat{\mathbf{y}})]. \quad (4.17)$$

The brute-force method would check all $2^{|\mathbf{Y}|}$ possible permutations of $\hat{\mathbf{y}}$, which only works when the length of vector (i.e., the number of items) is very small. Such difficulty motivates our studies on best response algorithms described in the following subsections.

4.2.1 Best Response for F_1 Score

We leverage a recently developed method for efficiently maximizing the F_1 score when a distribution over opponent strategy is given (39). The key insight is that the problem can be separated into an inner greedy maximization over item sets of a certain size k and an outer maximization to select the best set size k from $\{0, \dots, n\}$, where n is the number of items. This method can be directly applied to find the best response of the estimator player, \hat{Y} , since, as showed in Equation 3.11, the Lagrangian terms of the game payoff matrix are invariant to the

choice of $\hat{\mathbf{y}}$. For finding the best response for the adversary player \check{Y} (performance-minimizer), we give the derivation:

$$\begin{aligned}
\check{\mathbf{y}}^* &= \arg \min_{\check{\mathbf{y}}} \mathbb{E}_{P(\hat{\mathbf{y}}|\mathbf{x})} [F_1(\hat{\mathbf{Y}}, \check{\mathbf{y}}) - \psi(\check{\mathbf{y}})] \\
&= \arg \min_{\check{\mathbf{y}}} \sum_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}|\mathbf{x}) [F_1(\hat{\mathbf{y}}, \check{\mathbf{y}}) - \psi(\check{\mathbf{y}})] \\
&= \arg \min_{\check{\mathbf{y}}^{(k)*}} \sum_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}|\mathbf{x}) [F_1(\hat{\mathbf{y}}, \check{\mathbf{y}}^{(k)*}) - \psi(\check{\mathbf{y}}^{(k)*})], \tag{4.18} \\
\check{\mathbf{y}}^{(k)*} &= \arg \min_{\check{\mathbf{y}} \in \check{Y}^{(k)}} \sum_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}|\mathbf{x}) [F_1(\hat{\mathbf{y}}, \check{\mathbf{y}}) - \psi(\check{\mathbf{y}})] \\
&= \arg \min_{\check{\mathbf{y}} \in \check{Y}^{(k)}} \sum_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}|\mathbf{x}) \left[\frac{2 \sum_{i=1}^n \hat{y}_i \check{y}_i}{\sum_{i=1}^n \hat{y}_i + \sum_{i=1}^n \check{y}_i} - \sum_{i=1}^n \psi_i \check{y}_i \right] \\
&= \arg \min_{\check{\mathbf{y}} \in \check{Y}^{(k)}} \sum_{i=1}^n \left[\frac{\sum_{\hat{\mathbf{y}}} 2p(\hat{\mathbf{y}}|\mathbf{x}) \hat{y}_i \check{y}_i}{s_{\hat{\mathbf{y}}} + k} - \psi_i \check{y}_i \right] \\
&= \arg \min_{\check{\mathbf{y}} \in \check{Y}^{(k)}} \sum_{i=1}^n \check{y}_i \left[\frac{\sum_{\hat{\mathbf{y}}} 2p(\hat{\mathbf{y}}|\mathbf{x}) \hat{y}_i}{s_{\hat{\mathbf{y}}} + k} - \psi_i \right] \\
&= \arg \min_{\check{\mathbf{y}} \in \check{Y}^{(k)}} \sum_{i=1}^n \check{y}_i \left[\sum_{s=1}^n \frac{\hat{p}_{is}}{s + k} - \psi_i \right], \tag{4.19}
\end{aligned}$$

where we write F_1 in its vector form:

$$F_1(\hat{\mathbf{y}}, \check{\mathbf{y}}) = \frac{2\hat{\mathbf{y}} \cdot \check{\mathbf{y}}}{\|\hat{\mathbf{y}}\|_1 + \|\check{\mathbf{y}}\|_1} = \frac{2 \sum_{i=1}^n \hat{y}_i \check{y}_i}{\sum_{i=1}^n \hat{y}_i + \sum_{i=1}^n \check{y}_i},$$

and $s_{\hat{\mathbf{y}}} = \sum_{i=1}^n \hat{y}_i$ is the total number of ‘1’s in $\hat{\mathbf{y}}$; $k = \sum_{i=1}^n \check{y}_i$ is the total number of ‘1’s in $\check{\mathbf{y}}$; and $\psi_i = \boldsymbol{\theta} \cdot \boldsymbol{\phi}(\mathbf{x}_i)$ for each position i . The marginal probability of $\hat{\mathbf{Y}}$ that there are total s ‘1’s, and i -th position is ‘1’ is defined as \hat{p}_{is} :

$$\hat{p}_{is} = p(\hat{Y}_i = 1, s_{\hat{\mathbf{y}}} = s | \mathbf{x}) = \sum_{\hat{\mathbf{y}}: s_{\hat{\mathbf{y}}} = s} \hat{y}_i p(\hat{\mathbf{y}} | \mathbf{x}). \quad (4.20)$$

Then matrix \mathbf{W} with element $w_{sk} = \frac{1}{s+k}$, where $s, k \in \{1, \dots, n\}$, can be precomputed. Computing the marginal probability matrix $\hat{\mathbf{P}}$ that has \hat{p}_{is} as its elements, requires going through every $\hat{\mathbf{y}}$ once. Algorithm 2 shown below obtains the best response for the adversary player \check{Y} , that incorporates the Lagrangian potentials, and has a time complexity $\mathcal{O}(n^3)$.

Algorithm 2 General F-score minimizer for MPG adversary player \check{Y}

```

1: procedure GFM4MPG(marginal probability matrix  $\hat{\mathbf{P}}$ , Lagrangian potentials  $\psi$ )
2:   define matrix  $\mathbf{W}$  with element  $\mathbf{W}_{s,k} = \frac{1}{s+k}$ ,  $s, k \in \{1, \dots, n\}$ 
3:   construct matrix  $\mathbf{F} = 2 \times \hat{\mathbf{P}} \times \mathbf{W} - \psi^T \times \mathbf{1}^n$   $\triangleright \mathbf{1}^n$  is the all ones  $1 \times n$  vector
4:   for  $k = 1$  to  $n$  do
5:     solve the inner optimization problem:
6:      $\check{\mathbf{y}}^{(k)*} = \arg \min_{\check{\mathbf{y}} \in \check{\mathbf{Y}}^{(k)}} \sum_{i=1}^n \check{y}_i f_{ik}$   $\triangleright \check{\mathbf{Y}}^{(k)} = \{\check{\mathbf{y}} \in \{0, 1\}^n \mid \sum_{i=1}^n \check{y}_i = k\}$ 
7:     set  $\check{y}_i^{(k)} = 1$  for the  $k$ -th column of  $\mathbf{F}$ 's smallest  $k$  elements, and  $\check{y}_i = 0$  for the rest;
8:     store a value of  $\mathbb{E}_{P(\hat{\mathbf{y}}|\mathbf{x})}[\mathbf{F}(\hat{\mathbf{y}}, \check{\mathbf{y}}^{(k)*})] = \sum_{i=1}^n \check{y}_i^{(k)*} f_{ik}$ 
9:   end for
10:  for  $k = 0$  take  $\check{\mathbf{y}}^{(k)*} = \mathbf{0}^n$ , and  $\mathbb{E}_{P(\hat{\mathbf{y}}|\mathbf{x})}[\mathbf{F}(\hat{\mathbf{y}}, \mathbf{0}^n)] = p(\hat{\mathbf{Y}} = \mathbf{0}^n | \mathbf{x})$ 
11:  solve the outer optimization problem:
12:   $\check{\mathbf{y}}^* = \arg \min_{\check{\mathbf{y}} \in \{\check{\mathbf{y}}^{(0)*}, \dots, \check{\mathbf{y}}^{(n)*}\}} \mathbb{E}_{P(\hat{\mathbf{y}}|\mathbf{x})}[\mathbf{F}(\hat{\mathbf{y}}, \check{\mathbf{y}})]$ 
13:  return  $\check{\mathbf{y}}^*$  and  $\mathbb{E}_{P(\hat{\mathbf{y}}|\mathbf{x})}[\mathbf{F}(\hat{\mathbf{y}}, \check{\mathbf{y}}^*)]$ 
14: end procedure

```

4.2.2 Best Response for F_1 Score with Multi-class Linear-chain Structure

Leveraging sequential structure improves performance in tasks like named-entity recognition (53). This has previously been accomplished using conditional random fields (12), which are based on the logarithmic loss rather than the F_1 score often used to evaluate performance for such tasks. Better aligning the learning method’s objective and the desired evaluation measure within the MPG framework requires us to find efficient multi-class best response algorithms with linear-chain structure for each player.

We consider the F_1 score for a particular class C and define two players in the zero-sum game: player $\hat{\mathbf{Y}}$ makes predictions that maximizes F_1 score, and player $\check{\mathbf{Y}}$ adversarially approximates the evaluation distribution. For each set of adversarial sequences \mathbb{Y} , and its distribution $P(\mathbb{Y})$, the best response should be found efficiently:

$$\hat{\mathbf{y}}^* = \arg \max_{\hat{\mathbf{y}} \in \hat{\mathbb{Y}}} \sum_{\check{\mathbf{y}}} p(\check{\mathbf{y}}|\mathbf{x}) [F_{1_C}(\hat{\mathbf{y}}, \check{\mathbf{y}})], \quad (4.21)$$

$$\check{\mathbf{y}}^* = \arg \min_{\check{\mathbf{y}} \in \check{\mathbb{Y}}} \sum_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}|\mathbf{x}) [F_{1_C}(\hat{\mathbf{y}}, \check{\mathbf{y}}) - \psi(\check{\mathbf{y}})]. \quad (4.22)$$

In our proposed model, there are two types of features: unigram $\Phi^u(\mathbf{x}_t, y_t)$, and bigram $\Phi^b(\mathbf{x}_t, y_{t-1}, y_t)$. We encode the linear-chain structure information in the weight vectors (i.e., Lagrange multipliers in Equation 4.22). For example, suppose we have m classes $C_t \in \mathbb{C} = \{C^1, \dots, C^m\}$ in the dataset. Then each distinct consecutive pair of classes (C_{t-1}, C_t) has its own weight vector $\theta^b(C_{t-1}, C_t)$, for a total number of pairs and weight vectors m^2 . An optional *START* tag can be added in front of a sequence, forming m additional pairs $(START, C_1)$ (and

corresponding weight vectors). Besides these pair vectors that accommodate bigram features, each class also has a separate weight vector $\theta^u(C_t)$ for unigram features in our model. So in total, we use m unigram feature vectors, and $m^2(+m)$ bigram feature vectors to capture the linear chain information.

From Equation 4.21, we can see that the Lagrange potentials $\psi(\tilde{\mathbf{y}})$ are related to the choice of $\tilde{\mathbf{y}}$ only and is discarded, and due to the binary nature of the F_1 score of a specific target class (15), the *GFM* algorithm (39) can be applied directly to the binarized sequences for the target.

For the adversary's best response, we rewrite Equation 4.22 for a particular class C as:

$$\begin{aligned} \tilde{\mathbf{y}}^* &= \arg \min_{\tilde{\mathbf{y}}^{(k)*}} \sum_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}|\mathbf{x}) \left\{ \frac{2 \sum_{t=1}^n \hat{y}_{Ct} \tilde{y}_{Ct}^{(k)*}}{\alpha + k} - \sum_{t=1}^n \left[\psi_t^u(\tilde{y}_t^{(k)*}) + \psi_t^b(\tilde{y}_{t-1}^{(k)*}, \tilde{y}_t^{(k)*}) \right] \right\}, \quad (4.23) \\ \tilde{\mathbf{y}}^{(k)*} &= \arg \min_{\tilde{\mathbf{y}} \in \tilde{\mathbf{Y}}^{(k)}} \sum_{t=1}^n \left\{ \left(\sum_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}|\mathbf{x}) \frac{2 \hat{y}_{Ct} \tilde{y}_{Ct}}{\alpha + k} \right) - \left[\psi_t^u(\tilde{y}_t) + \psi_t^b(\tilde{y}_{t-1}, \tilde{y}_t) \right] \right\} \\ &= \arg \min_{\tilde{\mathbf{y}} \in \tilde{\mathbf{Y}}^{(k)}} \sum_{t=1}^n \left\{ \sum_{\alpha=1}^n \frac{2 \hat{p}_{t\alpha}}{\alpha + k} \tilde{y}_{Ct} - \left[\psi_t^u(\tilde{y}_t) + \psi_t^b(\tilde{y}_{t-1}, \tilde{y}_t) \right] \right\} \\ &= \arg \max_{\tilde{\mathbf{y}} \in \tilde{\mathbf{Y}}^{(k)}} \sum_{t=1}^n \left\{ \underbrace{\left[\psi_t^u(\tilde{y}_t) + \psi_t^b(\tilde{y}_{t-1}, \tilde{y}_t) \right]}_{f_{tk}} - \sum_{\alpha=1}^n \frac{2 \hat{p}_{t\alpha}}{\alpha + k} \tilde{y}_{Ct} \right\}, \quad (4.24) \end{aligned}$$

where $\hat{y}_{Ct} = I(\hat{y}_t = C)$, $\tilde{y}_{Ct} = I(\tilde{y}_t = C)$, and $\alpha = \sum_{t=1}^n \hat{y}_{Ct}$. $\hat{p}_{t\alpha}$ is the marginal probability that $|\hat{y}_C| = k$ and $\hat{y}_t = C$:

$$\hat{p}_{t\alpha} = p(\hat{Y}_t = C, \sum_t I(\hat{Y}_t = C) = k | \mathbf{x}). \quad (4.25)$$

The difficulty of solving Equation 4.24 comes from the linear-chain structure in the Lagrange potential term $[\psi_t^u(\tilde{y}_t) + \psi_t^b(\tilde{y}_{t-1}, \tilde{y}_t)]$. To solve this problem efficiently, we propose a dynamic programming algorithm, for the particular class C , *Linear-Chain F-score Minimizer (LCFM)* in Algorithm 3. Figure 5 shows an example, where $k = 2$, the target class $C = LOC$. We start from the end of the linear-chain structure. For each $t - 1$ position, the expectation value $[\psi_t^u(\tilde{y}_t) + \psi_t^b(\tilde{y}_{t-1}, \tilde{y}_t) - f_{tk}I(\tilde{y}_t = C)]$ is accumulated from position t . Also, for each possible class node, the algorithm keeps tracking the best expectation values with and without involving the target class, and the total number of target class appearances so far, to guarantee such number equals to k at the end of the procedure. When there are more than one possible route with same number of target class appearances (e.g., in the red rectangle in the bottom figure in Figure 5), the one with largest accumulated expectation value is selected. Looping through n subroutines of *MSUM* can be accomplished in $\mathcal{O}(m^2n^3)$ time, which characterizes the overall complexity of the algorithm.

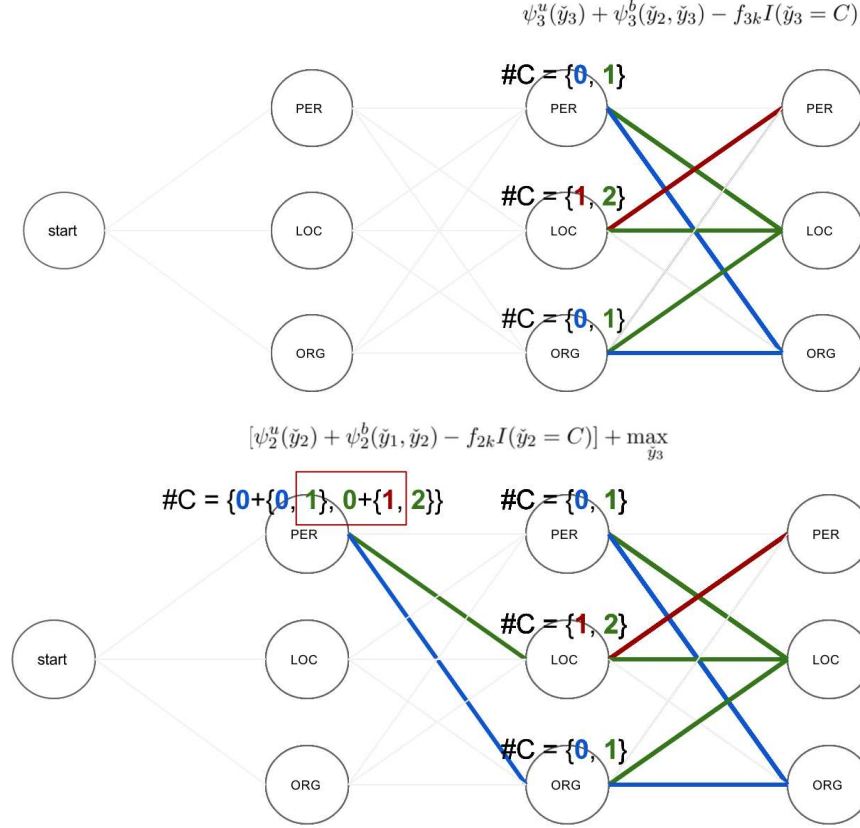


Figure 5: An example of linear-chain structure, with $k = 2$, and the target class $C = LOC$. Green links represent the chains with the target class C , the others are links without it.

Algorithm 3 Linear-Chain F-score Minimizer.

```

1: procedure LCFM( $\hat{\mathbf{P}}, \psi^u, \psi^b, \mathbf{C}$ )
2:   define matrix  $\mathbf{W}$  with element  $w_{tk} = \frac{1}{t+k}$ 
3:   compute matrix  $\mathbf{F} = 2 \times \hat{\mathbf{P}} \times \mathbf{W}$  ▷ each element is  $f_{tk}$  in Equation 4.24
4:    $[\check{\mathbf{y}}^{(0)*}, \mathbb{E}_{F_1}(\check{\mathbf{y}}^{(0)*})] = \text{MSUM}(\text{START}, 1, 0, 0, \mathbf{F})$ 
5:    $\mathbb{E}_{F_1}(\check{\mathbf{y}}^{(0)*}) = \mathbb{E}_{F_1}(\check{\mathbf{y}}^{(0)*}) - p_0$  ▷  $p_0$  is the marginal probability that  $|\hat{y}_C| = 0$ 
6:   for  $k = 1$  to  $n$  do
7:      $[\check{\mathbf{y}}^{(k)*}, \mathbb{E}_{F_1}(\check{\mathbf{y}}^{(k)*})] = \text{MSUM}(\text{START}, 1, k, k, \mathbf{F})$ 
8:   end for
9:    $\mathbb{E}_{F_1}(\check{\mathbf{y}}^*) = \max(\mathbb{E}_{F_1}(\check{\mathbf{y}}^{(k)*}))$ 
10:   $\check{\mathbf{y}}^* = \arg \max_{\check{\mathbf{y}}^{(k)*}} \mathbb{E}_{F_1}(\check{\mathbf{y}}^*)$ 
11:  return  $[\check{\mathbf{y}}^*, -\mathbb{E}_{F_1}(\check{\mathbf{y}}^*)]$ 
12: end procedure

```

4.2.3 Best Response for Precision at k

According to its definition, *precision at k*, or $P@k$, restricts the number of positive predictions to k , which is a constant. Equation 3.11 that finding the best response for the maximizer $\hat{\mathbf{Y}}$ is rewritten as:

$$\begin{aligned}
\hat{\mathbf{y}}^* &= \hat{\mathbf{y}}^{(k)*} = \arg \max_{\check{\mathbf{y}}} \mathbb{E}_{P(\check{\mathbf{y}}|\mathbf{x})} [P@k(\hat{\mathbf{y}}, \check{\mathbf{Y}})] \\
&= \arg \max_{\check{\mathbf{y}} \in \hat{\mathbf{Y}}^{(k)}} \sum_{\check{\mathbf{y}}} p(\check{\mathbf{y}}|\mathbf{x}) [P@k(\hat{\mathbf{y}}, \check{\mathbf{y}})] \\
&= \arg \max_{\check{\mathbf{y}} \in \hat{\mathbf{Y}}^{(k)}} \sum_{\check{\mathbf{y}}} p(\check{\mathbf{y}}|\mathbf{x}) \frac{\sum_{i=1}^n \hat{y}_i \check{y}_i}{k} \\
&= \arg \max_{\check{\mathbf{y}} \in \hat{\mathbf{Y}}^{(k)}} \frac{1}{k} \sum_{i=1}^n \hat{y}_i \sum_{\check{\mathbf{y}}} p(\check{\mathbf{y}}|\mathbf{x}) \check{y}_i \\
&= \arg \max_{\check{\mathbf{y}} \in \hat{\mathbf{Y}}^{(k)}} \frac{1}{k} \sum_{i=1}^n \hat{y}_i \check{p}_i,
\end{aligned} \tag{4.26}$$

Algorithm 4 MSUM subroutine of LCFM.

```

1: procedure MSUM( $c_{t-1}, t, r, k, \mathbf{F}$ )
2:   if  $[\tilde{\mathbf{y}}^*, \mathbb{E}_{F_1}(\tilde{\mathbf{y}}^*)] = \text{cache}(c_{t-1}, t, r, k)$  exists then
3:     return  $[\tilde{\mathbf{y}}^{(k)*}, \mathbb{E}_{F_1}(\tilde{\mathbf{y}}^{(k)*})]$ 
4:   end if
5:   if  $t > n$  and  $r > 0$  then return  $[\Phi, -\infty]$ 
6:   else if  $t > n$  and  $r \leq 0$  then return  $[\Phi, 0]$ 
7:   end if
8:   for  $c_t \in \mathbb{C}$  do
9:     if  $r > 0 \mid c_t \neq C$  then
10:       $r' = r - I(c_t = C)$ 
11:       $[\tilde{\mathbf{y}}^{c_t}, \mathbb{E}_{F_1}^{c_t}(\tilde{\mathbf{y}})] = \text{MSUM}(c_t, t+1, r', k, \mathbf{F})$ 
12:    end if
13:  end for
14:   $\psi(c_t) = \psi_t^u(c_t) + \psi_t^b(c_{t-1}, c_t)$ 
15:   $f = f_{tk} \times I(c_t = C)$   $\triangleright f_{tk}$  is one element in  $\mathbf{F}$ 
16:   $\mathbb{E}_{F_1}(\tilde{\mathbf{y}}^*) = \max(\psi(c_t) - f + \mathbb{E}_{F_1}^{c_t}(\tilde{\mathbf{y}}))$ 
17:   $\tilde{\mathbf{y}}^* = \arg \max_{(c_t \oplus \tilde{\mathbf{y}}^{c_t})} \mathbb{E}_{F_1}(\tilde{\mathbf{y}}^*)$ 
18:  return  $\text{cache}(c_{t-1}, t, r, k) = [\tilde{\mathbf{y}}^*, \mathbb{E}_{F_1}(\tilde{\mathbf{y}}^*)]$ 
19: end procedure

```

where \check{p}_i is the marginal probability of $\check{\mathbf{Y}}$ that i -th position is ‘1’, i.e., $\check{p}_i = \sum_{\check{\mathbf{y}}} p(\check{\mathbf{y}}|\mathbf{x})\check{y}_i$. To get the best response $\hat{\mathbf{y}}^*$, we sort \check{p}_i in descending order, and set the corresponding top k variables in $\hat{\mathbf{y}}$ to ‘1’.

For the minimizer $\check{\mathbf{Y}}$'s best response, the Lagrangian terms must also be included:

$$\begin{aligned}
\check{\mathbf{y}}^* &= \arg \min_{\check{\mathbf{y}}} \mathbb{E}_{P(\hat{\mathbf{y}}|\mathbf{x})} [P@k(\hat{\mathbf{Y}}, \check{\mathbf{y}}) - \psi(\check{\mathbf{y}})] \\
&= \arg \min_{\check{\mathbf{y}}} \sum_{\hat{\mathbf{y}} \in \hat{\mathbf{Y}}^{(k)}} p(\hat{\mathbf{y}}|\mathbf{x}) [P@k(\hat{\mathbf{y}}, \check{\mathbf{y}}) - \sum_{i=1}^n \psi_i \check{y}_i] \\
&= \arg \min_{\check{\mathbf{y}}} \sum_{\hat{\mathbf{y}} \in \hat{\mathbf{Y}}^{(k)}} p(\hat{\mathbf{y}}|\mathbf{x}) \left(\frac{\sum_{i=1}^n \hat{y}_i \check{y}_i}{k} - \sum_{i=1}^n \psi_i \check{y}_i \right) \\
&= \arg \min_{\check{\mathbf{y}}} \sum_{i=1}^n \check{y}_i \left(\frac{\sum_{\hat{\mathbf{y}} \in \hat{\mathbf{Y}}^{(k)}} p(\hat{\mathbf{y}}|\mathbf{x}) \hat{y}_i}{k} - \psi_i \right) \\
&= \arg \min_{\check{\mathbf{y}}} \sum_{i=1}^n \check{y}_i \left(\frac{\hat{p}_{ik}}{k} - \psi_i \right), \tag{4.27}
\end{aligned}$$

where $\hat{p}_{ik} = \sum_{\hat{\mathbf{y}} \in \hat{\mathbf{Y}}^{(k)}} p(\hat{\mathbf{y}}|\mathbf{x}) \hat{y}_i = p(\hat{Y}_i = 1, \sum_{\hat{\mathbf{y}}} \hat{Y}_i = k|\mathbf{x})$, is the marginal probability of $\hat{\mathbf{Y}}$ that there are total k '1's, and i -th position is '1'. Since k is a known variable, as long as the value of each included term, $(\hat{p}_{ik}/k - \psi_i)$, is negative, the sum is the smallest, and the corresponding response is the best for the minimizer.

4.2.4 Best Response for Discounted Cumulative Gain

For computing the best response for maximizer, we rewrite Equation 3.11 as following:

$$\begin{aligned}
\hat{\sigma}^* &= \arg \max_{\hat{\sigma}} \mathbb{E}_{P(\check{\mathbf{y}}|\mathbf{x})} [DCG(\hat{\sigma}, \check{\mathbf{Y}})] \\
&= \arg \max_{\hat{\sigma}} \sum_{\check{\mathbf{y}}} p(\check{\mathbf{y}}|\mathbf{x}) \sum_{i=1}^n \frac{2^{\check{y}_{\hat{\sigma}(i)}} - 1}{\log_2(i+1)} \\
&= \arg \max_{\hat{\sigma}} \sum_{i=1}^n \frac{1}{\log_2(i+1)} \left(\sum_{\check{\mathbf{y}}} p(\check{\mathbf{y}}|\mathbf{x}) 2^{\check{y}_{\hat{\sigma}(i)}} - 1 \right). \tag{4.28}
\end{aligned}$$

Since $1/(\log_2(i+1))$ is monotonically decreasing, computing and sorting $(\sum_{\check{\mathbf{y}}} p(\check{\mathbf{y}}|\mathbf{x})2^{\check{y}_i} - 1)$ with descending order and greedily assign the order to $\hat{\sigma}$ is optimal.

The minimizers best response using additive features is obtained:

$$\begin{aligned}
\check{\mathbf{y}}^* &= \arg \min_{\check{\mathbf{y}}} \mathbb{E}_{P(\hat{\sigma}|\mathbf{x})} [DCG(\hat{\Sigma}, \check{\mathbf{y}}) - \psi(\check{\mathbf{y}})] \\
&= \arg \min_{\check{\mathbf{y}}} \sum_{\hat{\sigma}} p(\hat{\sigma}|\mathbf{x}) \left[DCG(\hat{\sigma}, \check{\mathbf{y}}) - \sum_{i=1}^n \psi_i(\check{y}_i) \right] \\
&= \arg \min_{\check{\mathbf{y}}} \sum_{\hat{\sigma}} p(\hat{\sigma}|\mathbf{x}) \left[\sum_{i=1}^n \frac{2^{\check{y}_{\hat{\sigma}(i)}} - 1}{\log_2(i+1)} - \sum_{i=1}^n \psi_i(\check{y}_i) \right] \\
&= \arg \min_{\check{\mathbf{y}}} \sum_{i=1}^n \left[\sum_{\hat{\sigma}} p(\hat{\sigma}|\mathbf{x}) \frac{2^{\check{y}_{\hat{\sigma}(i)}} - 1}{\log_2(i+1)} - \psi_i(\check{y}_i) \right]. \tag{4.29}
\end{aligned}$$

For each individual possible relevance score value s of \check{y}_i (for example, $s \in \{0, 1, 2\}$), it has a separate weight vector $\theta(s)$. We greedily assign the value s to each position i , in the best response $\check{\mathbf{y}}^*$, where s is the one that gives the smallest expectation $\left(\sum_{\hat{\sigma}} p(\hat{\sigma}|\mathbf{x}) \frac{2^s - 1}{\log_2(i+1)} - \psi_i(s) \right)$.

4.2.5 Best Response for Alignment Error Rate

In the MPG for $1 - AER$, the two players are: $\mathbf{y} \in \mathbb{Y}$, which adversarially approximates the gold standard alignment distribution; and $\mathbf{a} \in \mathbb{A}$, which maximizes $1 - AER$ (hence, minimizes AER), where \mathbb{A} and \mathbb{Y} are the domain of \mathbf{a} with a distribution $Q(\mathbb{A})$, and \mathbf{y} with a distribution

$Q(\mathbb{Y})$ respectively. For a sequence of alignments of length n , the objective of finding the best responses \mathbf{a}^* (Equation 3.11) and \mathbf{y}^* (Equation 3.12) are:

$$\mathbf{a}^* = \arg \max_{\mathbf{a} \in \mathbb{A}} \sum_{\mathbf{y} \in \{0, S, P\}^n} q(\mathbf{y}) [1 - AER(\mathbf{a}, \mathbf{y})], \quad (4.30)$$

$$\mathbf{y}^* = \arg \min_{\mathbf{y} \in \mathbb{Y}} \sum_{\mathbf{a} \in \{0, A\}^n} q(\mathbf{a}) [1 - AER(\mathbf{a}, \mathbf{y}) - \psi(\mathbf{y})]. \quad (4.31)$$

We focus first on the adversary's best response (Equation 4.31), which must incorporate the Lagrangian potential term, $\psi(\mathbf{y})$. For the choice of alignment \mathbf{y}^* , we separate the choices among all possible numbers of S tags, $k = 0, \dots, n$, for the alignment sequence of length n , and denote these sets as $\mathbb{Y}^{(k)}$. The best choice of a certain k is rewritten as follows, where a_i is the notational shorthand of the indicator function $I(a_i = A)$, and $y_{si} = I(y_i = S)$, $y_{ri} = I(y_i = P, y_i \neq S)$; the number of A tags in the alignment is $\alpha = \sum_{i=1}^n a_i$:

$$\begin{aligned} \mathbf{y}^{(k)*} &= \arg \min_{\mathbf{y} \in \mathbb{Y}^{(k)}} \sum_{\mathbf{a} \in \{0, A\}^n} q(\mathbf{a}) \left(\frac{\sum_{i=1}^n (2a_i y_{si} + a_i y_{ri})}{\alpha + k} - \sum_{i=1}^n \psi_i(y_i) \right) \\ &= \arg \min_{\mathbf{y} \in \mathbb{Y}^{(k)}} \sum_{\mathbf{a} \in \{0, A\}^n} q(\mathbf{a}) \sum_{i=1}^n \left(\frac{a_i (2y_{si} + y_{ri})}{\alpha + k} - \psi_i^{(S)} y_{si} - \psi_i^{(R)} y_{ri} \right) \\ &= \arg \min_{\mathbf{y} \in \mathbb{Y}^{(k)}} \sum_{i=1}^n y_{si} \underbrace{2 \sum_{\alpha=1}^n \left(\frac{q_{i\alpha}}{\alpha + k} - \psi_i^{(S)} \right)}_{f_{sik}} + \sum_{i=1}^n y_{ri} \underbrace{\sum_{\alpha=1}^n \left(\frac{q_{i\alpha}}{\alpha + k} - \psi_i^{(R)} \right)}_{f_{rik}}. \end{aligned} \quad (4.32)$$

Here, in Equation 4.32, $q_{i\alpha}$ is the marginal probability that alignment \mathbf{a} has $|\mathbf{a}_S| = \alpha$ and $a_i = S$ (i.e., the number of S tags equals to α , and the i -th position is S). We separate the Lagrangian potential ψ into two terms: $\psi^{(S)}$ for S tags, $\psi^{(R)}$ for P tags that are not also S

Algorithm 5 AER Maximizer

```

1: procedure AERMAX( $Q(\mathbb{A}), \psi^{(S)}, \psi^{(R)}$ )
2:   define vector  $W_0$  with element  $w_{i0} = \frac{1}{i}$ 
3:   define matrix  $W$  with element  $w_{ik} = \frac{1}{i+k}$ , where  $i, k \in \{1, \dots, n\}$ 
4:   compute vector  $F_0 = QW_0 - \psi^{(R)}$ 
5:   compute matrix  $F_S = 2QW - \psi^{(S)\top} \mathbf{1}^n$ 
6:   compute matrix  $F_R = QW - \psi^{(R)\top} \mathbf{1}^n$ 
7:   set positions of  $y^{(0)*}$  with  $f_{i0} < 0$  to ‘P’
8:    $\mathbb{E}_{(1-AER)'}(\mathbf{y}^{(0)*}) = \sum_{i=1}^n \min(f_{i0}, 0)$ 
9:   for  $k = 1$  to  $n$  do
10:    find  $\mathbf{y}^{(k)*}$  by:
11:    sort  $f_{ik} = f_{sik} - \min(f_{rik}, 0)$  in ascending order
12:    set positions with top  $k$ ’s  $f_{ik}$  to ‘S’
13:    set each rest position  $i$  to ‘P’ if  $f_{rik} < 0$ 
14:     $\mathbb{E}_{(1-AER)'}(\mathbf{y}^{(k)*}) = \sum_{i=1}^n y_{si}f_{sik} + y_{ri}f_{rik}$ 
15:   end for
16:   return  $\mathbf{y}^* = \arg \min_{\mathbf{y}^{(k)*}} \mathbb{E}_{(1-AER)'}(\mathbf{y}^{(k)*})$ 
17: end procedure

```

tags. To get the permutation that minimizes this equation, for tag S at position i we pay f_{sik} , for tag P we pay f_{rik} , and for tag N we pay 0. Without the $\mathbf{y} \in \mathbb{Y}^{(k)}$ constraint, to compute the minimum, all that we need to do is finding the smallest of these three terms for each i . With the constraint, we have to set exactly k tags to S , so we choose the k positions where the f_{sik} cost exceeds the best alternative, $\min(f_{rik}, 0)$, by as little as possible. Thus, we sort $(f_{sik} - \min(f_{rik}, 0))$ in ascending order, set the top k positions to S , P , or N accordingly. The detailed algorithm is shown as Algorithm 5.

The best response for the AER minimizer (Equation 4.30) is simpler to obtain since the Lagrangian terms $\psi(\mathbf{y})$ are invariant to the choice of alignment \mathbf{a}^* . The approach of (39) can

be used after replacing $F = PW$ with $F' = Q_S W_S + Q_P W_P$, where matrix Q_S is the marginal probability for S tag, Q_P is for P tag, and permutation matrices W_S , W_P are for S and P respectively, where each element (with index i, k) in the matrix $w_S^{ik} = \frac{2}{i+k}$, and $w_P^{ik} = \frac{1}{i+k-1}$.

4.2.6 Best Response for Context-free Grammar Parsing

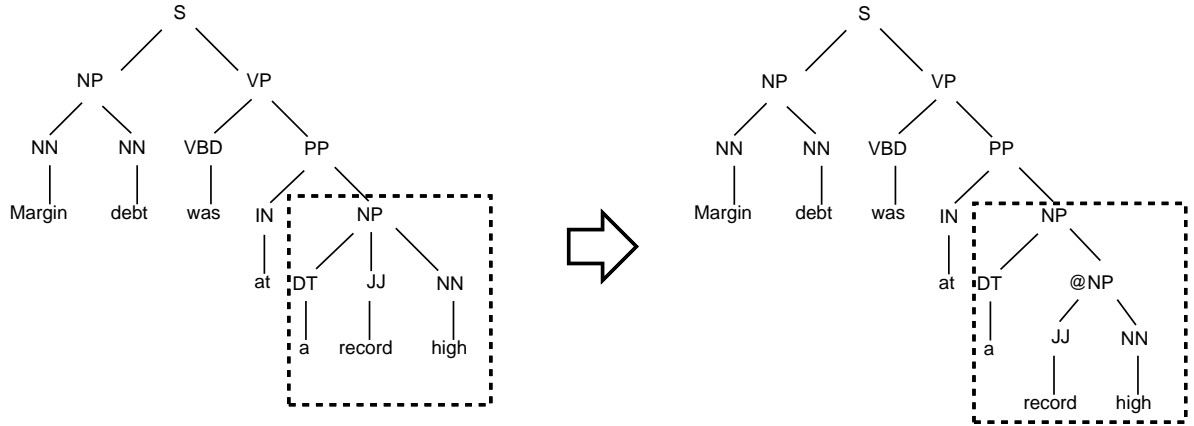


Figure 6: The parse tree (left) for the sentence “Margin debt was at a record high,” and its corresponding binarized parse tree (right), where ‘@NP’ is a synthetic node added to make the tree binary.

Syntactic parsing assigns a syntactic tree structure to a given sentence according to a set of predefined context-free grammar (CFG) (15). For example, Figure 6 shows a parse tree structure of the sentence “Margin debt was at a record high,” which uses grammars from the Penn Treebank (54). Note that the noun phrase (NP) node in the left square with dotted

line can be binarized to the one in the right square, which contains a synthetic node $@NP$. Binarization is necessary for efficient parsing algorithms, like CYK (15; 55; 56). The binarized parse tree is in Chomsky normal form (CNF) (57; 58; 59), which allows rules of two forms: $A \rightarrow BC$ (binary rule), and $D \rightarrow w$ (lexicon rule); where A , B , C and D are non-terminal symbols (state tags of grammar), and w is a terminal symbol (word in sentence). The parse tree can be represented in the form of a *parse table* (Figure 7). The row number of a non-terminal symbol indicates the rule's start index of its span of words, while the column number indicates its end index.

	0	1	2	3	4	5	6
0	Margin	NN	NP				S
1		debt	NN				
2			was	VBD			VP
3				at	IN		PP
4					a	DT	NP
5						record	JJ @NP
6							high NN

Figure 7: The binarized parse table for the sentence “Margin debt was at a record high.” The row number of a non-terminal symbol indicates the rule's start index of its span of words, while the column number indicates its end index.

For the syntactic parsing problem, we model each parse tree structure as \mathbf{y} , each rule r has a corresponding feature vector $\psi(r)$ to characterize its statistics from the training data. The score between two parse tree structures $\hat{\mathbf{y}}$ and $\check{\mathbf{y}}$ is measured as $(\hat{\mathbf{y}} \cap \check{\mathbf{y}})/(\hat{\mathbf{y}} \cup \check{\mathbf{y}})$, based on each constituent with span information (e.g., $VP \rightarrow VBD, PP, start = 2, end = 6$, in Figure 6). We extend the Viterbi-style dynamic programming algorithm CYK (55; 15) to support efficient MPG learning (see Algorithm 6, referring to the *parse table* in Figure 6). We adversarially optimize Hamming loss due to its tractability in tree structures.

There are two key differences between the original CYK and our CYK4MPG algorithm:

1. Since we are dealing with a best response for an adversary’s strategy (multiple possible actions) each time, the expected score($\hat{\mathbf{Y}}, \check{\mathbf{Y}}$) in Equation 3.2 is accumulated as parts of the marginal probability (see line 2 - 8).
2. The best score of each constituent r is computed by combining marginal probability (i.e., score) and Lagrange potential (see line 13, 23).

Suppose the grammar size is $|G|$, the overall time complexity of CYK4MPG is $\mathcal{O}(|G|n^3)$, where n is the length of the sentence to parse.

Algorithm 6 CYK algorithm for MPG

```

1: procedure CYK4MPG( $\hat{Y}$ ,  $P(\hat{Y})$ ,  $\psi$ , grammar, words)
2:   for  $\hat{y} \in \hat{Y}$  do ▷ Record  $p(\hat{y})$  for each constituent
3:     for constituent  $r \in \hat{y}$  do
4:       if  $r$  not synthetic then ▷ Synthetic constituent starts with '@'
5:          $prob[r.(end - start)][r.start][r] += p(\hat{y})$ 
6:       end if
7:     end for
8:   end for
9:
10:  for  $i = 0$  to  $(\#(words) - 1)$  do ▷ For each lexicon rule
11:     $w = words[i]$ 
12:    for  $r = (s \rightarrow w) \in \text{grammar}$  do
13:       $best[0][i][s] = prob[0][i][r] - \psi(r)$ 
14:    end for
15:  end for
16:
17:  for  $i = 1$  to  $(\#(words) - 1)$  do
18:    for  $j = 0$  to  $(\#(words) - i - 1)$  do
19:      for  $k = 0$  to  $(i - 1)$  do ▷ For each binary split
20:        for  $r = (a \rightarrow bc) \in \text{grammar}$  do
21:           $lbest = best[k][j][b]$ 
22:           $rbest = best[i - k - 1][j + k + 1][c]$ 
23:           $score = prob[i][j][r] - \psi(r) + lbest + rbest$ 
24:          if  $score < best[i][j][a]$  then
25:             $best[i][j][a] = score$ 
26:             $back[i][j][a] = (i, j, k, b, c)$  ▷ Record binary back pointer
27:          end if
28:        end for
29:      end for
30:    end for
31:  end for
32:  return  $[\hat{y}^*, score^*] = \text{buildTree}(back, best)$ 
33: end procedure

```

CHAPTER 5

APPLICATIONS AND EXPERIMENTS

(This chapter contains and expands on materials originally presented in Wang, H., Xing, W., Asif, K., and Ziebart, B.: Adversarial prediction games for multivariate losses. In Advances in Neural Information Processing Systems, pages 2728-2736, 2015.)

In this chapter, we show the Information Retrieval and Natural Language Processing tasks that can benefit from directly optimizing their evaluation metrics that are discussed in the previous chapter. We apply our approach, *Multivariate Prediction Games (MPG)*, to the following five core IR/NLP tasks:

- *binary classification* with F_1 score as evaluation metric;
- *named-entity recognition* that incorporates multi-class linear-chain structure constraint and evaluated by F_1 score metric;
- *ranking of query results* which evaluated by *precision at k ($P@k$)* and *discounted cumulative gain (DCG)*;
- world alignment quality of *machine translation* evaluated by *alignment error rate (AER)*;
- *syntactic context-free grammar parsing* with *Hamming loss (accuracy)* as evaluation metric.

Experiments for those tasks are conducted on several widely used datasets to support our study. We also compare MPG with the well studied ERM methods in each following section.

5.1 Binary Classification with F_1 Score

Binary classification are the most common task in Information Retrieval, it decides whether a web page or a document should be retrieved, i.e., be classified as relevant to the user query. Our primary point of comparison is with structured support vector machines (SSVM)(21) to better understand the trade-offs between convexly approximating the loss function with the hinge loss versus adversarially approximating the training data using our approach. Besides, we also compare our approach with the traditional ERM algorithm, logistic regression (LR), which uses logarithmic loss as a convex approximation, to demonstrate the benefits from directly optimizing the desired performance measure, F_1 score.

We employ an optical recognition of handwritten digits (OPTDIGITS) dataset (60) (10 classes, 64 features, 3,823 training examples, 1,797 test examples), and an income prediction dataset ('a4a' ADULT¹, two classes, 123 features, 3,185 training examples, 29,376 test examples) (60). Following the same evaluation method used in (21) for OPTDIGITS, the multi-class dataset is converted into multiple binary datasets and we report the macro-average of the performance of all classes on test data. For OPTDIGITS/ADULT, we use a random 1/3 of the training data as a holdout validation data to select the ℓ_2 regularization parameter trade-off $C \in \{2^{-6}, 2^{-5}, \dots, 2^6\}$.

From the results in Table VIII, we see that our approach, MPG, works better than SSVM on both the OPTDIGITS and the ADULT dataset. The nature of the running time required for

¹ <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html>

TABLE VIII: F_1 score performances on OPTDIGITS and ADULT datasets.

F_1 score	OPTDIGITS	ADULT
<i>MPG</i>	0.920	0.697
<i>SSVM</i>	0.915	0.673
<i>LR</i>	0.914	0.639

validation and testing is very different for SSVM, which must find the maximizing set of variable assignments, and MPG, which must interactively construct a game and its equilibrium. Model validation and testing require ≈ 30 seconds for SSVM on the OPTDIGITS dataset and ≈ 3 seconds on the ADULT dataset, while requiring ≈ 1397 seconds and ≈ 252 seconds for MPG F-measure optimization. For the more difficult problem of maximizing the F-score of ADULT over 29,376 test examples, the MPG game becomes quite large and requires significantly more computational time. Though our MPG method is not as finely optimized as existing SSVM implementations, this difference in run times will remain as the game formulation is inherently more computationally demanding for difficult prediction tasks.

5.2 Named-entity Recognition with F_1 Score

As we discussed in the previous chapters, leveraging sequential structure improves performance in tasks like named-entity recognition (NER). Like in the example, “*University of Illinois at Chicago*” is an ORGANIZATION, while “*Chicago*” alone is a LOCATION, the dependencies among consecutive words are important information, and are previously captured by the CRF model. In the comparison to the CRF model against our MPG model with linear-chain constraint, we use the well known CoNLL-2003 English dataset (61). We consider each sentence as one sequence example, and create three different sizes of subsets from the CoNLL-2003 data for our experiments, to demonstrate the benefit that directly optimizing F_1 score can bring, with respect to the training data size. The first dataset contains the first 300 sentences from ‘train,’ 300 sentences from ‘testa,’ and 300 sentences from ‘testb.’ The second dataset contains 1000 ($\times 3$) sentences, and the last contains 3000 ($\times 3$) sentences. Features are extracted using implementations from Stanford NER,¹ with the first-order CRF configuration². The number of features in each dataset is 31979, 67513, and 166737 respectively. The other statistics of each dataset can be found in Table IX.

¹<http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/crf/CRFFeatureExporter.html>

²<http://nlp.stanford.edu/software/crf-faq.shtml>

TABLE IX: The statistics of three linear-chain datasets.

dataset	conll.300			conll.1000			conll.3000		
	train	testa	testb	train	testa	testb	train	testa	testb
max length	48	47	125	53	56	125	59	110	125
avg. length	16	12	14	13	15	13	15	16	15
PER	152	275	616	1151	1097	1165	2967	2998	2464
LOC	246	179	234	769	628	514	2134	1857	1772
ORG	200	221	90	574	536	746	1634	1926	1969
MISC	144	88	85	339	283	280	1149	1121	843
# features	31979			67513			166737		

We evaluate the F_1 score of MPG on each dataset, with the same evaluation method as Stanford’s CRF implementation ¹. The performances of both the CRF model and the MPG model can be found in Table X.

MPG with linear-chain constraint works better for all NER tags than CRF on the 300 sentences datasets. Even though MPG does not outperform CRF on ‘testa’ of ‘conll.3000’ set, it has F-score 68.37% on ‘testb’, which is better than CRF. However, we notice that the results show MPG does not work as well as CRF for ‘LOC’ tag on 1000, and 3000 sentences datasets. The results suggest that optimizing F_1 directly can reduce the need of using larger dataset.

¹<http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/AbstractSequenceClassifier.html>

TABLE X: F_1 scores of CRF and MPG on ‘testa’ and ‘testb’.

dataset		testa		testb	
		CRF	MPG	CRF	MPG
conll.300	PER	17.88%	48.00%	40.00%	68.86%
	LOC	30.94%	48.19%	52.10%	57.21%
	ORG	7.19%	31.54%	14.16%	17.85%
	MISC	16.22%	26.09%	30.43%	39.19%
conll.1000	PER	78.68%	84.80%	76.36%	82.48%
	LOC	80.00%	77.68%	76.65%	76.03%
	ORG	58.24%	59.86%	61.14%	62.93%
	MISC	62.30%	65.99%	59.28%	67.13%
conll.3000	PER	85.72%	87.44%	81.05%	83.20%
	LOC	85.74%	84.56%	82.80%	80.41%
	ORG	75.69%	73.44%	66.18%	68.37%
	MISC	78.82%	79.41%	70.05%	74.02%

5.3 Ranking with Precision at k and Discounted Cumulative Gain

With different best response implementations, we optimize two evaluation metrics directly for assessing qualities of ranking: precision@k (Section 4.2.3), and discounted cumulative gain (DCG, Section 4.2.4). For *precision@k*, we evaluate the performance of our approach against the comparison method, SSVMs, where k is half the number of positive examples (i.e. $k = \frac{1}{2}POS$). Note that the original SSVM’s implementation uses the restriction k during training, but not during testing. We modified the code by ordering SSVM’s prediction value for each test example, and select the top k predictions as positives, the rest are considered as negatives. We denote the original implementation as *SSVM*, and the modified version as *SSVM’*.

From the results in Table XI, we see that our approach, MPG, works slightly better than SSVM on the OPTDGITS dataset. For the ADULT dataset, MPG provides equivalent perfor-

TABLE XI: Precision@k score performances on OPTDIGITS and ADULT datasets.

Precision@k	OPTDIGITS	ADULT
<i>MPG</i>	0.990	0.805
<i>SSVM</i>	0.956	0.638
<i>SSVM'</i>	0.989	0.805

mance for *precision at k*. Model testing time required by MPG is within an order of magnitude (better for OPTDIGITS, worse for ADULT), i.e., ≈ 30 seconds for SSVM on the OPTDIGITS dataset, and ≈ 3 seconds on the ADULT dataset, while requiring ≈ 9 seconds and ≈ 25 seconds for MPG respectively.

For *discounted cumulative gain*, we compare the performance of our approach and comparison methods using five-fold cross validation on the MQ2007 dataset. We measure performance using Normalized DCG (NDCG), which divides the realized DCG by the maximum possible DCG for the dataset, based on a slightly different variant of DCG employed by LETOR4.0 (62): $DCG''(\hat{\sigma}, \mathbf{y}) = 2^{y_{\hat{\sigma}(1)}} - 1 + \sum_{i=2}^n \frac{2^{y_{\hat{\sigma}(i)} - 1}}{\log_2 i}$. The comparison methods are:

- RankSVM-Struct (63), part of SVM^{struct} which uses structured SVM to predict the rank;
- ListNet (64), a list-wise ranking algorithm employing cross entropy loss;
- AdaRank-NDCG (65), a boosting method using ‘weak rankers’ and data reweighing to achieve good NDCG performance;
- AdaRank-MAP uses Mean Average Precision (MAP) (10) rather than NDCG;
- RankBoost (66), which reduces ranking to binary classification problems on instance pairs.

TABLE XII: MQ2007 NDCG results.

Method	Mean NDCG
<i>MPG</i>	0.5220
<i>RankSVM</i>	0.4966
<i>ListNet</i>	0.4988
<i>AdaRank-NDCG</i>	0.4914
<i>AdaRank-MAP</i>	0.4891
<i>RankBoost</i>	0.5003

Table XII reports the NDCG@ k averaged over all values of k (between 1 and, on average 41) while Figure 8 reports the results for each value of k between 1 and 10. From this, we can see that our MPG approach provides better rankings on average than the baseline methods except when k is very small ($k = 1, 2$). In other words, the adversary focuses most of its effort in reducing the score received from the first item in the ranking, but at the expense of providing a better overall NDCG score for the ranking as a whole.

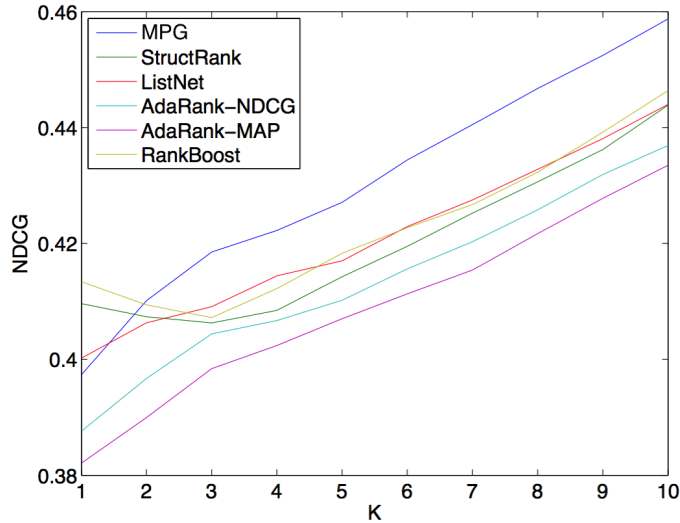


Figure 8: NDCG@k as k increases.

5.4 Machine Translation with Alignment Error Rate

We evaluate our approach against *maximum margin structured prediction model / SSVM* (14; 21) for alignment error rate. Since the maximum margin method’s implementation is not available, we implemented it ourselves following the algorithm description (30; 14; 67). Following Taskar et al.’s modeling (67) of the AER task as a maximum weighted bipartite matching problem (68) using large margin structured prediction, we build a bipartite best response module also for MPG (MPG_{bip}). The best responses can be computed using widely used maximum margin bipartite matching algorithms (69), by incorporating Lagrangian potentials as edge weights.

We use the NAACL 2003 Hansards data (70) for the AER task. It contains 1,470,000 unlabeled sentence pairs, 447 labeled pairs, and a separate validation set of 37 labeled pairs. We

experiment with translation from English to French, following the same setting as (14; 49). We use first 100 English-French sentence pairs from the original labeled data as training examples, the remaining 347 sentence pairs as test examples, and the same 37 validation pairs as validation examples. Since we can't get the features described in (14), we duplicate them with our best efforts ¹. We train the maximum margin structured model ($SSVM$), the MPG model with maximum weight bipartite matching (MPG_{bip}), and the MPG model for AER (MPG_{aer}) using those features extracted from the training dataset. Also, following the same setting of (14), we include GIZA++'s unsupervised prediction from the $1^5H^53^34^3$ training scheme (13) as an additional feature, and train another three models: $SSVM^{+GIZA}$, MPG_{bip}^{+GIZA} , MPG_{aer}^{+GIZA} . We select ℓ_2 regularization values for the MPG models based on performance on the validation dataset.

TABLE XIII: AER of different models.

Model	Valid	Test
$SSVM$	25.51%	22.13%
MPG_{bip}	31.12%	27.31%
MPG_{aer}	28.30%	26.13%
$SSVM^{+GIZA}$	13.98%	13.34%
MPG_{bip}^{+GIZA}	15.82%	14.52%
MPG_{aer}^{+GIZA}	6.97%	7.28%

¹Differences in $SSVM$ performance from (14) suggest that our features differ from those used previously.

The performances of models on validation and test datasets are listed in Table XIII. With our extracted features, the *SSVM* model performs better than the *MPG* models. When comparing MPG_{bip} against MPG_{aer} , it shows the advantage of modeling AER more directly over modeling the alignment problem as a maximum weight bipartite matching. After including GIZA’s prediction into the features, all models’ performances increase, and MPG_{bip}^{+GIZA} outperforms any other model, which demonstrates the effectiveness of MPG_{aer} when given high quality features to constrain the adversary.

5.5 Syntactic Context-free Grammar Parsing with Hamming loss

For the context-free grammar parsing task, the dataset we use is the Penn Treebank Wall Street Journal corpus (54). Following (21), we start based on the part-of-speech tags, and the number of times a certain rule occurs in the tree is used as the value of each feature. The comparison algorithm is SSVM for context-free grammar with Hamming loss ¹, and Stanford Parser ², which is a CYK algorithm that uses Markovization encoded grammar rules (58), and is also one of the state-of-art parsers available online. Using previous experimental methodology (21), we consider sentences of length at most 10 in the dataset. We use the standard section splits: sections 2-21 are our training set (4071 sentences), section 22 with 163 sentences is used as test set *testa*, and 270 sentences in section 23 are *testb*. Besides $(1 - \text{Hamming loss})$, we also report the Evalb F_1 (71) for each algorithm. The results are shown in Table XIV.

TABLE XIV: Results for CFG parsing on the Penn Treebank.

Model	1- Hamming		F_1	
	testa	testb	testa	testb
<i>SSVM</i>	71.94%	70.91%	78.27%	80.81%
<i>MPG_{cfg}</i>	85.77%	85.73%	86.53%	87.04%
<i>Stanford</i>	79.94%	85.27%	83.04%	87.02%

¹http://www.cs.cornell.edu/people/tj/svm_light/svm_cfg.html

²<http://nlp.stanford.edu/software/lex-parser.shtml>

We observe that MPG_{cfg} outperforms SSVM on both *testa* and *testb* with Hamming loss and F_1 score, perhaps benefiting from both the directly Hamming loss optimization and its guarantees of Fisher consistency (34). It also works better than Stanford parser on both the two datasets. With the ability of incorporating arbitrary features, we believe MPG_{cfg} can achieve better performances by carefully engineering rich features.

CHAPTER 6

CONCLUSION AND FUTURE PROSPECTS

6.1 Conclusion

In this thesis, we focus on solving the mismatch between Machine Learning algorithms’ optimization objective and application performance measures in Information Retrieval and Natural Language Processing, by building an adversarial prediction framework—*Multivariate Prediction Game (MPG)*—to approximate training data (adversarially) while optimizing the desired performance metrics directly. MPG treats the multivariate prediction as an adversarial zero-sum game between a loss-minimizing prediction player and a loss-maximizing evaluation player constrained to match specified properties of training data. By solving the problem of effectively finding the best responses to the opponent’s strategies, and applying the double oracle constraint generation method, the framework avoids the non-convexity of empirical risk minimization, and more directly optimizes the metrics. We summarize the main contributions of this thesis as follows:

- We reviewed several traditional empirical risk minimization (ERM) algorithms which use convex surrogates to make their optimization tractable. We pointed out the mismatch between such convex surrogates and the desired application performance measures, which motivates our study of this thesis.

- We proposed our Multivariate Prediction Game (MPG) framework that overcomes the mismatch by adversarially optimizing the performance metrics of interest directly. We gave very detailed descriptions to each component in MPG framework.
- We discussed the algorithms for finding the best response to opponent’s strategies for several commonly used metrics in NLP and IR tasks.
- By conducting a series of experiments and comparing against the widely used ERM algorithms (e.g., CRFs, SSVMs) on five core IR/NLP tasks with six datasets, we demonstrated the effectiveness of MPGs.


6.2 Future Prospects



There are many future studies can be pursued beyond this thesis’s work. A very recent work (72) proposes that for multivariate losses that satisfy a monotonicity property, and when the conditional probability of the positive class is given, optimal responses can be efficiently obtained. Following this direction, further studies on the multivariate losses satisfying such characteristics could be carried out to incorporate more metrics into the MPG framework. The study (73) proposed a double oracle variant algorithm for generating equilibrium. Additional experiments could be conducted after implementing this algorithm and plugging into MPG, to compare against current double oracle implementation. Currently, MPG supports only linear-chain structure when optimizing F_1 score, and when it is applied to context-free grammar parsing, the Hamming loss function is used. Possible future extensions can be either integrating more complex structure for F_1 optimization, or using more sophisticated metrics, like F_1 score, for CFG parsing.


APPENDICES

Appendix A

No permission is needed for the copyright of previous published work (1).




 Login



Please check the [Frequently Asked Questions](#) list to see if your question has already been answered.

nips.cc technical support: info@nips.cc

Paper submission, review, and CMT questions: program-chairs@nips.cc

Meeting and registration questions: info@nips.cc




Online proceeding questions: romangarnett@gmail.com




Follow NIPS on twitter: [@NipsConference](https://twitter.com/NipsConference)

Address

Neural Information Processing Systems Foundation
 10010 North Torrey Pines Road
 La Jolla, CA 92037

Phone: 858-453-1623
 Fax: 858-587-0417

Regarding nips published paper copyright  Inbox x  





Hong Wang <hwang207@uic.edu> 10/6/16 ☆  

to [romangarnett](mailto:romangarnett@gmail.com)

Hi NIPS,

I am the author of paper "Adversarial Prediction Games for Multivariate Losses", which was accepted by NIPS 2015. Right now, I am writing my PhD dissertation, and I am including the content of that paper into the dissertation. I am wondering is there any copyright issue that I need to worry about?

Thanks very much!


Roman Garnett <romangarnett@gmail.com> 10/6/16 ☆  

to [Hong](mailto:hwang207@uic.edu)

Hi Hong,

Absolutely not. You own the copyright to that paper; you only gave NIPS non-exclusive permission to distribute the paper. You are free to do as you like with the content!

Cheers,
 Roman

CITED LITERATURE

1. Wang, H., Xing, W., Asif, K., and Ziebart, B.: Adversarial prediction games for multivariate losses. In Advances in Neural Information Processing Systems, pages 2728–2736, 2015.
2. Asif, K., Xing, W., Behpour, S., and Ziebart, B. D.: Adversarial cost-sensitive classification. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, 2015.
3. Hoffgen, K.-U., Simon, H.-U., and Vanhorn, K. S.: Robust trainability of single neurons. Journal of Computer and System Sciences, 50(1):114–125, 1995.
4. Musicant, D. R., Kumar, V., and Ozgur, A.: Optimizing F-measure with support vector machines. In FLAIRS Conference, pages 356–360, 2003.
5. Jansche, M.: Maximum expected F-measure training of logistic regression models. In Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 692–699. Association for Computational Linguistics, 2005.
6. Parambath, S. P., Usunier, N., and Grandvalet, Y.: Optimizing F-measures by cost-sensitive classification. In Advances in Neural Information Processing Systems, pages 2123–2131, 2014.
7. Joachims, T.: A support vector method for multivariate performance measures. In Proceedings of the International Conference on Machine Learning, pages 377–384. ACM, 2005.
8. Ranjbar, M., Mori, G., and Wang, Y.: Optimizing complex loss functions in structured prediction. In Proceedings of the European Conference on Computer Vision, pages 580–593. Springer, 2010.
9. Cortes, C. and Mohri, M.: AUC optimization vs. error rate minimization. In Advances in Neural Information Processing Systems, pages 313–320, 2004.
10. Manning, C. D., Raghavan, P., Schütze, H., et al.: Introduction to information retrieval, volume 1. Cambridge university press Cambridge, 2008.

11. Manning, C. D. and Schütze, H.: Foundations of statistical natural language processing, volume 999. MIT Press, 1999.
12. Lafferty, J., McCallum, A., and Pereira, F. C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.
13. Och, F. J. and Ney, H.: A systematic comparison of various statistical alignment models. Computational linguistics, 29(1):19–51, 2003.
14. Taskar, B., Lacoste-Julien, S., and Klein, D.: A discriminative matching approach to word alignment. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 73–80. Association for Computational Linguistics, 2005.
15. Jurafsky, D. and Martin, J. H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR, 2 edition, 2008.
16. Finkel, J. R., Kleeman, A., and Manning, C. D.: Efficient, feature-based, conditional random field parsing. In ACL, volume 46, pages 959–967, 2008.
17. Haghighi, A., Blitzer, J., DeNero, J., and Klein, D.: Better word alignments with supervised itg models. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, pages 923–931. Association for Computational Linguistics, 2009.
18. Wang, M., Che, W., and Manning, C. D.: Joint word alignment and bilingual named entity recognition using dual decomposition. In ACL (1), pages 1073–1082, 2013.
19. Durrett, G. and Klein, D.: Neural crf parsing. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 302–312, 2015.
20. Cortes, C. and Vapnik, V.: Support-vector networks. Machine learning, 20(3):273–297, 1995.

21. Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In Proceedings of the International Conference on Machine Learning, page 104. ACM, 2004.
22. Liu, Y.: Fisher consistency of multicategory support vector machines. In International Conference on Artificial Intelligence and Statistics, pages 291–298, 2007.
23. Dalvi, N., Domingos, P., Sanghai, S., Verma, D., et al.: Adversarial classification. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 99–108. ACM, 2004.
24. Vapnik, V.: Principles of risk minimization for learning theory. In Advances in Neural Information Processing Systems, pages 831–838, 1992.
25. Grünwald, P. D. and Dawid, A. P.: Game theory, maximum entropy, minimum discrepancy, and robust Bayesian decision theory. Annals of Statistics, 32:1367–1433, 2004.
26. Sutton, C. and McCallum, A.: An introduction to conditional random fields. Machine Learning, 4(4):267–373, 2011.
27. Boyd, S. and Vandenberghe, L.: Convex optimization. Cambridge university press, 2004.
28. Murphy, K. P.: Machine learning: a probabilistic perspective. MIT press, 2012.
29. Burges, C. J.: A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery, 2(2):121–167, 1998.
30. Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C.: Learning structured prediction models: A large margin approach. In Proceedings of the International Conference on Machine Learning, pages 896–903. ACM, 2005.
31. von Neumann, J. and Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, 1947.
32. Ferguson, T. S.: Game theory, second edition. 2014.
33. Fathony, R., Liu, A., Asif, K., and Ziebart, B.: Adversarial multiclass classification: A risk minimization perspective. In Advances In Neural Information Processing Systems, pages 559–567, 2016.

34. Li, J., Asif, K., Wang, H., Ziebart, B. D., and Berger-Wolf, T.: Adversarial sequence tagging. 2016.
35. Shi, Q., Reid, M., Caetano, T., Van den Hengel, A., and Wang, Z.: A hybrid loss for multiclass and structured prediction. IEEE transactions on pattern analysis and machine intelligence, 37(1):2–12, 2015.
36. Hazan, T., Keshet, J., and McAllester, D. A.: Direct loss minimization for structured prediction. In Advances in Neural Information Processing Systems, pages 1594–1602, 2010.
37. Topsøe, F.: Information theoretical optimization techniques. Kybernetika, 15(1):8–27, 1979.
38. McMahan, H. B., Gordon, G. J., and Blum, A.: Planning in the presence of cost functions controlled by an adversary. In Proceedings of the International Conference on Machine Learning, pages 536–543, 2003.
39. Dembczynski, K. J., Waegeman, W., Cheng, W., and Hüllermeier, E.: An exact algorithm for F-measure maximization. In Advances in Neural Information Processing Systems, pages 1404–1412, 2011.
40. Zeiler, M. D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.
41. Berkelaar, M., Eikland, K., Notebaert, P., et al.: lpsolve: Open source (mixed-integer) linear programming system. Eindhoven U. of Technology, 2004.
42. Optimization, G.: Gurobi optimizer reference manual version 5.6, 2014.
43. Gilpin, A., Peña, J., and Sandholm, T.: First-order algorithm with $o(\ln(1/\epsilon))$ convergence for ϵ -equilibrium in two-person zero-sum games. In AAAI Conference on Artificial Intelligence, pages 75–82, 2008.
44. Liu, D. C. and Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical programming, 45(1-3):503–528, 1989.
45. Lewis, A. S. and Overton, M. L.: Nonsmooth optimization via BFGS. 2008.

46. Järvelin, K. and Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS), 20(4):422–446, 2002.
47. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G.: Learning to rank using gradient descent. In Proceedings of the 22nd international conference on Machine learning, pages 89–96. ACM, 2005.
48. Croft, W. B., Metzler, D., and Strohmann, T.: Search engines. Pearson Education, 2010.
49. Cherry, C. and Lin, D.: Soft syntactic constraints for word alignment through discriminative training. In Proceedings of the COLING/ACL on Main conference poster sessions, pages 105–112. Association for Computational Linguistics, 2006.
50. Dyer, C., Chahuneau, V., and Smith, N. A.: A simple, fast, and effective reparameterization of ibm model 2. Association for Computational Linguistics, 2013.
51. Kociský, T., Hermann, K. M., and Blunsom, P.: Learning bilingual word representations by marginalizing alignments. pages 224–229, 2014.
52. Och, F. J. and Ney, H.: Improved statistical alignment models. In Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pages 440–447. Association for Computational Linguistics, 2000.
53. Finkel, J. R., Grenager, T., and Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pages 363–370. Association for Computational Linguistics, 2005.
54. Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B.: Building a large annotated corpus of english: The penn treebank. Computational linguistics, 19(2):313–330, 1993.
55. Younger, D. H.: Recognition and parsing of context-free languages in time n^3 . Information and control, 10(2):189–208, 1967.
56. Wang, W., Knight, K., and Marcu, D.: Binarizing syntax trees to improve syntax-based machine translation accuracy. In EMNLP-CoNLL, pages 746–754. Citeseer, 2007.
57. Chomsky, N.: On certain formal properties of grammars. Information and control, 2(2):137–167, 1959.

58. Klein, D. and Manning, C. D.: Accurate unlexicalized parsing. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1, pages 423–430. Association for Computational Linguistics, 2003.
59. Klein, D. and Manning, C. D.: Parsing with treebank grammars: Empirical bounds, theoretical models, and the structure of the penn treebank. In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, pages 338–345. Association for Computational Linguistics, 2001.
60. Lichman, M.: UCI machine learning repository, 2013.
61. Tjong Kim Sang, E. F. and De Meulder, F.: Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, pages 142–147. Association for Computational Linguistics, 2003.
62. Qin, T. and Liu, T.-Y.: Introducing LETOR 4.0 datasets. arXiv preprint arXiv:1306.2597, 2013.
63. Joachims, T.: Optimizing search engines using clickthrough data. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, pages 133–142. ACM, 2002.
64. Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H.: Learning to rank: from pairwise approach to listwise approach. In Proceedings of the International Conference on Machine Learning, pages 129–136. ACM, 2007.
65. Xu, J. and Li, H.: Adarank: a boosting algorithm for information retrieval. In Proc. of the International Conference on Research and Development in Information Retrieval, pages 391–398. ACM, 2007.
66. Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y.: An efficient boosting algorithm for combining preferences. The Journal of machine learning research, 4:933–969, 2003.
67. Taskar, B., Lacoste-Julien, S., and Jordan, M.: Structured prediction via the extragradient method. In NIPS, pages 1345–1352, 2005.
68. Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C.: Introduction to algorithms, volume 6. MIT press Cambridge, 2001.

- 69. Lawler, E. L.: Combinatorial optimization: networks and matroids. Courier Corporation, 2001.
- 70. Mihalcea, R. and Pedersen, T.: An evaluation exercise for word alignment. In Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond-Volume 3, pages 1–10. Association for Computational Linguistics, 2003.
- 71. Sekine, S. and Collins, M.: Evalb bracket scoring program. URL: <http://www.cs.nyu.edu/cs/projects/proteus/evalb>, 1997.
- 72. Natarajan, N., Koyejo, O., Ravikumar, P., and Dhillon, I.: Optimal classification with multivariate losses. In Proceedings of The 33rd International Conference on Machine Learning, pages 1530–1538, 2016.
- 73. Zinkevich, M., Bowling, M., and Burch, N.: A new algorithm for generating equilibria in massive zero-sum games. In PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, volume 22, page 788. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

VITA

NAME: Hong Wang

EDUCATION:

University of Illinois at Chicago *Chicago, Illinois USA*
Ph.D. – Department of Computer Science *2017*

Nanjing University of Posts and Telecommunications *Nanjing, China*
B.E. – College of Optoelectronic Engineering *2008*

ACADEMIC EXPERIENCE:

University of Illinois at Chicago *Chicago, Illinois USA*
Research Assistant *2010 - 2016*
Teaching Assistant *2011 - 2016*

PUBLICATIONS:

A. Liu, **H. Wang**, B. D. Ziebart, *Robust Covariate Shift Classification Using Multiple Feature Views*, NIPS Reliable Machine Learning in the Wild workshop, Barcelona, Spain, December 2016.

J. Li, K. Asif, **H. Wang**, B. D. Ziebart, T. Berger-Wolf, *Adversarial Sequence Tagging*, International Joint Conference on Artificial Intelligence (IJCAI-16), New York, USA, July 2016.

H. Wang, W. Xing, K. Asif, B. D. Ziebart, *Adversarial Prediction Games for Multivariate Losses*, Advances in Neural Information Processing Systems (NIPS 2015), Montréal Canada, December 2015.

H. Wang, A. Liu, J. Wang, B. Ziebart, C. Yu, W. Shen, *Context Retrieval for Web Tables*, The ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR 2015), Northampton, Massachusetts, USA, September 2015.

E. Dragut, **H. Wang**, C. Yu, P. Sistla, W. Meng, *Polarity Consistency Checking for Domain Independent Sentiment Dictionaries*, IEEE Transactions on Knowledge and Data Engineering (TKDE), Volume 27, Issue 3, March 2015.

E. Dragut, **H. Wang**, C. Yu, P. Sistla, W. Meng, *Polarity Consistency Checking for Sentiment Dictionaries*, The 50th Annual Meeting of the Association for Computational Linguistics (ACL'12), Jeju, South Korea, July 2012.

INDUSTRIAL EXPERIENCE:

Google Inc. - Ads & Commerce *Mountain View, California USA*
Software Engineering Intern – Model Based Ads Retrieval *May 2015 - August 2015*

Google Inc. - Ads & Commerce *Kirkland, Washington USA*
Software Engineering Intern – DoubleClick Search *May 2014 - August 2014*

Google Inc. - Research *Mountain View, California USA*
Software Engineering Intern – Structured Data Group *May 2013 - August 2013*

Huawei Technologies Co. Ltd. *Shenzhen/Nanjing, China*
Software Engineer – Business Management Platform (B/OSS) *August 2008 - August 2010*