# Structured Learning for Information Retrieval

James Petterson

October 2011

A thesis submitted for the degree of Doctor of Philosophy
of the Australian National University

*To my parents.*

# Declaration

The work in this thesis is my own except where otherwise stated.

James Petterson

# Acknowledgements

First of all, I would like to thank the members of my committee, Tibério Caetano, Wray Buntine and Scott Sanner, for their advice and supervision. In particular, I would like to express my utmost gratitude to my supervisor and friend Tibério Caetano. When I decided, in 2007, to pursue an academic career after an hiatus of a decade working in the industry, he was the one that put all the effort to make this possible. In the last four years he has been not only my supervisor, but a great friend and my point of support for all matters, both academic and personal. I also thank him and his wife Camila for their warm welcome when I arrived in Australia in 2008.

I would also like to thank Alex Smola for trusting Tibério's recommendation and bringing me to NICTA as a visitor in 2008. That visit, and the job offer that followed a few months later, made this all possible. I also thank him for hosting me during my internship at Yahoo! and for all his advice during the collaborations we had during these years.

I would like to thank Rogerio Feris for hosting me for an internship at IBM Research, and for the advice and collaborations that followed. Working at IBM, applying machine learning to real-life problems, was an interesting experience.

A special thanks goes to all my friends and collaborators at NICTA, Yahoo! and IBM: Akshay Asthana, Marconi Barbosa, Debdeep Banerjee, Edwin Bonilla, Lisa Brown, Lan Du, Kishor Gawande, Ayman Ghoneim, Marcus Hutter, Cong Phuoc Huynh, Dmitry Kamenetsky, John Langford, Roslyn Lau, John Lim, Quoc Viet Le, Julian McAuley, Sachiko Miyazawa, Sharath Pankanti, Novi Quadrianto, Mark Reid, Avi Ruderman, Kristina Scherbaum, Qinfeng Shi, Behjat Siddiquie, Peter Sunehag, Choon Hui Teo, Owen Thomas, S. V. N. Vishwanathan, Markus Weimer, Chris Webers and Jin Yu. I hope I haven't forgotten anyone. In particular, I would like to thank Julian for his very careful and detailed revision of this thesis.

I thank NICTA and the Australian National University for funding my studies.

In particular, I would like to thank Bethany Flanders, Sue Van Haeften, Diane Kossatz, Jill Mayo, Michelle Moravec, Jonathan Peters, Debbie Pioch and Fiona Quinlan for helping me keep up with all the administrative issues.

Finally, I would like to thank my family. I own everything I am to my parents, Clarisse and Carlos. They have always been there to support me, and they have always been a model of what kind of person I want to be. I don't have words to express my gratitude to them. Last, but not least I would like to thank my wife Janine. She was the person most present during this venture. I am mostly grateful for her continuous support, optimism, cheerfulness and love during these years. Thank you, my sunshine.

# Abstract

Information retrieval is the area of study concerned with the process of searching, recovering and interpreting information from large amounts of data. In this Thesis we show that many of the problems in information retrieval consist of *structured learning*, where the goal is to learn predictors of complex output structures, consisting of many inter-dependent variables. We then attack these problems using principled machine learning methods that are specifically suited for such scenarios. In the process of doing so, we develop new models, new model extensions and new algorithms that, when integrated with existing methodology, comprise a new set of tools for solving a variety of information retrieval problems.

Firstly, we cover the multi-label classification problem, where we seek to predict a *set of labels* associated with a given object; the output in this case is structured, as the output variables are interdependent. Secondly, we focus on document ranking, where given a query and a set of documents associated with it we want to rank them according to their relevance with respect to the query; here, again, we have a structured output - a *ranking* of documents. Thirdly, we address topic models, where we are given a set of documents and attempt to find a compact representation of them, by learning latent topics and associating a topic distribution to each document; the output is again structured, consisting of word and topic *distributions*.

For all the above problems, we obtain state-of-the-art solutions as attested by empirical performance in publicly available real-world datasets.

# Contents

# Abbreviations

| | |
|---|---|
| AUC | Area Under Curve |
| BM | Binary Method |
| BMRM | Bundle Methods for Risk Minimization |
| C-CRF | Continuous Conditional Random Field |
| CC | Classifier Chains |
| CCA | Canonical Correlation Analysis |
| CDF | Cumulative Distribution Function |
| CDN | Cumulative Distribution Network |
| CVB | Collapsed Variational Bayesian |
| DAG | Directed Acyclic Graph |
| EBM | Ensemble of Binary Method |
| ECC | Ensemble of Classifier Chains |
| EM | Exponential Model |
| EP | Expectation Propagation |
| EPCC | Ensemble of Probabilistic Classifier Chains |
| EPS | Ensemble of Pruned Sets |
| IR | Information Retrieval |
| LDA | Latent Dirichlet Allocation |
| LMIR | Language Models for Information Retrieval |

| | |
|---|---|
| LSI | Latent Semantic Indexing |
| MAP | Maximum a Posteriori |
| ML | Maximum Likelihood |
| MLC | Multi-Label Classification |
| MM | Max-Margin |
| MS | Meta Stacking |
| NDCG | Normalised Discounted Cumulative Gain |
| OWA | Ordered Weighted Average |
| PCC | Probabilistic Classifier Chains |
| PLSI | Probabilistic Latent Semantic Indexing |
| PS | Pruned Sets |
| RAKEL | RAndom K-labELsets |
| RLSI | Regularised Latent Semantic Indexing |
| SVM | Support Vector Machine |
| VB | Variational Bayesian |

# List of Figures

# List of Tables

# Chapter 1

# Introduction

> Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it.
>
> Samuel Johnson, 1775.

With personal computers becoming popular in the late 1980s and the internet reaching critical mass in the early 1990s, we are now living in a world were information is no longer a scarce resource; in fact, the situation is quite the opposite: we are being overwhelmed by information. Here are some examples:

- more than 48 hours of video are added to YouTube every day [1];
- according to Netcraft, as of October of 2011 there exist more than 500 million websites, growing at a rate that exceeds 600 thousand new ones every day [2];
- the largest online encyclopaedia, Wikipedia, grows by more than one thousand articles a day [3].

As a consequence, the demand for efficient ways of searching for information has increased so much that Google, the company that hosts one of the most popular search engines, was the ninth US company by market value in 2010 [4]. Accordingly, the academic field of study concerned with the process of searching, recovering and interpreting information from large amounts of data – Information retrieval (IR) – has recently been the subject of intense research.[1]

In this thesis we will present new methods and algorithms for structured learning – that is, learning when the output is not a simple class, but a more complex

[1] See, for example, http://www.wikicfp.com/cfp/call?conference=information%20retrieval for a list of conferences dedicated to IR.

*structured* object. We divide the thesis in two parts: in the first part we will deal with structured learning in a supervised setting, where we are going to focus in multi-label classification and document ranking; in the second part we will address structured learning in an unsupervised setting, focusing on topic models. Although the methods differ in formulation and scope, they have in common the fact that they are all relevant to key problems in information retrieval, which we detail next:

**Multi-Label Classification (MLC)** In MLC we are given an object and the task is to predict a *set* of labels associated with it. A simple example is image annotation: we are given an image and we want to predict a set of tags associated with it [5]. The output is structured, since the variables in the set are interdependent – for example, in an image bicycles tend to co-occur with humans. MLC is a fundamental problem in IR [6], and its application to automatic image annotation [7] is vital to image retrieval. MLC is also loosely related to the relevance problem in IR, which deals with finding what classes a text is about, and has been subject of intense research [8, 9]. Finally, social networks provides us with massive MLC problems – some examples are tag recommendations in Flickr and link recomendations in LinkedIn, Twitter and Facebook[2]. In Chapters 3 and 4 we will propose new formulations and solutions to MLC problems.

**Document Ranking** In document ranking we are given a query and a set of documents associated with it, and the task is to rank them according to the relevance with respect to the query. Again, we have a structured output – in this case, a rank of the documents. This is highly pertinent to IR, as in many retrieval systems we want to rank results according to relevance. The significance of ranking to IR is evident by the number of workshops dedicated to learning to rank (See, for example, [10] and [11] for two recent examples.), as well as the continuing work on creating standard datasets such as LETOR [12] to facilitate research on the subject. In Chapter 5 we will propose a novel solution to the document ranking problem.

**Topic Modeling** Topic models are concerned with finding compact representations of documents by learning latent topics and associating each document with a topic distribution. The outputs of a topic model are highly structured, comprising interdependent word distributions for each topic and topic distributions for each document. Topic models have been an active area of research in IR for more than a decade. Some examples are the work of [13] on Latent Semantic

[2]Flickr: http://www.flickr.com/; LinkedIn: http://www.linkedin.com/; Twitter: http://twitter.com/; Facebook: http://www.facebook.com/.

Indexing (LSI) for information filtering; the successor of LSI, Probabilistic Latent Semantic Indexing (PLSI) [14]; and the recent work of [15] on Regularised Latent Semantic Indexing (RLSI), all of them published in IR conferences. Topic models are indeed the subject of whole chapters of textbooks on IR (see, for example, Chapter 18 of [6]). In Chapter 6 we will propose an extension to a well-known type of topic model, Latent Dirichlet Allocation (LDA).

## 1.1 Thesis Outline

The remainder of this thesis is organised into six chapters, whose content we summarise below. In Figure 1.1 we depict the dependencies between them.

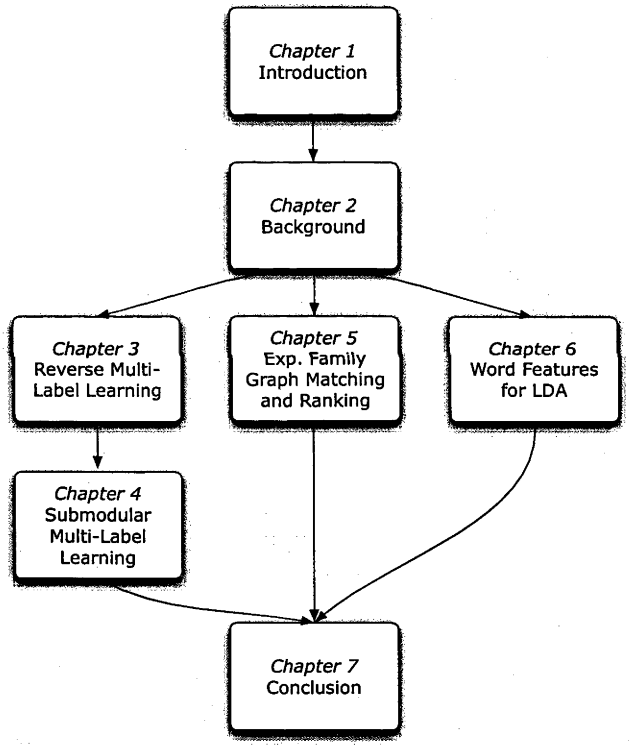



Figure 1.1: Outline of this thesis.

**Chapter 2: Background** In this chapter we will cover the background knowledge necessary to understand the subsequent chapters. We will first provide background on supervised structured learning, following with a description of two well-known estimators for it: max-margin with linear models and maximum a posteriori (MAP) with exponential families. The max-margin estimator shall be

the base of the work described in Chapters 3 and 4, while the MAP estimator will be applied in Chapter 5. Next we will provide background on unsupervised structured learning, where we will cover the very basics of graphical models, Gibbs sampling and topic models, all of which are needed to introduce LDA, an extension of which is the subject of Chapter 6.

**Part I: Supervised Structured Learning**

**Chapter 3: Reverse Multi-Label Learning** In this chapter we will apply the framework of max-margin structured output learning to the problem of learning a multi-label classifier. We will leverage this framework to directly optimise appropriate surrogates for a variety of performance measures commonly used in IR. We will apply our method to real-world datasets, and show that we can obtain state-of-the-art results.

**Chapter 4: Submodular Multi-Label Learning** In this chapter we will extend the work of Chapter 3 by encoding pairwise label interactions, so that we can leverage the co-ocurrence of pairs of labels in order to improve the quality of our predictions. Learning in this setting will become substantially more involved and will require the solution of an intractable combinatorial optimisation problem, so we will have to resort to an approximate algorithm. We will prove, however, that this algorithm is *sound*, in the sense that it never predicts incorrect labels. We will apply our method to real-world datasets, and show that we can obtain improvements over our previous methods.

**Chapter 5: Exponential Family Graph Matching and Ranking** In this chapter we will present a new estimator for learning graph matching predictors based on MAP estimation in conditional exponential families. We will apply it to the page ranking problem, with state-of-the-art results.

**Part II: Unsupervised Structured Learning**

**Chapter 6: Word Features for LDA** In this chapter we will extend LDA to allow for the encoding of side information in the distribution over words. This will result in a variety of new capabilities, such as improved estimates for infrequently occurring words, as well as the ability to leverage thesauri and dictionaries in order to boost topic cohesion within and across languages. This has direct application to IR, where the topic distributions generated by LDA can be used as a compact representation of a document.

**Chapter 7: Conclusion** In this chapter we will summarise the main results of this thesis, and discuss future directions of research.

## 1.2 Thesis Contributions

This thesis includes modelling and algorithmic contributions to the field of machine learning. The main contributions are:

1. Model and algorithms for generating multi-label predictors optimised for a variety of performance measures.

2. An extension of this model that allows for the encoding of pairwise label information. This includes: a constraint generation algorithm for loss-augmented inference where the scoring of the pair (input-output) is a submodular set function and the loss is derived from the $F$-score; an efficient algorithm which serves as a certificate of optimality for this constraint generation algorithm.

3. A new method for learning max-weight matching predictors in bipartite graphs using MAP estimation in exponential families.

4. An extension of LDA to allow for the encoding of side information in the distribution over words.

## 1.3 Publications

This thesis is based on the following published papers:

- James Petterson, Tiberio Caetano, Julian McAuley, and Jin Yu. Exponential family graph matching and ranking. In *Advances in Neural Information Processing Systems*, 2009.

- James Petterson, Alex Smola, Tiberio Caetano, Wray Buntine, and Shravan Narayanamurthy. Word features for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, 2010.

- James Petterson and Tiberio Caetano. Reverse multi-label learning. In *Advances in Neural Information Processing Systems*, 2010.

- James Petterson and Tiberio Caetano. Submodular multi-label learning. In *Advances in Neural Information Processing Systems*, 2011.

The following publications relate to other areas of machine learning and were not included in the thesis for consistency reasons:

- Rogerio Feris, Behjat Siddiquie, Yun Zhai, James Petterson, Lisa Brown, and Sharath Pankanti. Attribute-based vehicle search in crowded surveillance videos. In *ACM International Conference on Multimedia Retrieval*, 2011.

- Rogerio Feris, James Petterson, Behjat Siddiquie, Lisa Brown, and Sharath Pankanti. Large-scale vehicle detection in challenging urban surveillance environments. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, 2011.

- Novi Quadrianto, Alex Smola, Tiberio Caetano, S.V.N. Vishwanathan, and James Petterson. Multitask learning without label correspondences. In *Advances in Neural Information Processing Systems*, 2010.

- Novi Quadrianto, James Petterson, and Alex Smola. Distribution matching for transduction. In *Advances in Neural Information Processing Systems*, 2009.

- Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola and S.V.N. Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research - Special Topic on Large Scale Learning*, 2009.

- Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, Alex Strehl, and S. V. N. Vishwanathan. Hash kernels. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*. 2009.

# Chapter 2

# Background

In this chapter we will introduce the essential background knowledge that is necessary for the development of the subsequent chapters.

In the first half of this chapter we will provide some background on supervised structured learning, and describe two generic estimation strategies for it: one based in max-margin estimators, and another in MAP estimators in exponential families. These are going to be the basis for the work described in Chapters 3, 4 and 5.

In the second half of this chapter we will introduce the unsupervised structured learning problem. We will briefly review graphical models, a tool for dealing with complex probabilistic models, and Gibbs sampling, an algorithm that can be applied to generate samples from a probability distribution described by a graphical model. We will then focus on a particular case of an unsupervised structured learning problem: topic models. Finally, we will describe a popular type of topic model, Latent Dirichlet Allocation, an extension of which is the subject of Chapter 6.

## 2.1 Supervised Structured Learning

In a *supervised* learning setting we are given a set of training examples, each consisting of an input object and an output label. The goal is then to train a predictor so as to minimise a specified loss that is a function of the predicted labels and the true labels. As we will see later, in the case of max-margin estimators this loss is usually based on a score that reflects that evaluation measure we want to optimise for, while in the case of MAP estimators in exponential families the loss is normally based on the likelihood of the underlying probabilistic model.

In a typical supervised learning setting we are interested in learning a mapping from an input feature space $x \in \mathbb{R}^d$ to an output space with small cardinality – for example, $y \in \{1, \ldots, K\}$, for small $K$, in the case of classification. In certain application domains, however, the output space is more complex: in multi-label learning, for example, it is a set of mutually dependent variables; in natural language parsing, it is a parse tree. Supervised learning in these settings, where the outputs can be arbitrarily complex *structured objects*, is commonly denominated *supervised structured learning*, or *structured output learning*.

One simple approach to deal with these problems is to assign a label $k \in \{1, \ldots, K\}$ to each possible structured output and treat the resulting learning problem as an instance of multi-class learning. Quite often, however, this is not feasible, as the number of labels $K$ becomes extremely large. In multi-label learning, for example, where the output is a set of labels $y \in \mathcal{Y}$ from a dictionary of $V$ possible labels, $K$ is $2^V$, which becomes impractical for even small values of $V$. Clearly there is a need for a better approach.

In the following sections we describe two generic estimation strategies for producing structured output predictors. One is based on max-margin estimators [16, 17] and the other on maximum-likelihood (ML) or MAP estimators in exponential family models [18].

### 2.1.1 Max-Margin Estimators

The first approach we are going to describe is a generalisation of support vector machines (SVMs) to structured outputs, popularised by [16].

Let $x \in \mathcal{X}$ be an input and $y \in \mathcal{Y}$ a discrete output, where $\mathcal{Y}$ is a *structured* output space. We are given a training set $\{(x^n, y^n)\}_{n=1}^N$, and our task is to estimate a mapping $f : \mathcal{X} \to \mathcal{Y}$ that has good agreement with the training set but also generalises well to new data.

We assume the training data is generated i.i.d. from an unknown distribution $P$ on $\mathcal{X} \times \mathcal{Y}$, and that we are given a class of predictors $\mathcal{F}$ from which the predictor $f$ will be picked. We also assume we have a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_+$, which quantifies the loss associated with a prediction given the true output value.

In a *risk minimisation* setting we would like to choose $f$ so as to minimise the expected loss with respect to $P$ (i.e., the generalisation error):

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \underbrace{\int \Delta(y, f(x)) dP(x, y)}_{\textbf{generalisation error}} . \tag{2.1}$$

However, since we don't know $P$, we need an induction principle. We assume the principle of *regularised risk minimisation* which consists of solving

$$\underset{f \in \mathcal{F}}{\operatorname{argmin}} \underbrace{\frac{1}{N} \sum_{n} \Delta(f(x^n), y^n)}_{\textbf{empirical risk}} + \underbrace{\Omega(f)}_{\textbf{regulariser}} . \tag{2.2}$$

We choose a class of predictors $\mathcal{F}$ parametrized by a vector $\theta$ that, for a given input $x$, returns the maximiser of a linear score of $\theta$ and some arbitrary encoding of the data $\phi(x, y)$:

$$f_\theta(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \langle \phi(x, y), \theta \rangle . \tag{2.3}$$

With a regulariser $\Omega(\theta) = \frac{\lambda}{2} \|\theta\|^2$ to penalise complex solutions (where $\lambda$ controls the amount of regularisation), we have an *ideal* estimator taking the form

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \left[ \left( \frac{1}{N} \sum_{n=1}^{N} \Delta(\bar{y}^n(x^n; \theta), y^n) \right) + \frac{\lambda}{2} \|\theta\|^2 \right] . \tag{2.4}$$

where $\bar{y}^n$ is our prediction for the instance $n$:

$$\bar{y}^n = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \langle \phi(x^n, y^n), \theta \rangle . \tag{2.5}$$

That is, we want to find a model that minimises the average prediction loss in the training set *plus* a quadratic regulariser that penalises complex solutions. Estimators of this type are known as regularised risk minimisers [19].

This optimisation problem, however, is non-convex. Even worse, the loss is a piecewise constant function of $\theta$, since there is a countable number of loss values but an uncountable number of parameter values, so there are large equivalence classes of parameter values that correspond to precisely the same loss (see Figure 2.1). A similar problem occurs when one aims at optimising a 0/1 loss in binary classification; in that case, a typical workaround consists of minimising a surrogate convex loss function which upper bounds the 0/1 loss, for example the hinge loss, what gives rise to support vector machines.

The approach proposed by [16] consists of optimising a convex upper bound

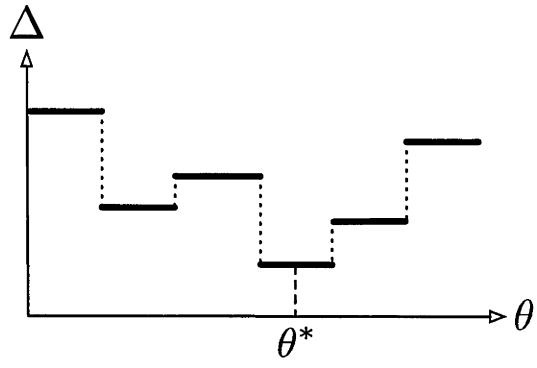



Figure 2.1: The loss in the optimisation problem of (2.4) is piecewise constant in $\theta$: there is a countable number of loss values (since the loss function is discrete) but an uncountable number of parameter values (since $\theta$ is continuous), so there are large equivalence classes of parameter values that correspond to precisely the same loss.

on the structured loss of (2.4)[1]:

$$[\theta^*, \xi^*] = \underset{\theta,\xi}{\operatorname{argmin}} \left[ \frac{1}{N} \sum_{n=1}^{N} \xi_n + \frac{\lambda}{2} \|\theta\|^2 \right] \tag{2.6a}$$

$$\text{s.t. } \langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y), \theta \rangle \geq \Delta(y, y^n) - \xi_n, \quad \xi_n \geq 0 \tag{2.6b}$$

$$\forall n, y \in \mathcal{Y}.$$

Intuitively, what this does is to enforce a loss-sensitive margin: $\theta$ is learned so that mispredictions $y$ that incur some loss end up with a score $\langle \phi(x^n, y), \theta \rangle$ that is smaller than the score $\langle \phi(x^n, y^n), \theta \rangle$ of the correct prediction $y^n$ by a margin equal to that loss (minus the slack $\xi$).

More formally, it can be seen that $\xi_n$ upper bounds $\Delta(\bar{y}^n, y^n)$, and therefore the objective in (2.6) upper bounds that of (2.4) for the optimal solution. First note that since the constraints (2.6b) hold for all $y$, they also hold for $\bar{y}^n$:

$$\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, \bar{y}^n), \theta \rangle \geq \Delta(\bar{y}^n, y^n) - \xi_n.$$

Second, the left hand side of this inequality must be non-positive from the definition of $\bar{y}$ in (2.5):

$$\underbrace{\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, \bar{y}^n), \theta \rangle}_{\leq 0} \geq \Delta(\bar{y}^n, y^n) - \xi_n.$$

[1]Here we are describing the *margin re-scaling* approach; an alternative is *slack re-scaling*, which we do not describe here as the concept is very similar; for details see [16].

It then follows that $\xi_n \geq \Delta(\bar{y}^n, y^n)$. The estimator is therefore well-motivated since it minimises an upper bound on the loss we really care about.

The optimisation problem (2.6), however, has $N|\mathcal{Y}|$ constraints, where $N$ is the number of training instances and $|\mathcal{Y}|$ is the cardinality of our output space, which can be extremely large. We therefore resort to a constraint generation strategy, which consists of starting with no constraints and iteratively adding the most violated constraint for the current solution of the optimisation problem. As shown by [16], such an approach is assured to find an $\epsilon$-close approximation of the solution of (2.6) after including only $O(\epsilon^{-2})$ constraints.

The key problem that needs to be solved at each iteration is *constraint generation*, i.e., to find the constraint that induces the maximum value of the violation margin $\xi_n$:

$$y_n^* \in \operatorname*{argmax}_{y \in \mathcal{Y}} \left[\Delta(y, y^n) + \langle \phi(x^n, y), \theta \rangle\right]. \tag{2.7}$$

This is similar to the optimisation problem that we need to solve for inference (2.5), with the difference being the additional term due to the loss. Note that (2.7) has to be solved $N$ times at each iteration of the learning algorithm, and (2.5) has to be solved for each instance during inference, so for this framework to be practical a suitable choice of $\phi(x, y)$ and $\Delta$ has to be made such that both (2.7) and (2.5) can be solved efficiently.

This max-margin structured learning framework is appealing in that it tries to minimise the loss in which we are interested. This is what we will discuss in Chapters 3 and 4, where we apply it to minimise losses relevant to IR. Its drawback, however, is that it is not a probabilistic method and therefore cannot be easily integrated into a larger system for purposes of Bayesian inference in decision-theoretic scenarios.

In the next section we describe an alternative strategy, maximum a posteriori estimation in exponential families, which is a fully probabilistic model that can easily be incorporated as a module in larger probabilistic models.

### 2.1.2 Exponential Family Estimators

Consider the same setting as before: $x \in \mathcal{X}$ is an input and $y \in \mathcal{Y}$ a discrete output, and we are given a training set $\{X, Y\} = \{(x^n, y^n)\}_{n=1}^N$. Now, however, instead of a generic map $f : \mathcal{X} \to \mathcal{Y}$, we are modelling a probability distribution: $p(y|x)$ – that is, given an input $x$ we want to estimate the probabilities of all possible outputs $y \in \mathcal{Y}$.

A well-known technique for this is to use ML or MAP estimation in conditional exponential families. Assuming an exponential family model, we have

$$p(y|x;\theta) = \frac{1}{Z(x;\theta)} \exp\left(\langle \phi(x,y), \theta \rangle\right). \tag{2.8}$$

Here, as before, $\phi(x,y)$ is an arbitrary encoding of the data, and in this case it also encodes the *sufficient statistics* of our distribution. $Z(x;\theta)$ ensures that (2.8) is normalised:

$$Z(x;\theta) = \sum_{y \in \mathcal{Y}} \exp \langle \phi(x,y), \theta \rangle. \tag{2.9}$$

Inference amounts to finding the most likely $y$, which has the same form as (2.5):

$$\bar{y}^n = \operatorname*{argmax}_{y \in \mathcal{Y}} p(y|x^n;\theta) = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \phi(x^n, y^n), \theta \rangle . \tag{2.10}$$

MAP estimation amounts to finding the $\theta$ that maximises the conditional likelihood of the training data (with a prior on $\theta$):

$$\theta^* = \operatorname*{argmax}_{\theta} p(Y|X;\theta)p(\theta) = \operatorname*{argmax}_{\theta} p(\theta|Y,X). \tag{2.11}$$

Assuming iid sampling, we have $p(Y|X;\theta) = \prod_{n=1}^{N} p(y^n|x^n;\theta)$. Therefore,

$$p(\theta|Y,X) \propto \exp\left( \log p(\theta) + \sum_{n=1}^{N} \left( \langle \phi(x^n, y^n), \theta \rangle - g(x^n;\theta) \right) \right) \tag{2.12}$$

where

$$g(x;\theta) = \log Z(x;\theta) = \log \sum_{y \in \mathcal{Y}} \exp \langle \phi(x,y), \theta \rangle \tag{2.13}$$

is the log-partition function, which is a convex and infinitely differentiable function of $\theta$ [20].

Instead of maximising the posterior we can equivalently minimise the negative log-posterior which, assuming a Gaussian prior on $\theta$, becomes:

$$\ell(Y|X;\theta) = \frac{\lambda}{2} \|\theta\|^2 + \frac{1}{N} \sum_{n=1}^{N} \left( g(x^n;\theta) - \langle \phi(x^n, y^n), \theta \rangle \right). \tag{2.14}$$

Here we have suppressed the constant terms, and $\lambda$ is a regularisation constant that controls the strength of our prior. ML estimation is essentially the same as

MAP estimation, but without the prior on $\theta$ – which is equivalent to setting $\lambda$ to zero in (2.14).[2]

Note that $\ell(Y|X;\theta)$ is a convex function of $\theta$, since the log-partition function $g(\theta)$ is a convex function of $\theta$ [20] and the other terms are clearly convex in $\theta$, so minimising it amounts to solving an unconstrained and convex optimisation problem. The key difficulty is the computation of the partition function $g(x;\theta)$, which often becomes intractable for structured outputs, since it is a sum over $|\mathcal{Y}|$ items, and $|\mathcal{Y}|$ can be extremely large.

This estimator, compared to the max-margin one, has the advantage of being a fully probabilistic model. To successfully apply it to structured outputs, however, the sufficient statistics must be carefully chosen, so as to keep both learning and inference tractable. In Chapter 5 we will use this estimator in a new learning algorithm for bipartite matching predictors.

## 2.2 Unsupervised Structured Learning

In the *supervised* learning setting just described we are given a set of training examples, where each example consists of an input object and an output label. In a *unsupervised* learning setting, however, we are not given labels, only objects. As a consequence, we can no longer classify examples, but we can still seek to summarise and explain the data.

One possible approach for summarising the data is clustering: assigning objects into groups so that objects in the same group are in some sense similar. Topic models, which we describe later, can be seen as a type of *soft* clustering, where each instance has a probability distribution on the clusters to which it belongs.

In the unsupervised setting we can no longer rely on a loss defined over labels, as we do not have them. One possible approach then is to define a *generative model* for the data, and maximise the likelihood of this model with respect to the data we observe. We will describe this in Section 2.2.3, but first we will briefly review graphical models and Gibbs sampling, as they are key to understanding what follows.

[2]In the particular case where we have a binary output this formulation is known as *logistic regression*; this can be seen by assuming $y \in \{-1, 1\}$ and choosing $\phi(x, y) = xy$.

### 2.2.1 Graphical Models

A graphical model [21] is a diagrammatic representation of a probabilistic model. It provides a way of visualising the structure of a model, while allowing the use of graphical manipulations to express the computations required to perform learning and inference (see, for example, belief propagation [22]).

A graph $G = (V, E)$ is composed of a set of vertices $V$ connected by edges $E$. In a graphical model each vertex represents a random variable, and edges represent the structure of conditional independences among the random variables. There are two main classes of graphical models: *directed graphical models*, also known as a *Bayesian networks*, in which the edges have directions, and *undirected graphical models*, or *Markov random fields*, in which the edges have no directional significance. Both types have its applications, but here we are going to focus on directed graphical models, as they are the basis of the LDA topic model, which is the model we are going to extend in chapter 6.

In a directed graphical model, the joint probability distribution of a set of random variables $x = (x_v)_{v \in V}$ can be written as:

$$p(x) = \prod_{v \in V} p(x_v | x_{pa(v)}) \tag{2.15}$$

where $pa(v)$ is the set of parents of $v$ – that is, those vertices pointing to $v$ via a single edge. As an example, consider the graph of Figure 2.2: this is a directed graphical model that encodes the structure of a probability distribution over four variables: $x_1$, $x_2$, $x_3$ and $x_4$. This graph represents the following factorisation of the joint distribution of these variables:

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_2, x_3) \tag{2.16}$$

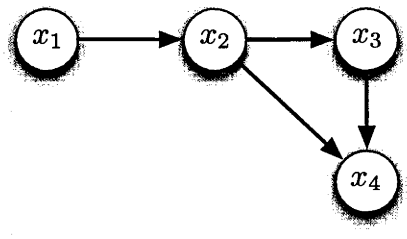



Figure 2.2: A simple graphical model.

Directed graphical models are subjected to one condition: there must be no *directed cycles*; in other words, there should be no path starting on a vertex,

following the edges in the direction pointed by the arrows and ending on the same vertex. Graphs that obey this restriction are also called *directed acyclic graphs* (DAGs).

### 2.2.2 Gibbs Sampling

Gibbs sampling [23] is a Markov chain Monte Carlo algorithm to generate a sequence of samples from the joint probability distribution of two or more variables. It is normally applied when it is difficult to sample from the joint distribution of all variables, but it is easy to sample from the distribution of each variable conditional on the values of the others. Using the graphical model of Figure 2.2 as an example, it could be the case that it is hard to sample directly from $p(x_1, x_2, x_3, x_4)$, but easy (or easier) to sample from the conditional distributions $p(x_1)$, $p(x_2|x_1)$, $p(x_3|x_2)$ and $p(x_4|x_2, x_3)$.

The algorithm works by assigning initial values for the variables (normally at random), and then iteratively replacing the value of each variable by sampling a value for it conditionally on the current values of all the remaining variables. This is repeated by cycling through all variables until some stopping condition is reached.

More formally, suppose we have a distribution over $N$ variables: $p(x_1, x_2, \ldots x_N)$. We start assigning random values for all of them. Then, at each step $t$, we cycle through all $x_i, i \in 1 \ldots N$ variables and replace $x_i^t$ with a sample from $p(x_i|x_{\neg i}^{t-1})$ (here $\neg i$ denotes all variables except variable $i$). It can be shown [24] that this sequence of samples constitutes a Markov chain, whose stationary distribution is the joint distribution $p(x_1, x_2, \ldots x_N)$. Therefore, in the limit when the number of iterations goes to infinity, the algorithm produces a sample from the correct distribution.

### 2.2.3 Topic Models

Topic models are a class of latent variable models for analysing the semantic content of a collection of documents. They were first introduced in the context of natural language processing, but have later found applications in several other fields, such as computer vision [25], finance [26], bioinformatics [27], music [28] and others. In particular, they are useful in IR [29], where they can be applied to create compact representations of documents while keeping most of the intra-document statistical structure.

In this section we will focus on what is arguably one of the most popular types of topic model: Latent Dirichlet Allocation.

### Latent Dirichlet Allocation

The key idea of LDA [30] is to model each document as a mixture over $K$ topics, with each topic $k$ coming from a multinomial distribution $\psi_k$ over a vocabulary of $V$ words. It is an *unsupervised* method, and its output, when applied to a corpus of documents, is

1. A topic for each word of each document.
2. A topic distribution for each document.
3. A word distribution for each topic.

The topic distribution for each document is particularly useful for IR: it can be seen as a compact representation of a document, and given an input document it makes it possible to look for similar/related documents by defining some distance measure in the space of topic distributions.

LDA's generative model works as follows (please refer to Table 2.1 for a summary of the notation used in this section):

- For a given document $m$ we first draw a topic distribution $\theta_m$ from a Dirichlet distribution parametrized by $\alpha$:

$$\theta_m \sim \text{Dir}(\alpha). \tag{2.17}$$

- Then, for each word $w_{mn}$ in the document we draw a topic $z_{mn}$ from a multinomial distribution with parameter $\theta_m$:

$$z_{mn} \sim \text{Mult}(\theta_m). \tag{2.18}$$

- Finally, we draw the word $w_{mn}$ from the multinomial distribution parametrized by $\psi_{z_{mn}}$:

$$w_{mn} \sim \text{Multi}(\psi_{z_{mn}}), \tag{2.19}$$

  which in turn comes from a Dirichlet distribution parametrized by $\beta$:

$$\psi_{z_{mn}} \sim \text{Dir}(\beta). \tag{2.20}$$

Table 2.1: Notation for LDA.

| variable | description |
|---|---|
| $K$ | number of topics |
| $M$ | number of documents |
| $N_m$ | number of words in document $m$ |
| $V$ | dictionary size |
| $\alpha$ | Dirichlet prior for $\theta$ (hyperparameter) |
| $\beta$ | Dirichlet prior for $\psi$ (hyperparameter) |
| $\theta$ | distribution of topics per document |
| $\psi$ | distribution of words per topic |
| $z_{mn}$ | topic $(1..K)$ of word $n$ of document $m$ |
| $w_{mn}$ | term index $(1..V)$ of word $n$ of document $m$ |
| $n_{kv}^{KV}$ | number of times the term $v$ has been observed with topic $k$ |
| $n_k^K$ | number of times topic $k$ has been observed in all documents |
| $n_{km}^{KM}$ | number of times topic $k$ has been observed in a word of document $m$ |

The graphical model representing this process is depicted in Figure 2.3, where we use plate notation: variables inside the plate are repeated according to the *for loop* in the bottom of it [31]. Note that the only observed variable in this model is $w$ (shaded in the figure); $\theta$, $z$ and $\psi$ are latent variables, and $\alpha$ and $\beta$ are hyperparameters. The joint probability distribution described by this graphical model (see Section 2.2.1) is:

$$\begin{aligned} p(w, z, \theta, \psi, \alpha, \beta) &= \prod_{k=1}^{K} p(\psi_k|\beta) \prod_{m=1}^{M} p(\theta_m|\alpha) \prod_{n=1}^{N_m} p(w_{mn}|\psi_{z_{mn}}) p(z_{mn}|\theta_m) \\ &= \prod_{k=1}^{K} Dir(\psi_k|\beta) \prod_{m=1}^{M} Dir(\theta_m|\alpha) \prod_{n=1}^{N_m} \psi_{z_{mn}, w_{mn}} \theta_{m, z_{mn}}. \end{aligned}$$

### Inference in LDA

The first inference method for LDA, proposed in the seminal paper of [30], was Variational Bayesian (VB), which made use of Jensen's inequality to obtain an adjustable lower bound on the log likelihood. Later other methods were suggested: Expectation Propagation (EP) [32], Collapsed Gibbs Sampling [33, 34] and Collapsed Variational Bayesian (CVB) inference [35], to cite a few. In this thesis we will focus on Collapsed Gibbs Sampling since it is relatively simple and, with recent advancements [36], also competitively fast.

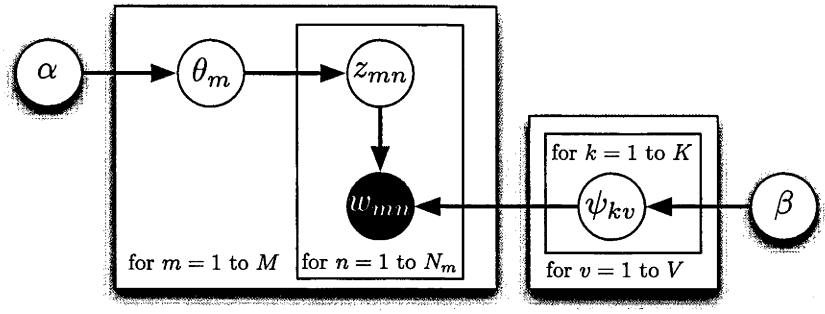



Figure 2.3: LDA model.

### Collapsed Gibbs sampling

Collapsed Gibbs sampling, initially proposed by [33], is an application of the Gibbs sampling algorithm described in Section 2.2.2, but with the latent variables ($\theta$ and $\psi$) integrated out. Since $\alpha$ and $\beta$ are hyperparameters, and $w$ is observed, what is left after the integration is $P(z_{mn}|z_{\neg mn})$: the distribution on the topic of a word of a document given the topics of all other words in the corpus.

The algorithm initially assigns random topics to all words. Then, at each step $t$, it iterates over each of the latent topic variables $z_{11}$, ..., $z_{mn}$, replacing $z^t_{mn}$ with a sample from $P(z_{mn}|z^{t-1}_{\neg mn})$. We are not going to dwell on the details here, since this has already been well explained in [33, 34], but in essence what we need to do is to sample from the distribution:

$$p(z_{mn} = k|z_{\neg mn}) \propto \frac{\left(n^{KV}_{kv-} + \beta\right)}{\left(n^{K}_{k-} + V\beta\right)} \left(n^{KM}_{km-} + \alpha\right). \tag{2.21}$$

In simple terms, to sample the topic of a word of a document given all the other words and topics we need, for each $k$ in $\{1, \ldots, K\}$:

1. $n^{KV}_{kv-}$: the total number of times the word has been observed with topic $k$ (the subscript "-" denotes that we are excluding from the statistics the word we are sampling for).

2. $n^{K}_{k-}$: the total number of times topic $k$ has been observed in all documents (excluding the word we are sampling for).

3. $n^{KM}_{km-}$: the number of times topic $k$ has been observed in a word of this document (excluding the word we are sampling for).

The main computation cost of this originates from the need of computing a normalisation constant $Z = \sum_{k=1}^{K} p(z_i = k|z_{\neg i}, w)$ to obtain a probability distribution that can be sampled from:

$$p(z_{mn} = k|z_{\neg mn}, w) = \frac{1}{Z} \frac{\left(n_{kv-}^{KV} + \beta\right)}{\left(n_{k-}^{K} + V\beta\right)} \left(n_{km-}^{KM} + \alpha\right). \tag{2.22}$$

Thus each iteration of collapsed Gibbs sampling has complexity $O(N_T K)$, where $N_T$ is the total number of words in the corpus, and $K$ is the number of topics.

Recently, however, [36] showed how to break (2.21) into three components and leverage the resulting sparsity in $k$. That, combined with an efficient storage scheme, leads to significant speed improvements on the sampling process. We are going to apply a similar approach to our extension of LDA that we describe in Chapter 6.

# Part I

# Supervised Structured Learning

# Chapter 3

# Reverse Multi-Label Learning

In this chapter we apply the max-margin estimator for structured outputs described in Section 2.1.1 to the problem of multi-label classification. In MLC we are concerned with classification problems where each instance is assigned to multiple labels – one example would be image annotation, where each image can have several tags associated with it. Unlike most existing work on the subject, the method we propose can be optimised for a variety of performance measures, in particular for the macro-measures widely used in IR: macro-$F_1$, macro-precision and macro-recall.

## 3.1 Introduction

In contrast to multi-class classification, where each instance is assigned a single label, in multi-label classification each instance can potentially have many labels. This reflects the situation in many real-world problems: in document classification, one document can cover multiple subjects [37]; in biology, a gene can be associated with a set of functional classes [38]; in image annotation, one image can have several tags [7].

As diverse as the applications are the evaluation measures used to assess the performance of different methods. This is understandable, since different applications have different goals. In e-discovery applications [39] it is mandatory that all relevant documents are retrieved, so recall is the most relevant measure. In web search, on the other hand, precision is also important, so the $F_1$-score, which is the harmonic mean of precision and recall, might be more appropriate.

In this chapter we present a method for MLC which is able to optimise appropriate surrogates for a variety of these measures. This means that the objective

function being optimised by the method is tailored to the performance measure on which we want to do well in our specific application.

Goal: To design a learning algorithm that produces multi-label classifiers optimised for a specific performance measure.

This is in contrast with probabilistic approaches, which typically aim for maximisation of likelihood scores rather than the performance measure used to assess the quality of the results. In addition, the method is based on well-understood facts from the domain of structured output learning (Section 2.1.1), which gives us theoretical guarantees[1] regarding the accuracy of the results obtained.

An interesting aspect of the method is that we are only able to optimise the desired performance measures because we formulate the prediction problem in a *reverse* manner, in the spirit of [40]. We pose the prediction problem as predicting sets of instances given the labels. When this insight is fit into max-margin structured output methods, we obtain surrogate losses for the most widely used performance measures for multi-label classification. We perform experiments against state-of-the-art methods in five publicly available benchmark datasets for MLC, and the proposed approach is the best performing overall.

### 3.1.1 Related Work

A straightforward way to deal with multiple labels is to solve a binary classification problem for each one of them, treating them independently. This approach is known as *Binary Method* (BM) [41]. The method of *Classifier Chains* (CC) [42] extends that by building a chain of binary classifiers, one for each possible label, but with each classifier augmented by all prior relevance predictions. Since the order of the classifiers in the chain is arbitrary, an ensemble method is also proposed – *Ensemble of Classifier Chains* (ECC) – where several random chains are combined with a voting scheme. The method of *Probabilistic Classifier Chains* (PCC) [43] extends CC to the probabilistic setting, with *Ensemble of Probabilistic Classifier Chains* (EPCC) [43] being its corresponding ensemble method.

Another way of working with multiple labels is to consider each possible set of labels as a class, thus encoding the problem as single-label classification. The problem with such an approach is the exponentially large number of classes, which implies a large computational cost, and the fact that most classes will have no

[1]With exact constraint generation we can achieve an approximation $\epsilon$-close to optimal in $O(\epsilon^{-2})$ iterations of Algorithm 1 (Theorem 18 of [16])

positive training instances. *RAndom K-labELsets* (RAKEL) [44] addresses this issue by proposing an ensemble of classifiers, each one taking a small random subset of the labels and learning a single-label classifier for the prediction of each element in the power set of this subset. Other proposed ensemble methods are *Ensemble of Binary Method* (EBM) [42], which applies a simple voting scheme to a set of BM classifiers, and *Ensemble of Pruned Sets* (EPS) [45], which combines a set of Pruned Sets (PS) classifiers. PS is essentially a problem transformation method that maps sets of labels to single labels while pruning away infrequently occurring sets.

In a different line of research *Canonical Correlation Analysis* (CCA) [46] exploits label relatedness by using a probabilistic interpretation of CCA as a dimensionality reduction technique and applying it to learn useful predictive features for multi-label learning. *Meta Stacking* (MS) [47] also exploits label relatedness by combining text features and features indicating relationships between classes in a discriminative framework.

Two papers closely related to our proposed method from the methodological point of view, which are however not tailored particularly to the multi-label learning problem, are [48] and [49]. [48] proposes a smooth but non-concave relaxation of the $F$-measure for binary classification problems using a logistic regression classifier, and optimisation is performed by taking the maximum across several runs of BFGS [50] starting from random initial values. [49] proposes a method for optimising multivariate performance measures in a general setting in which the loss function is not assumed to be additive in the instances nor in the labels. The method also consists of optimising a convex relaxation of the derived losses, but it is specialised for the case in which the loss can be computed from a contingency table (such as in micro-measures) and is restricted to binary labels.

After our research was published there has been continuing work in MLC, with two main lines of research being pursued. The first one is on ensemble techniques: [51] proposed a method for combining a set of multi label learners that are both accurate and diverse, involving a multi-objective optimisation problem with a correlation penalty term in the error function of each individual base learner to enforce them to be as different as possible on the training set; [52] proposed combining two groups of classifiers: one assuming independence between the labels and another considering a dense dependency structure.

The second line of research that has had increasing attention recently is the incorporation of label dependencies in the models [53]. In Chapter 4 we will describe our own work in this direction, along with recent related work.

Table 3.1: Notation used throughout this chapter.

| | |
|---|---|
| $\mathcal{X}$ | set of possible labels |
| $N$ | number of possible labels ($\|\mathcal{X}\|$) |
| $\mathcal{V}$ | set of possible instances |
| $V$ | number of possible instances ($\|\mathcal{V}\|$) |
| $\mathcal{Y}$ | set of outputs (power set of $\mathcal{V}$) $\mathcal{Y} = \{0,1\}^V$ |
| $D$ | dimensionality of instance features |
| $x$ | input label, $x \in \mathcal{X}$ ($N \times 1$ vector) |
| $y$ | set of instances, $y \in \mathcal{Y}$ ($V \times 1$ vector) |
| $\bar{y}$ | predicted set of instances |
| $\theta$ | parameter matrix ($D \times N$ matrix) |
| $\theta^n$ | the $n^{th}$ column of matrix $\theta$ |
| $\phi(x,y)$ | feature map: $\phi(x,y) = \sum_{v=1}^{V} y_v(\psi_v \otimes x)$ ($D \times N$ matrix) |
| $\psi_v$ | vector with feature representation for instance $v$ ($D \times 1$ vector) |
| $\Psi$ | a $V \times D$ matrix with row $v$ corresponding to $\psi_v$ |

## 3.2 The Model

Let the input $x \in \mathcal{X}$ denote a label (e.g., a tag of an image), and the output $y \in \mathcal{Y}$ denote a *set of instances*, (e.g., a set of training images). Let $N = |\mathcal{X}|$ be the number of labels and $V$ be the number of instances. An input label $x$ is encoded as $x \in \{0,1\}^N$, s.t. $\sum_i x_i = 1$. For example if $N = 5$ the second label is denoted as $x = [0\ 1\ 0\ 0\ 0]$. An output instance $y$ is encoded as $y \in \{0,1\}^V$ ($\mathcal{Y} := \{0,1\}^V$), and $y_i^n = 1$ if and only if instance $x^n$ was annotated with label $i$. For example if $V = 10$ and only instances 1 and 3 are annotated with label 2, then the $y$ corresponding to $x = [0\ 1\ 0\ 0\ 0]$ is $y = [1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$.

We assume a given training set $\{(x^n, y^n)\}_{n=1}^N$, where $\{x^n\}_{n=1}^N$ comprises the entire label space ($\{x^n\}_{n=1}^N = \mathcal{X}$), and $\{y^n\}_{n=1}^N$ represents the sets of instances associated with those labels. The task consists of estimating a map $f : \mathcal{X} \to \mathcal{Y}$ which accurately reproduces the outputs of the training set (i.e., $f(x^n) \approx y^n$) but also generalises well to new test instances. Please refer to Table 3.1 for the notation used throughout this chapter.

Note that our *input* is a label, and our *output* is a set of instances. $N$, the size of our training set, is the number of possible labels. $V$, the size of our output, is the number of instances. In other words, we pose our prediction problem as one of predicting the instances given the label.

### 3.2.1 Loss Functions

The reason for this *reverse prediction* is the following: most widely accepted performance measures target IR applications – that is, given a label we want to find a set of relevant instances. As a consequence, the measures are *averaged over the set of possible labels.* This is the case for, in particular, *Macro-precision*, *Macro-recall*, *Macro-$F_\beta$*[2] and the *Hamming loss* [44]:

$$\begin{aligned}
\text{Macro-precision} &= \frac{1}{N}\sum_{n=1}^{N} p(y^n, \bar{y}^n), \\
\text{Macro-recall} &= \frac{1}{N}\sum_{n=1}^{N} r(y^n, \bar{y}^n), \\
\text{Macro-F}_\beta &= \frac{1}{N}\sum_{n=1}^{N} (1+\beta^2)\frac{p(y^n, \bar{y}^n) r(y^n, \bar{y}^n)}{\beta^2 p(y^n, \bar{y}^n) + r(y^n, \bar{y}^n)}, \\
\text{Hamming loss} &= \frac{1}{N}\sum_{n=1}^{N} h(y^n, \bar{y}^n),
\end{aligned}$$

where

$$h(y, \bar{y}) = \frac{y^T \mathbf{1} + \bar{y}^T \mathbf{1} - 2y^T \bar{y}}{V}, \quad p(y, \bar{y}) = \frac{y^T \bar{y}}{\bar{y}^T \bar{y}}, \quad r(y, \bar{y}) = \frac{y^T \bar{y}}{y^T y}.$$

Here, $\bar{y}^n$ is our prediction for input label $n$, and $y^n$ the corresponding ground-truth. Since these measures average over the labels, in order to optimise them we need to average over the labels as well, and this happens naturally in a setting in which the empirical risk is additive on the labels.[3]

Instead of maximising a performance measure we frame the problem as minimising a loss function associated with the performance measure. We assume a given loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ which assigns a non-negative number to every possible pair of outputs. This loss function encodes how much we want to penalise a prediction $\bar{y}$ when the correct prediction is $y$, i.e., it has the opposite semantics of a performance measure. As already mentioned, we will be able to deal with a variety of loss functions in this framework, but for concreteness of exposition we will focus on a loss derived from the *Macro-$F_\beta$* score defined above,

[2]Macro-$F_1$ is the particular case of this when $\beta$ equals to 1. Macro-precision and macro-recall are particular cases of macro-$F_\beta$ for $\beta \to 0$ and $\beta \to \infty$, respectively.

[3]The Hamming loss also averages over the instances so it can be optimised in the 'normal' (not reverse) direction as well.

whose particular case for $\beta$ equal to 1 ($F_1$) is arguably the most popular performance measure for multi-label classification. In our notation, the $F_\beta$ score of a given prediction is

$$F_\beta(y, \bar{y}) = (1 + \beta^2)\frac{y^T \bar{y}}{\beta^2 y^T y + \bar{y}^T \bar{y}}, \tag{3.1}$$

and since $F_\beta$ is a measure of *alignment* between $y$ and $\bar{y}$, one possible choice for the loss is $\Delta(y, \bar{y}) = 1 - F_\beta(y, \bar{y})$, which is the one we focus on in this chapter,

$$\Delta(y, \bar{y}) = 1 - (1 + \beta^2)\frac{y^T \bar{y}}{\beta^2 y^T y + \bar{y}^T \bar{y}}. \tag{3.2}$$

### 3.2.2 Features and Parameterisation

Our next assumption is that the prediction for a given input $x$ returns the maximiser(s) of a linear score of the model parameter vector $\theta$, i.e., a prediction is given by $\bar{y}$ such that [4]

$$\bar{y} \in \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \phi(x, y), \theta \rangle . \tag{3.3}$$

Here we assume that $\phi(x, y)$ is linearly composed of features of the instances encoded in each $y_v$, i.e., $\phi(x, y) = \sum_{v=1}^{V} y_v(\psi_v \otimes x)$. The vector $\psi_v$ is the feature representation for the instance $v$. The map $\phi(x, y)$ will be the zero vector whenever $y_v = 0$, i.e., when the instance $v$ does not have label $x$. The feature map $\phi(x, y)$ has a total of $DN$ dimensions, where $D$ is the dimensionality of our features ($\psi_v$) and $N$ is the number of labels. Therefore $DN$ is the dimensionality of our parameter $\theta$ to be learned.

### 3.2.3 Optimisation Problem

We are now ready to formulate our estimator. We assume an initial, 'ideal' estimator taking the form

$$\theta^* = \operatorname*{argmin}_{\theta} \left[ \left( \frac{1}{N} \sum_{n=1}^{N} \Delta(\bar{y}^n(x^n; \theta), y^n) \right) + \frac{\lambda}{2} \|\theta\|^2 \right]. \tag{3.4}$$

In other words, we want to find a model that minimises the average prediction loss in the training set *plus* a quadratic regulariser that penalises complex solutions (the parameter $\lambda$ determines the trade-off between data fitting and good generalisation).

[4] $\langle A, B \rangle$ denotes the inner product of the vectorised versions of $A$ and $B$

## 3.3 Optimisation

### 3.3.1 Convex Relaxation

The optimisation problem (3.4) is non-convex – the loss is a piecewise constant function of $\theta$. We will therefore resort to a convex upper bound on the structured loss of (3.4). As already discussed in Section 2.1.1, the key problem that needs to be solved at each iteration of our learning algorithm is *constraint generation*, i.e., to find the maximiser of the violation margin $\xi_n$:

$$y_n^* \in \operatorname*{argmax}_{y \in \mathcal{Y}} \left[\Delta(y, y^n) + \langle \phi(x^n, y), \theta \rangle\right]. \tag{3.5}$$

The difficulty in solving the above optimisation problem depends on the choice of $\phi(x, y)$ and $\Delta$. Next we investigate how this problem can be solved for our particular choices of these functions.

### 3.3.2 Constraint Generation

Using (3.2) and $\phi(x, y) = \sum_{v=1}^{V} y_v(\psi_v \otimes x)$, (3.5) becomes

$$y_n^* \in \operatorname*{argmax}_{y \in \mathcal{Y}} \langle y, z_n \rangle . \tag{3.6}$$

where

$$z_n = \Psi\theta^n - \frac{(1+\beta^2)y^n}{\|y\|^2 + \beta^2 \|y^n\|^2}, \tag{3.7}$$

and

- $\Psi$ is a $V \times D$ matrix with row $v$ corresponding to $\psi_v$;
- $\theta^n$ is the $n^{th}$ column of matrix $\theta$;

We now investigate how to solve (3.6) for a fixed $\theta$. We first describe a simple, naïve algorithm. In the next section we then present a more involved but much faster algorithm.

A simple algorithm can be obtained by first noticing that $z_n$ depends on $y$ only through the number of its nonzero elements. Consider the set of all $y$ with precisely $k$ nonzero elements, i.e., $\mathcal{Y}_k =: \{y : \|y\|^2 = k\}$. Then the objective in (3.6), if the maximisation is instead restricted to the domain $\mathcal{Y}_k$, is effectively *linear* in $y$, since $z_n$ in this case is a constant with respect to $y$. Therefore we can

**Algorithm 1** Reverse Multi-Label Learning

**Input:** training set $\{(x^n, y^n)\}_{n=1}^N$, $\lambda$, $\beta$, **Output:** $\theta$
Initialise $i = 1$, $\theta_1 = 0$, MAX= $-\infty$
**repeat**
  **for** $n = 1$ **to** $N$ **do**
    Compute $y_n^*$ (Naïve: Algorithm 2. Improved: Algorithm 3)
  **end for**
  Compute subgradient $g_i$ (3.10) and objective $o_i$ (3.9)
  $\theta_{i+1} := \operatorname{argmin}_\theta \frac{\lambda}{2} \|\theta\|^2 + \max(0, \max_{j \leq i} \langle g_j, \theta \rangle + o_j)$; $i \leftarrow i + 1$
**until** converged (see [55])
**return** $\theta$

consider each $\mathcal{Y}_k$ separately by finding the top $k$ entries in $z_n$. Finding the top $k$ elements of a list of size $V$ can be done in $O(V)$ time [54]. Therefore we have an $O(V^2)$ algorithm (for every $k$ from 1 to $V$, solve $\operatorname{argmax}_{y \in \mathcal{Y}_k} \langle y, z \rangle$ in $O(V)$ time). Algorithm 1 describes in detail the optimisation problem, as solved by Bundle Methods for Risk Minimization (BMRM) [55], and Algorithm 2 shows the naïve constraint generation routine. The BMRM solver requires both the value of the objective function for the slack corresponding to the most violated constraint and its gradient. The value of the slack variable corresponding to $y_n^*$ is

$$\xi_n^* = \Delta(y_n^*, y^n) + \langle \phi(x^n, y_n^*), \theta \rangle - \langle \phi(x^n, y^n), \theta \rangle , \tag{3.8}$$

thus the objective function from (2.6) becomes

$$\frac{1}{N} \sum_n \Delta(y_n^*, y^n) + \langle \phi(x^n, y_n^*), \theta \rangle - \langle \phi(x^n, y^n), \theta \rangle + \frac{\lambda}{2} \|\theta\|^2 , \tag{3.9}$$

whose gradient (with respect to $\theta$) is

$$\lambda\theta - \frac{1}{N} \sum_n (\phi(x^n, y^n) - \phi(x^n, y_n^*)). \tag{3.10}$$

We need both expressions (3.9) and (3.10) in Algorithm 1.

### 3.3.3 Faster Constraint Generation

A faster constraint generation algorithm can be obtained by finding via binary search successively improved lower bounds on the cardinality $k$ of nonzero entries in any optimal solution $y^*$. Only once we cannot further improve this lower bound

**Algorithm 2** Naïve Constraint Generation

1: **Input:** $(x^n, y^n)$, $\Psi$, $\theta$, $\beta$, $V$, **Output:** $y_n^*$
2: MAX= $-\infty$
3: **for** $k = 1$ **to** $V$ **do**
4: $\quad z_n = \Psi\theta^n - \frac{(1+\beta^2)y^n}{k+\beta^2\|y^n\|^2}$
5: $\quad y^* = \text{argmax}_{y \in \mathcal{Y}_k} \langle y, z_n \rangle$ (i.e. find top $k$ entries in $z_n$ in $O(V)$ time)
6: $\quad$ CURRENT= $\max_{y \in \mathcal{Y}_k} \langle y, z_n \rangle$
7: $\quad$ **if** CURRENT>MAX **then**
8: $\quad\quad$ MAX = CURRENT
9: $\quad\quad y_n^* = y^*$
10: $\quad$ **end if**
11: **end for**
12: **return** $y_n^*$

do we resort to a linear search in $k$, just as Algorithm 2 does. Below we describe the algorithm, and afterwards an explanatory proof of its correctness. The worst case complexity is still $O(V^2)$ since a linear search is required at the end, however in practice the algorithm is very fast since many if not most of the cardinalities $k$ are never considered as they fall below the final lower bound.

**Theorem 1** *Algorithm 3 finds an optimal solution to* $\text{argmax}_{y \in \mathcal{Y}} \langle y, z_n \rangle$.

**Proof** There are two key ideas in the algorithm. The first is in lines 2-3. The subvector $z_{(0)}$ is constant, since it only depends on $(\Psi\theta)_{(0)}$. In particular, its own subvector obtained by restriction to the positive entries is also constant: $z_{(0+)}$. The idea is that the cardinality of $(0^+)$ is a lower bound on $k$, since removing a single 1 from $y^*_{(0+)} = \mathbf{1}$ has two effects: (i) it will necessarily decrease the inner product $\left\langle y^*_{(0+)}, z_{(0+)} \right\rangle$ since all entries of $z_{(0+)}$ are positive, and (ii) the remaining terms of the inner product $\langle y^*, z \rangle$ will either be decreased or kept constant. Therefore collectively the inner product $\langle y, z \rangle$ is decreased. This gives us a first lower bound. A succession of larger lower bounds can be obtained by incorporating a second idea, implemented in lines 6-18. We start with $k$ as the first lower bound, and compute the amount of positive entries $POS$ in the resulting $z$ evaluated at that particular $k$ (lines 9-10). The key insight now is that, if $POS > k$, then certainly $POS$ is also a lower bound. This is true because by going from $k$ to $POS$ we necessarily *increase* the entries in $z$, so the indices that were positive continue to be positive and by the same previous argument any $k < POS$ will

decrease the inner product. We then propose a new $k$ halfway between the new lower bound and a (previously computed in line 7) upper bound $UB^+$ on the *number of positive entries of any* $z$. This is done because we cannot expect that $POS > k$ if $k = UB^+$, so the test in line 11 (which gives us a better lower bound)

**Algorithm 3** Constraint Generation

1: **Input:** $(x, y)$, $\Psi$, $\theta$ **Output:** $y^*$ (index $n$ is left implicit)
2: Compute $I = \{i : y_i = 0\}$, and $z_{(0)}$, the subvector of $z$ restricted to $I$.
3: Compute the index set $(0^+)$ of the positive entries of $z_{(0)}$. Set $y^*_{(0^+)} = \mathbf{1}$.
4: $LB$ = cardinality of $(0^+)$
5: $k = LB$
6: Compute the index set $(V^+)$ of positive entries of $z = \Psi\theta - \frac{(1+\beta^2)y}{V+\beta^2\|y\|^2}$
7: $UB^+$ = cardinality of $(V^+)$ (upper bound on no. positive entries of $z$)
8: **repeat**
9: $z = \Psi\theta - \frac{(1+\beta^2)y}{k+\beta^2\|y\|^2}$
10: Compute POS = # of positive entries in $z$
11: **if** $k < POS$ **then**
12: LB = POS
13: $k = \lfloor (UB^+ + LB)/2 \rfloor$
14: **end if**
15: **if** $k > POS$ **then**
16: $k = \lfloor (k + LB)/2 \rfloor$
17: **end if**
18: **until** $k = LB$
19: (in the following for loop all computations can be restricted to the index
20: set $\mu = \{1, ..., V\} \backslash (0^+)$, since we know $y^*_{(0^+)} = \mathbf{1}$)
21: **for** $k = LB$ **to** V **do**
22: $z = \Psi\theta - \frac{(1+\beta^2)y}{k+\beta^2\|y\|^2}$
23: $y' = \text{argmax}_{y \in \mathcal{Y}_k} \langle y, z \rangle$ (i.e. find top $k$ entries in $z$ in $O(V)$ time)
24: CURRENT= $\max_{y \in \mathcal{Y}_k} \langle y, z \rangle$
25: **if** CURRENT>MAX **then**
26: MAX = CURRENT
27: $y^* = y'$
28: **end if**
29: **end for**
30: **return** $y^*$

will never be accepted for $k > UB^+$. If however $POS < k$, then we don't have a new lower bound, and we decrease the proposed $k$ halfway towards the current lower bound. The end of the binary search happens when the number of positive entries in $z$ agrees with the proposed $k$, i.e. when the lower bound cannot be further improved. The fact that this will necessarily happen after some point follows from the fact that the sequence of lower bounds is increasing and $UB^+$ is an upper bound on this sequence. Once the binary search has finished, we simply revert to the naïve version of the algorithm, as described in Algorithm 2, and search for $k$ between this best lower bound and $V$. ■

### 3.3.4 Prediction

The problem to be solved at test time (3.3) has the same form as the problem of constraint generation (3.5), the only difference being that $z_n = \Psi\theta^n$ (i.e., the second term in (3.7), due to the loss, is not present). Since $z_n$ is a constant vector, the solution $y_n^*$ for (3.5) is the vector that indicates the positive entries of $z_n$, which can be efficiently found in $O(V)$. Therefore inference at prediction time is very fast.

### 3.3.5 Other Scores

Up to now we have focused on optimising Macro-$F_\beta$, which already gives us several scores, in particular Macro-$F_1$, macro-recall and macro-precision. We can however optimise other scores, in particular the popular Hamming loss – Table 3.2 shows a list of scores with the corresponding loss, which we then substitute in (3.4).

Note that for the *Hamming loss* and *macro-recall* the denominator is constant, and therefore it is not necessary to solve (3.6) multiple times as described earlier, which makes constraint generation as fast as test-time prediction (see subsection 3.3.4).

## 3.4 Experimental Results

In this section we evaluate our method in several real world datasets, for both *macro-$F_\beta$* and *Hamming loss*. These scores were chosen because macro-$F_\beta$ is a

Table 3.2: Evaluation scores and corresponding losses.

| score | $\Delta(y, \bar{y})$ |
|---|---|
| macro-$F_\beta$ | $1 - \frac{(1+\beta^2)(y^T\bar{y})}{\beta^2 y^T y + \bar{y}^T\bar{y}}$ |
| macro-precision | $1 - \frac{y^T\bar{y}}{\bar{y}^T\bar{y}}$ |
| macro-recall | $1 - \frac{y^T\bar{y}}{y^T y}$ |
| Hamming loss | $\frac{y^T\mathbf{1} + \bar{y}^T\mathbf{1} - 2y^T\bar{y}}{V}$ |

generalisation of the most relevant scores, and the Hamming loss is a generic, popular score in the multi-label classification literature.

### 3.4.1 Datasets

We used 5 publicly available[5] multi-label datasets: *yeast*, *scene*, *medical*, *enron* and *emotions*. We selected these datasets because they cover a variety of application domains – biology, image, text and music – and there are published results of competing methods on them for some of the popular evaluation measures for MLC (*macro-$F_1$* and *Hamming loss*). Table 3.3 describes them in more detail.

### 3.4.2 Model Selection

Our model requires only one parameter: $\lambda$, the trade-off between data fitting and good generalisation. For each experiment we selected it with 5-fold cross-validation using only the training data.

### 3.4.3 Implementation

Our implementation is in C++, using the BMRM package of [55] as a base. Source code is available[6] under the Mozilla Public License.[7]

### 3.4.4 Comparison to Published Results on Macro-$F_1$

In our first set of experiments we compared our model to published results on the Macro-$F_1$ score. We strived to make our comparison as broad as possible, but

[5]http://mulan.sourceforge.net/datasets.html
[6]http://users.cecs.anu.edu.au/~jpetterson/.
[7]http://www.mozilla.org/MPL/MPL-1.1.html

Table 3.3: Datasets. #train/#test denotes the number of observations used for training and testing respectively; $N$ is the number of labels and $D$ the dimensionality of the features.

| dataset | domain | #train | #test | N | D |
|---|---|---|---|---|---|
| yeast | biology | 1500 | 917 | 14 | 103 |
| scene | image | 1211 | 1196 | 6 | 294 |
| medical | text | 645 | 333 | 45 | 1449 |
| enron | text | 1123 | 579 | 53 | 1001 |
| emotions | music | 391 | 202 | 6 | 72 |

Table 3.4: Macro-$F_1$ results. Bold face indicates the best performance. Results for CCA in the Medical and Enron datasets are unavailable.

| Dataset | Ours | CCA | CC | BM | SM | MS | ECC | EBM | EPS | RAKEL |
|---|---|---|---|---|---|---|---|---|---|---|
| **Yeast** | **0.440** | 0.346 | 0.346 | 0.326 | 0.327 | 0.331 | 0.362 | 0.364 | 0.420 | 0.413 |
| **Scene** | 0.671 | 0.374 | 0.696 | 0.685 | 0.666 | 0.694 | 0.742 | 0.729 | **0.763** | 0.750 |
| **Medical** | **0.420** | - | 0.377 | 0.364 | 0.321 | 0.370 | 0.386 | 0.382 | 0.324 | 0.377 |
| **Enron** | **0.243** | - | 0.198 | 0.197 | 0.144 | 0.198 | 0.201 | 0.201 | 0.155 | 0.206 |

we limited ourselves to methods with published results on public datasets, where the experimental setting was described in enough detail to allow us to make a fair comparison.

We therefore compared our model to Canonical Correlation Analysis [46] (CCA), Binary Method [41] (BM), Classifier Chains [42] (CC), Subset Mapping [56] (SM), Meta Stacking [47] (MS), Ensembles of Binary Method [42] (EBM) , Ensembles of Classifier Chains [42] (ECC), Ensembles of Pruned Sets [45] (EPS) and Random K Label Subsets [44] (RAKEL).

Table 3.4 summarises our results, along with those of competing methods which were taken from compilations by [46] and [42]. We can see that our model has the best performance in *yeast*, *medical* and *enron*. In *scene* it doesn't perform as well – we suspect this is related to the label cardinality of this dataset: almost all instances have just one label, making this essentially equivalent to a multiclass dataset.

Table 3.5: Hamming loss results. Bold face indicates the best performance.

| Dataset | Ours | CC | PCC | ECC | EPCC |
|---|---|---|---|---|---|
| Scene | **0.1271** | 0.1780 | 0.1780 | 0.1503 | 0.1498 |
| Emotions | **0.2252** | 0.2448 | 0.2417 | 0.2428 | 0.2372 |

### 3.4.5 Comparison to Published Results on Hamming Loss

To illustrate the flexibility of our model we also evaluated it on the Hamming loss. Here, we compared our model to classifier chains [42] (CC), probabilistic classifier chains [43] (PCC), ensembles of classifier chains [42] (ECC) and ensembled probabilistic classifier chains [43] (EPCC). These are the methods for which we could find Hamming loss results on publicly available data.

Table 3.5 summarises our results, along with competing methods' which were taken from a compilation by [43]. As can be seen, our model has the best performance on both datasets.

### 3.4.6 Results on $F_\beta$

One strength of our method is that it can be optimised for the specific measure we are interested in. In Macro-$F_\beta$, for example, $\beta$ is a trade-off between *precision* and *recall*: when $\beta \to 0$ we recover *precision*, and when $\beta \to \infty$ we get *recall*. Unlike previous methods, given a desired precision/recall trade-off encoded in a choice of $\beta$, we can optimise our model such that it gets the best performance on Macro-$F_\beta$. To show this we ran our method on all five datasets, but this time with different choices of $\beta$, ranging from $10^{-2}$ to $10^2$. In this case, however, we could not find published results to compare to, so we used Mulan,[8] an open-source library for learning from multi-label datasets, to train three models: BM[41], RAKEL[44] and MLKNN[57]. BM was chosen as a simple baseline, and RAKEL and MLKNN are the two state-of-the-art methods available in the package.

MLKNN has two parameters: the number of neighbours $k$ and a smoothing parameter $s$ controlling the strength of the uniform prior. We kept both fixed to 10 and 1.0, respectively, as was done in [57]. RAKEL has three parameters: the number of models $m$, the size of the label set $k$ and the threshold $t$. Since a complete search over the parameter space would be impractical, we adopted the library's default for $t$ and $m$ (respectively 0.5 and $2 * N$) and set $k$ to $\frac{N}{2}$ as suggested by [42]. For BM we kept the library's defaults.

[8] http://mulan.sourceforge.net/

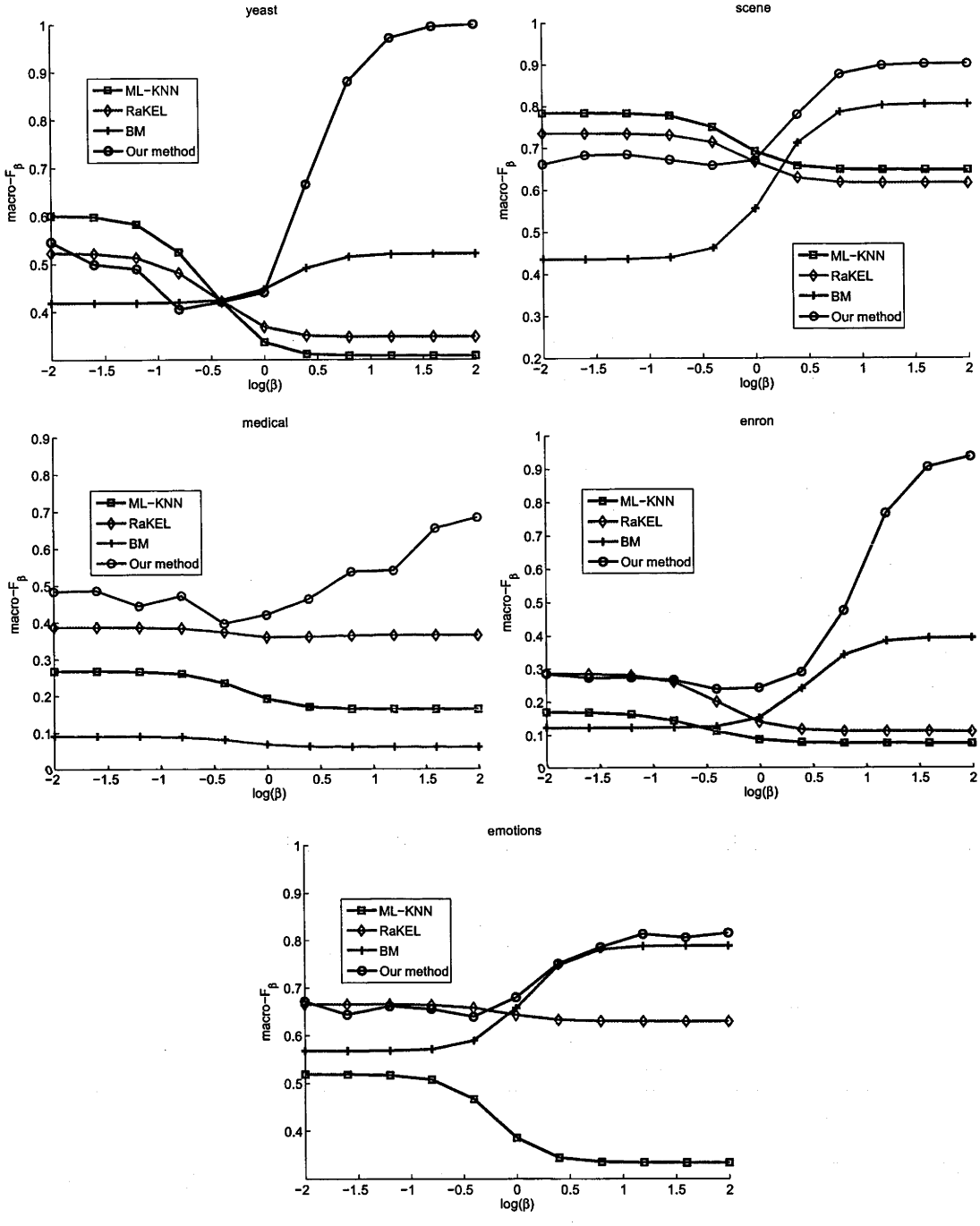



Figure 3.1: Macro-$F_\beta$ results on five datasets, with $\beta$ ranging from $10^{-2}$ to $10^2$ (i.e., $\log_{10}\beta$ ranging from -2 to 2). The centre point ($\log\beta = 0$) corresponds to macro-$F_1$. $\beta$ controls a trade-off between *Macro-precision* (left side) and *Macro-recall* (right side).

In Figure 3.1 we plot the results. We can see that BM tends to prioritise *recall* (right side of the plot), while ML-KNN and RAKEL give more emphasis to *precision* (left side). Our method, however, goes well in both sides, as it is trained separately for each value of $\beta$. In both *scene* and *yeast* it dominates the right side while it is still competitive on the left side. And in the other three datasets – *medical*, *enron* and *emotions* – it practically dominates over the entire range of $\beta$.

## 3.5 Summary

Multi-label classification is a key problem in several IR applications such as automatic image annotation [7] and document classification [37]. This chapter presented a new approach to this problem which consists of predicting sets of instances from labels. This apparently unintuitive approach is in fact natural since, once the problem is viewed from this perspective, many popular performance measures admit convex relaxations that can be directly and efficiently optimised with existing methods. The method only requires one parameter, as opposed to most existing methods, which have several. The method leverages existing tools from structured output learning, which gives us certain theoretical guarantees. A simple version of constraint generation is presented for small problems, but we also developed a scalable, fast version for dealing with large datasets. We presented a detailed experimental comparison against several state-of-the-art methods and overall our performance is notably superior.

The proposed approach can also be applied, with small modifications, to predict sets of labels given instances (the 'normal' direction). The main difference is that in this case we will optimise score measures averaged over instances, instead of averaged over labels. We are going to show this in the next chapter.

A fundamental limitation of the approach described in this chapter is that it does not handle dependencies among labels. It is however possible to include such dependencies by assuming for example a bivariate feature map on the labels, rather than univariate. We are also going to address this problem in the next chapter.

# Chapter 4

# Submodular Multi-Label Learning

In this chapter we extend the algorithm introduced in Chapter 3 to allow for submodular pairwise label interactions. This enables the algorithm to explicitly model the dependencies between pairs of labels, improving its performance. This comes at a cost, however: even though inference remains efficient and exact, training becomes more expensive and complex, and we have to resort to an approximate algorithm due to the intractability of the constraint generation problem. Nevertheless we give both theoretical and empirical evidence that this algorithm is very effective.

## 4.1 Introduction

In Chapter 3 we proposed to apply the max-margin structured prediction framework to the problem of multi-label classification. We showed that, by choosing an appropriate objective function, we can optimise our model to the specific measure we are interested in. One limitation of that approach, however, is that it does not explicitly handle dependencies among labels. For example, in automatic image tagging, if the labels *ocean* and *ship* have high co-occurrence frequency in the training set, the learned model should somehow boost the chances of predicting *ocean* if there is strong visual evidence for the label *ship* [58].

In this chapter we extend the method from Chapter 3 to address this aspect. We explicitly model the dependencies between pairs of labels, albeit restricting them to be submodular (in rough terms, we model only the positive pairwise label correlations). This enables exact and efficient prediction at test time, since

finding an optimal subset of labels reduces to the minimisation of a particular kind of submodular set function which can be done efficiently via graph-cuts.

> Goal: To design a learning algorithm that produces multi-label classifiers optimised for a specific performance measure *and* takes into account label interactions.

Unlike in the previous chapter, however, here we deal with the problem of predicting labels given instances (i.e., the 'normal' direction). This makes more sense, since we want to model dependencies in the labels, not in the instances. A consequence of this is that we are now going to optimise for a score measure averaged over instances – the mean over all instances of the $F$-score.[1]

The critical technical contribution in this chapter is a constraint generation algorithm for loss-augmented inference where the scoring of the pair (input-output) is a submodular set function and the loss is derived from the $F$-score. This is what enables us to fit our model into the estimator from [16]. Our constraint generation algorithm is only approximate since the problem is intractable. However we give theoretical arguments supporting our empirical findings that the algorithm is not only very accurate in practice, but in the majority of our real-world experiments it actually produces a solution which is exactly optimal. We also present an efficient algorithm which serves as a certificate of optimality for this constraint generation algorithm. We compare the proposed method with other benchmark methods on publicly available multi-label datasets, and results support our approach.

### 4.1.1 Related Work

In Chapter 3 we have already reviewed recent work in MLC (see section 3.1.1), so here we will focus on related work that specifically attempts to encode label dependencies.

One of the first publications in this line of research was [59], which proposed a generative model where each label generates different words. Even though it can handle correlations between labels, it uses greedy heuristics to search over the exponentially large label space.

On the discriminative side, most methods assume that the labels are arranged in a known hierarchy or a taxonomy, usually encoded in a tree. Examples of these

[1] In this chapter we will deal only with the $F_1$-score, which we will write as $F$-score to keep the notation uncluttered.

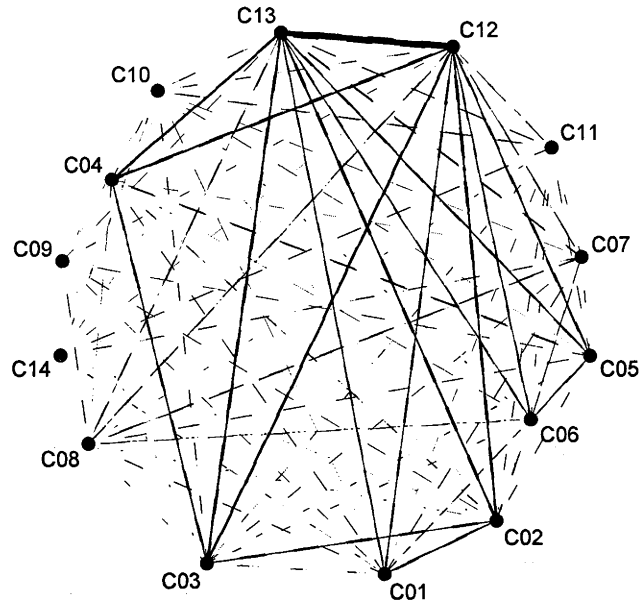



Figure 4.1: Graph of label co-occurrences in the *yeast* dataset (see Section 4.5.1 for a description of the datasets used). Nodes represent labels, and the thickness of the edges represents how often the corresponding labels co-occur in the training dataset, with no edge when there is no co-occurence. The graph is dense – 86 of the 91 possible edges are present.

methods are [60, 61, 62, 63, 64]. A more general approach with respect to the hierarchical structure is the method proposed in [53], as it deals with both tree and DAG-based dependencies among the labels. It does so by adapting decoding algorithms from signal processing to the problem of finding predictions consistent with the structures learned. Similarly, in [65] graphical models are used to impose structure in the label dependencies.

In many real-life problems, however, imposing a graph topology on the pairwise label interactions might not be realistic (see Figure 4.1 for an example). Our proposed method can therefore be seen as complementary to these methods, since we do not enforce any particular graph topology on the labels but instead we limit the nature of the interactions to be submodular.

[66] was perhaps the first to suggest the use of a max-margin framework to learn label interactions. Their method, however, deals only with the Hamming loss. [67] studies the multi-label problem under the assumption that prior knowledge on the density of label correlations is available. They also use a max-margin framework, similar in spirit to our formulation. Their main focus, however, is on

Table 4.1: Notation used throughout this chapter.

| | |
|---|---|
| $\mathcal{X}$ | set of instances |
| $N$ | number of instances |
| $\mathcal{Y}$ | set of outputs: $\mathcal{Y} = \{0,1\}^V$ |
| $V$ | number of possible labels |
| $D$ | dimensionality of instance features |
| $x$ | input instance, $x \in \mathcal{X}$ ($D \times 1$ vector) |
| $y$ | set of labels, $y \in \mathcal{Y}$ ($V \times 1$ vector) |
| $\bar{y}$ | predicted set of labels |
| $\theta^1$ | linear component of the parameter vector: $\theta^1 := [\dots \theta_i^{1T} \dots]^T$ |
| $\theta^2$ | quadratic component of the parameter vector: $\theta^2 := [\dots \theta_{ij}^2 \dots]^T$ |
| $\theta_i^1$ | parameter vector that defines how label $i$ weighs each feature of $x$ |
| $\theta_{ij}^2$ | scalar parameter that defines how label $i$ weighs label $j$ |
| $\theta$ | complete parameter vector: $\theta = [\theta^{1T}\ \theta^{2T}]^T$ |
| $\phi_1(x,y)$ | linear term of feature map: $\phi_1(x,y) = vec(x \otimes y)$ |
| $\phi_2(y)$ | quadratic term of feature map: $\phi_2(y) = vec(y \otimes y)$ |
| $\phi(x,y)$ | complete feature map: $\phi(x,y) = [\phi_1^T(x,y)\ \phi_2^T(y)]^T$ |
| $C_{ij}$ | normalised count of co-occurrence of labels $i$ and $j$ in the training set |

training speed, and they do not optimise for the F-score.

## 4.2 The Model

Let $x \in \mathcal{X}$ be a vector of dimensionality $D$ with the features of an instance (say, an image); let $y \in \mathcal{Y}$ be a *set of labels* for an instance (say, tags for an image), from a fixed dictionary of $V$ possible labels, encoded as $y \in \{0,1\}^V$. For example, $y = [1\ 1\ 0\ 0]$ denotes the first and second labels of a set of four. We assume we are given a training set $\{(x^n, y^n)\}_{n=1}^N$, and our task is to estimate a map $f : \mathcal{X} \to \mathcal{Y}$ that has good agreement with the training set but also generalises well to new data. Please refer to Table 4.1 for the notation used throughout this chapter.

In this section we define the class of functions $f$ that we will consider. In the next section we define the learning algorithm, i.e., a procedure to find a specific $f$ in the class.

### 4.2.1 The Loss Function Derived from the $F$-Score

Our notion of 'agreement' with the training set is given by a loss function. We focus on maximising the average over all instances of $F$, a score that considers both *precision* and *recall* and can be written in our notation as

$$\mathrm{F} = \frac{1}{N}\sum_{n=1}^{N} \frac{2\, p(y^n, \bar{y}^n) r(y^n, \bar{y}^n)}{p(y^n, \bar{y}^n) + r(y^n, \bar{y}^n)}, \text{ where } p(y, \bar{y}) = \frac{y^T \bar{y}}{\bar{y}^T \bar{y}} \text{ and } r(y, \bar{y}) = \frac{y^T \bar{y}}{y^T y}$$

Here $\bar{y}^n$ denotes our prediction for input instance $n$, $y^n$ is the corresponding ground-truth. Since our goal is to *maximise* the $F$-score a suitable choice of loss function is $\Delta(y, \bar{y}) = 1 - F(y, \bar{y})$, which is the one we adopt in this chapter. The loss for a single prediction is therefore

$$\Delta(y, \bar{y}) = 1 - \frac{2\, y^T \bar{y}}{y^T y + \bar{y}^T \bar{y}}. \tag{4.1}$$

### 4.2.2 Feature Maps and Parameterisation

We assume that the prediction for a given input $x$ is a maximiser of an expression that encodes both the unary dependency between labels and instances as well as the pairwise dependencies between labels:

$$\bar{y} \in \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, y^T A y \tag{4.2}$$

where $A$ is an upper-triangular matrix scoring the pair $(x, y)$, with diagonal elements $A_{ii} = \langle x, \theta_i^1 \rangle$, where $x$ is the input feature vector and $\theta_i^1$ is a parameter vector that defines how label $i$ weighs each feature of $x$. The off-diagonal elements are $A_{ij} = C_{ij}\theta_{ij}^2$, where $C_{ij}$ is the normalised count of co-occurrence of labels $i$ and $j$ in the training set, and $\theta_{ij}^2$ the corresponding scalar parameter ($\dim(\theta_{ij}^2) = 1$). We also define the complete parameter vectors $\theta^1 := [\ldots \theta_i^{1T} \ldots]^T$, $\theta^2 := [\ldots \theta_{ij}^2 \ldots]^T$ and $\theta = [\theta^{1T}\ \theta^{2T}]^T$, as well as the complete feature maps $\phi_1(x, y) = vec(x \otimes y)$, $\phi_2(y) = vec(y \otimes y)$ and $\phi(x, y) = [\phi_1^T(x, y)\ \phi_2^T(y)]^T$. This way the score in (4.2) can be written as $y^T A y = \langle \phi(x, y), \theta \rangle$. Note that the dimensionality of $\theta^2$ is the number of non-zero elements of $C$ – in this setting this is $\binom{V}{2}$, but it can be reduced by setting to zero elements of $C$ below a specified threshold.

In Appendix B we describe how to solve (4.2) via graph-cuts.

## 4.3 Learning Algorithm

### 4.3.1 Optimisation Problem

Direct optimisation of the loss defined in (4.1) is a highly intractable problem since it is a discrete quantity and our parameter space is continuous. Once again we follow the framework described in Section 2.1.1, and as a consequence at each iteration we need to maximise the violation margin $\xi_n$, which reduces to solving

$$y_n^* \in \operatorname*{argmax}_{y \in \mathcal{Y}} \left[\Delta(y, y^n) + \langle \phi(x^n, y), \theta \rangle\right]. \tag{4.3}$$

### 4.3.2 Learning Algorithm

Here we make use of the same quadratic solver as in Chapter 3 (BMRM [55]). For convenience, we repeat its description in Algorithm 4 (which requires Algorithm 5 as a subroutine). Note that other solvers could have been used instead. Our contribution lies not here, but in the routine of constraint generation for Algorithm 4, which is described in Algorithm 5.

BMRM requires the solution of constraint generation and the value of the objective function for the slack corresponding to the constraint generated, as well as its gradient. Soon we will discuss constraint generation. The other two ingredients we describe here. The slack at the optimal solution is

$$\xi_n^* = \Delta(y_n^*, y^n) + \langle \phi(x^n, y_n^*), \theta \rangle - \langle \phi(x^n, y^n), \theta \rangle, \tag{4.4}$$

thus the objective function from (2.6) becomes

$$\frac{1}{N} \sum_n \Delta(y_n^*, y^n) + \langle \phi(x^n, y_n^*), \theta \rangle - \langle \phi(x^n, y^n), \theta \rangle + \frac{\lambda}{2} \|\theta\|^2, \tag{4.5}$$

whose gradient is

$$\lambda\theta - \frac{1}{N} \sum_n (\phi(x^n, y^n) - \phi(x^n, y_n^*)). \tag{4.6}$$

Expressions (4.5) and (4.6) are then used in Algorithm 4.

### 4.3.3 Constraint Generation

The most challenging step consists of solving the constraint generation problem. Constraint generation for a given training instance $n$ consists of solving the combinatorial optimisation problem in (4.3), which, using our expression for the loss

**Algorithm 4** Bundle Method for Regularised Risk Minimisation (BMRM)

**Input:** training set $\{(x^n, y^n)\}_{n=1}^N$, $\lambda$, **Output:** $\theta$
Initialise $i = 1$, $\theta_1 = 0$, max$= -\infty$
**repeat**
  **for** $n = 1$ **to** $N$ **do**
    Compute $y_n^*$ ($y_{k_{max}}^{*n}$ returned by Algorithm 5.)
  **end for**
  Compute gradient $g_i$ (4.6) and objective $o_i$ (4.5)
  $\theta_{i+1} := \text{argmin}_\theta \frac{\lambda}{2} \|\theta\|^2 + \max(0, \max_{j \leq i} \langle g_j, \theta \rangle + o_j)$; $i \leftarrow i + 1$
**until** converged (see [55])
**return** $\theta$

in (4.1), as well as the correspondence $y^T A y = \langle \phi(x, y), \theta \rangle$, can be written as

$$y^{*n} \in \operatorname*{argmax}_y y^T A^n(y) y \tag{4.7}$$

where $\text{diag}(A^n) = \text{diag}(A) - \frac{2\, y^n}{|y| + |y^n|}$ and $\text{offdiag}(A^n) = \text{offdiag}(A)$. Note that the matrix $A^n$ depends on $y$. More precisely, a subset of its diagonal elements (those $A_{ii}^n$ for which $y^n(i) = 1$) depend on the quantity $|y|$, i.e., the number of nonzero elements in $y$. This makes solving (4.7) a formidable task. If $A^n$ were independent of $y$, then (4.7) could be solved exactly and efficiently via graph-cuts, just as our prediction problem in (4.2). A naïve strategy would be to aim for solving (4.7) $V$ times, one for each value of $|y|$, and constraining the optimisation to only include elements $y$ such that $|y|$ is fixed. In other words, we can partition the optimisation problem into $k$ optimisation problems conditioned on the sets $\mathcal{Y}_k := \{y : |y| = k\}$:

$$\max_y y^T A(y) y = \max_k \max_{y \in \mathcal{Y}_k} y^T A^{[k],n} y, \tag{4.8}$$

where $A^{[k],n}$ denotes the particular matrix $A^n$ that we obtain when $|y| = k$. However the inner maximisation above, i.e., the problem of maximising a supermodular function (or minimising a submodular function) subject to a cardinality constraint, is itself NP-hard [68]. We therefore do not follow this strategy, but instead seek a polynomial-time algorithm that in practice will give us an optimal solution most of the time.

Algorithm 5 describes our algorithm. In the worst case it calls graph-cuts $O(V)$ times, so the total complexity is $O(V^4)$.[2] The algorithm essentially searches

[2]The worst-case bound of $O(V^3)$ for graph-cuts is very pessimistic; in practice the algorithm is extremely efficient.

**Algorithm 5** Constraint Generation

**Input:** $(x^n, y^n)$, $\theta$, $V$, **Output:** $y^{*n}_{k_{max}}$
$k = 0$
$A^{[k],n}_{ij} = \langle \theta_{ij}, C_{ij} \rangle$ (for all $i, j : i \neq j$)
**while** $k \leq V$ **do**
  $diag(A^{[k],n}) = diag(A) - \frac{2y^n}{k+\|y^n\|^2}$
  $y^{*n}_k = \text{argmax}_y \, y^T A^{[k],n} y$ (graph-cuts)
  **if** $|y^{*n}_k| > k$ **then**
    $k_{max} = |y^{*n}_k|$; $k = k_{max}$
  **else if** $|y^{*n}_k| = k$ **then**
    $k_{max} = |y^{*n}_k|$; $k = k_{max} + 1$
  **else**
    $k = k + 1$
  **end if**
**end while**
**return** $y^{*n}_{k_{max}}$

for the largest $k$ such that solving $\text{argmax}_y \, y^T A^{[k],n} y$ returns a solution with $k$ *ones*. We call the $k$ obtained $k_{max}$, and the corresponding solution $y^{*n}_{k_{max}}$. Observe the fact that, as $k$ increases during the execution of the algorithm, $A^n_{ii}$ increases for those $i$ where $y^n(i) = 1$. The increment observed when $k$ increases to $k'$ is

$$\epsilon^{k'}_k := A^{[k'],n}_{ii} - A^{[k],n}_{ii} = 2\frac{k' - k}{(k' + |y^n|)(k + |y^n|)} \tag{4.9}$$

which is always a positive quantity. Although this algorithm is not provably optimal, Theorem 2 guarantees that it is sound in the sense that it never predicts incorrect labels. In the next section we present additional evidence supporting this algorithm, in the form of a test that if positive guarantees that the solution obtained is optimal.

We call a solution $y'$ a *partially optimal* solution[3] of $\text{argmax}_y \, y^T A^n(y) y$ if the labels it predicts as being present are indeed present in an optimal solution, i.e., if for those $i$ for which $y'(i) = 1$ we also have $y^{*n}(i) = 1$, for some $y^{*n} \in \text{argmax}_y \, y^T A^n(y) y$. Equivalently, we can write $y' \odot y^{*n} = y'$ (where $a \odot b$ denotes the element-wise product of vectors $a$ and $b$).

[3]This is related to the concept of weak autarky [69]. In the context where it is defined (binary partial labelling), weak autarky implies that whenever an algorithm labels a node, it will never increase the energy (loss) of the solution over an arbitrary labelling. This is equivalent to our concept of partially optimality, if we consider a value of 0 in $y$ as denoting unlabeled.

**Algorithm 6** Compute $\max_\alpha \beta_{A,\alpha}$

**Input:** $A^{[k_{max}],n}$, $y^{*n}_{k_{max}}$, $V$,
**Output:** max
$\max = -\infty$
$Z = \{i : y^{*n}_{k_{max}}(i) = 0\}$
$O = \{i : y^{*n}_{k_{max}}(i) = 1\}$
**for** $i \in Z$ **do**
  $O' = O \cup i$
  $\text{rmax} = \max_{y:y_{O'}=1} y^T A^{[k_{max}],n} y$ (graph-cuts)
  **if** rmax $>$ max **then**
    max = rmax
  **end if**
**end for**
$\max = \max - \max_y y^T A^{[k_{max}],n} y$
**return** max

**Theorem 2** *Upon completion of Algorithm 5,* $y^{*n}_{k_{max}}$ *is a partially optimal solution of* $\text{argmax}_y\, y^T A^n(y) y$.

The theorem means that whenever the algorithm predicts the presence of a label, it does so correctly; however there may be labels predicted as absent which are in fact present in the corresponding optimal solution.

The proof is conceptually simple but requires care due to the possible multiplicity of solutions. It consists of two main steps. First, we show that there must be an optimal solution to (4.7) with at least as many *ones* as the solution found by our algorithm. This is clearly a necessary condition for Theorem 2 to hold. Next we show that, for any possible solution found by our algorithm, there will be one such optimal solution which will indeed agree with our solution on all its *ones*. This proves the theorem. The first step is proven in proposition 5 below, which requires supporting results that we prove first. The theorem is subsequently proved.

We need some preliminary results.

**Lemma 3** *Let* $Y^*_k = \text{argmax}_y\, y^T A^{[k],n} y$, *where* $A^{[k],n}$ *is such that* $diag(A^{[k],n}) := diag(A) - \frac{2y^n}{k+\|y^n\|^2}$ *and* $A_{ij} = \langle \theta_{ij}, c_{ij} \rangle$ *for* $i \neq j$, *with* $\theta_{ij} \geq 0$ *and* $c_{ij} \geq 0$. *Now let* $l \geq k$ *and define similarly* $Y^*_l$. *Then for every* $y^*_k \in Y^*_k$ *there exists a* $y^*_l \in Y^*_l$ *such that* $y^*_k \odot y^*_l = y^*_k$ *(and, conversely, for every* $y^*_l \in Y^*_l$ *there exists a* $y^*_k \in Y^*_k$ *such that* $y^*_k \odot y^*_l = y^*_k$*).*

**Proof** The claim states that for any optimal solution $y_k^* \in Y_k^*$, its *ones* will also be present in some optimal solution $y_l^* \in Y_l^*$. The proof is by contradiction, assuming that there exists $y_k^* \in Y_k^*$ and an index $i$ such that $y_k^*(i) = 1$ and $y_l^*(i) = 0$ for all $y_l^* \in Y_l^*$. Now consider the binary vector $z_l$ which agrees with $y_l^*$ everywhere except in $i$, i.e. $z_l(j) = y_l^*(j)$ for $j \neq i$ and $z_l(i) = 1$. This implies that ${z_l}^T A^{[l],n} z_l = {y_l^*}^T A^{[l],n} y_l^* + A_{ii}^{[l],n} + \sum_{j:j\neq i} A_{ij}^{[l],n}$. Now note that we necessarily have $\sum_{j:j\neq i} A_{ij}^{[l],n} + A_{ii}^{[l],n} \geq 0$. This holds because, first, $\sum_{j:j\neq i} A_{ij}^{[k],n} + A_{ii}^{[k],n} \geq 0$ (otherwise $y_k^*$ would not be optimal), second, $A_{ii}^{[l],n} \geq A_{ii}^{[k],n}$ and third that $A_{ij}^{[l],n} = A_{ij}^{[k],n}$ for $i \neq j$. Therefore ${z_l}^T A^{[l],n} z_l \geq {y_l^*}^T A^{[l],n} y_l^*$, which implies that $z_l \in Y_l^*$, which contradicts the assumption that for all $y_l^* \in Y_l^*$, $y_l^*(i) = 0$. The converse is proved analogously. ■

Intuitively, the above result says that due to the non-negativity of the off-diagonal elements of $A$, the new objective function arising from an increase in the diagonal of $A$ surely has some maximiser which includes all *ones* already present in any maximiser of the previous objective function, prior to the increase in the diagonal.

**Corollary 4** *Let $l \geq k$. Then $\max_y y^T A^{[l],n} y \geq \max_y y^T A^{[k],n} y$.*

**Proof** Note that the set of ones potentially present in a $y_l^*$ but not in a $y_k^*$ for which $y_l^* \odot y_k^* = y_k^*$ necessarily has non-negative contribution to the objective function $y^T A^{[l],n} y$ (otherwise $y_l^*$ would not be optimal), jointly with the fact that the same is true for the set of ones present both in $y_l^*$ and $y_k^*$ since $A_{ii}^{[l],n} \geq A_{ii}^{[k],n}$ for any $i$. ■

We are now ready to prove that there exists an optimal solution to the constraint generation problem in (4.7) which contains at least $k_{max}$ *ones*.

**Proposition 5** *Let $k' \geq k_{max}$, where $k_{max}$ is as instantiated in Algorithm 5. Then there exists an optimal solution $y^{*n} \in \operatorname{argmax}_y y^T A^n(y) y$ such that for some $k'$, we have $y^{*n} \in \operatorname{argmax}_{y \in \mathcal{Y}_{k'}} y^T A^{[k'],n} y$.*

**Proof** This follows from (4.8) and from the claim that for all $k \leq k_{max}$, $\max_{y\in\mathcal{Y}_{k_{max}}} y^T A^{[k_{max}],n} y \geq \max_{y\in\mathcal{Y}_k} y^T A^{[k],n} y$, which we now prove. We know that $\max_y y^T A^{[k_{max}],n} y = \max_{y\in\mathcal{Y}_{k_{max}}} y^T A^{[k_{max}],n} y$ holds since, from lines 8 and 10 of Algorithm 5, we have $k_{max} = |y^{*n}_{k_{max}}|$. On the other hand, we have $\max_y y^T A^{[k],n} y \geq \max_{y\in\mathcal{Y}_k} y^T A^{[k],n} y$ since $\mathcal{Y}_k \subseteq \mathcal{Y}$. Both facts, when put together with corollary 4, prove the claim. ■

We are now able to give a proof of Theorem 2.

**Proof of Theorem 2** We show that for any solution $y^{*n}_{k_{max}}$ found by Algorithm 5, we have that $y^{*n}_{k_{max}} \odot y^{*n} = y^{*n}_{k_{max}}$ for some optimal solution $y^{*n}$ having at least $k_{max}$ *ones*. We proceed by contradiction, assuming that there is no optimal solution $y^{*n}$ respecting $|y^{*n}| \geq k_{max}$ such that $y^{*n}_{k_{max}} \odot y^{*n} = y^{*n}_{k_{max}}$ holds, for any $y^{*n}_{k_{max}}$. This is equivalent to saying that for every $y^{*n}$ respecting $|y^{*n}| \geq k_{max}$ there is an index $i$ (which can be different for different $y^{*n}$) such that $y^{*n}(i) = 0$ and $y^{*n}_{k_{max}}(i) = 1$ for any $y^{*n}_{k_{max}}$. Now consider a vector $z$ that agrees with $y^{*n}$ everywhere except in index $i$, i.e., $z(i) = 1$. The key observation now is that $z^T A^{[|z|],n} z \geq (y^{*n})^T A^{[|y^{*n}|],n} y^{*n}$, and therefore $z$ should be an optimal solution as well, which results in a contradiction. To see why this inequality holds, first note that $z^T A^{[|z|],n} z - (y^{*n})^T A^{[|y^{*n}|],n} y^{*n} = \sum_i (A^{[|z|],n}_{ii} - A^{[|y^{*n}|],n}_{ii}) + \sum_{j \neq i} A_{ij}$, where the first sum accounts for the potential change in the diagonal due to the increase in the cardinality of the solution, and the second sum accounts for the newly incorporated off-diagonal terms as a result of $z(i) = 1$. The result then follows from the submodularity assumption ($A_{ij} \geq 0, \forall i \neq j$) and from the fact that the diagonal is non-decreasing with respect to increases in the cardinality of the solution ($A^{[|z|],n}_{ii} - A^{[|y^{*n}|],n}_{ii} \geq 0, \forall i$, since $|z| > |y^{*n}|$). ■

## 4.4 Certificate of Optimality

As empirically verified in our experiments in section 4.5, our constraint generation algorithm (Algorithm 5) is indeed quite accurate: most of the time the solution obtained is optimal. In this section we present a test that if positive guarantees that an optimal solution has been obtained (i.e., a certificate of optimality). This can be used to generate empirical lower bounds on the probability that the algorithm returns an optimal solution (we explore this possibility in the experimental section).

We start by formalising the situation in which the algorithm will fail. Let $Z := \{i : y^{*n}_{k_{max}}(i) = 0\}$, and $\mathcal{P}_Z$ be the power set of $Z$ ($Z$ for 'zeros'). Let $O := \{i : y^{*n}_{k_{max}}(i) = 1\}$ ($O$ for 'ones'). Then the algorithm will fail if there exists $\alpha \in \mathcal{P}_Z$ such that

$$\underbrace{\sum_{i,j \in \alpha; i \neq j} A^n_{ij}}_{(a)} + \underbrace{\sum_{i \in \alpha, j \in O} A_{ij}}_{(b)} + \underbrace{\sum_{i \in \alpha} A^{[k_{max}+|\alpha|],n}_{ii}}_{(c)} + \underbrace{\epsilon^{k_{max}+|\alpha|}_{k_{max}} |y^n \odot y^{*n}_{k_{max}}|}_{(d)} > 0. \quad (4.10)$$

The above expression describes the situation in which, starting with $y^{*n}_{k_{max}}$, if we insert $|\alpha|$ *ones* in the indices defined by index set $\alpha$, we will obtain a new vector $y'$ which is a feasible solution of $\text{argmax}_y\, y^T A^n(y) y$ and yet has strictly larger score than solution $y^{*n}_{k_{max}}$. This can be understood by looking closely into each of the sums in expression (4.10). Sums $(a)$ and $(b)$ describe the increase in the objective function due to the inclusion of off-diagonal terms. Both $(a)$ and $(b)$ are non-negative due to the submodularity assumption. Term $(c)$ is the sum of the diagonal terms corresponding to the newly introduced *ones* of $y'$. Term $(c)$ is negative or zero, since each term in the sum is negative or zero (otherwise $y^{*n}_{k_{max}}$ would have included it). Finally, term $(d)$ is non-negative, being the total increase in the diagonal elements of $O$ due to the inclusion of $|\alpha|$ additional *ones*. We can write $(c)$ as

$$\underbrace{\sum_{i\in\alpha} A_{ii}^{[k_{max}+|\alpha|],n}}_{(c)} = \underbrace{\sum_{i\in\alpha} A_{ii}^{[k_{max}],n}}_{(e)} + \underbrace{\sum_{i\in\alpha}(A_{ii}^{[k_{max}+|\alpha|],n} - A_{ii}^{[k_{max}],n})}_{(f)} \tag{4.11}$$

and the last term can be bounded as

$$\sum_{i\in\alpha}(A_{ii}^{[k_{max}+|\alpha|],n} - A^{[k_{max}],n}) \leq \underbrace{\epsilon_{k_{max}}^{k_{max}+|\alpha|} v_\alpha}_{(g)} \tag{4.12}$$

where $v_\alpha = \min[|y^n| - |y^n \odot y^{*n}_{k_{max}}|, |\alpha|]$ is an upper bound on the number of indices $i \in \alpha$ such that $y^n(i) = 1$, and $\epsilon_{k_{max}}^{k_{max}+|\alpha|}$ is the increment in a diagonal element $i$ such that $y^n(i) = 1$ arising from increasing the cardinality of the solution from $k_{max}$ to $k_{max} + |\alpha|$. Incorporating bound (4.12) into (4.11), we get that $(c) \leq (e)+(g)$. We can then replace $(c)$ in inequality (4.10) by $(e)+(g)$, obtaining

$$\underbrace{\sum_{i,j\in\alpha; i\neq j} A_{ij}^n + \sum_{i\in\alpha, j\in O} A_{ij} + \sum_{i\in\alpha} A_{ii}^{[k_{max}],n}}_{:=\beta_{A,\alpha}} + \underbrace{\epsilon_{k_{max}}^{k_{max}+|\alpha|} v_\alpha + \epsilon_{k_{max}}^{k_{max}+|\alpha|} |y^n \odot y^{*n}_{k_{max}}|}_{:=\gamma_\alpha} > 0. \tag{4.13}$$

We know that, regardless of $A$ or $\alpha$, $\beta_{A,\alpha} \leq 0$ (otherwise $y^{*n}_{k_{max}} \notin \text{argmax}_y\, y^T A^{[k_{max}],n} y$, since $\beta$ is the increment in the objective function $y^T A^{[k_{max}],n} y$ obtained by adding *ones* in the entries of $\alpha$). The key fact coming to our aid is that $\gamma_\alpha$ is 'small', and a weak upper bound is 2. This is because

$$\begin{aligned}\epsilon_{k_{max}}^{k_{max}+|\alpha|} v_\alpha + \epsilon_{k_{max}}^{k_{max}+|\alpha|} |y^n \odot y^{*n}_{k_{max}}| \leq \epsilon_{k_{max}}^{k_{max}+|\alpha|} |y^n| \leq \epsilon_{k_{max}}^{V} |y^n| \leq \epsilon_0^V |y^n| = \\ = 2V|y^n|/((V + |y^n|)|y^n|) \leq 2.\end{aligned} \tag{4.14}$$

(Note that if $|y^n| = 0$ then $\gamma_\alpha = 0$ and our algorithm will always return an optimal solution since $\beta_{A,\alpha} \leq 0$).

Now, since $\beta_{A,\alpha} \leq 0$ for any $A$ and $\alpha \in \mathcal{P}_Z$, it suffices that we study the quantity $\max_\alpha \beta_{A,\alpha}$: if $\max_\alpha \beta_{A,\alpha} < -2$, then $\beta_{A,\alpha} < -2$ for any $\alpha \in \mathcal{P}_Z$. It is however very hard to understand theoretically the behaviour of the random variable $\max_\alpha \beta_{A,\alpha}$ even for a simplistic uniform i.i.d. assumption on the entries of $A$. This is because the domain of $\alpha$ ($\mathcal{P}_Z$) is itself a random quantity that depends on the particular $A$ chosen. This makes computing even the expected value of $\max_\alpha \beta_{A,\alpha}$ an intractable task, let alone obtaining concentration of measure results that could give us upper bounds on the probability of condition (4.13) holding under the assumed distribution on $A$.

However, for a *given* $A$ we can actually compute $\max_\alpha \beta_{A,\alpha}$ efficiently. This can be done with Algorithm 6. The algorithm effectively computes the gap between the scores of the optimal solution $y^{*n}_{k_{max}}$ and the highest scoring solution if one sets to 1 at least one of the zero entries in $y^{*n}_{k_{max}}$. It does so by solving graph-cuts constraining the solution $y$ to include the ones present in $y^{*n}_{k_{max}}$ but additionally fixing one of the zero entries of $y^{*n}_{k_{max}}$ to 1 (lines 7-8). This is done for every possible zero entry of $y^{*n}_{k_{max}}$, and the maximum score is recorded (lines 7-11). The gap between this and the score of the optimal solution $y^{*n}_{k_{max}}$ is then returned (line 13). This will involve $V - k_{max}$ calls to graph-cuts, and therefore the total computational complexity is $O(V^4)$. Once we compute $\max_\alpha \beta_{A,\alpha}$, we simply test wether $\max_\alpha \beta_{A,\alpha} + \epsilon^{|V|}_{k_{max}}|y^n| > 0$ holds (we use $\epsilon^{|V|}_{k_{max}}|y^n|$ rather than 2 as an upper bound for $\gamma_\alpha$ because, as seen from (4.14), it is the tightest upper bound which still does not depend on $\alpha$ and therefore can be computed). We have the following theorem

**Theorem 6** *Upon completion of Algorithm 6, if* $\max_\alpha \beta_{A,\alpha} + \epsilon^{|V|}_{k_{max}}|y^n| \leq 0$, *then* $y^{*n}_{k_{max}}$ *is an optimal solution of* $\operatorname{argmax}_y y^T A^n(y) y$.

**Proof** The theorem results directly from the fact that inequality (4.13) characterises the condition when the algorithm fails, as well as from the fact that $\max_\alpha \beta_{A,\alpha} \geq \beta_{A,\alpha}$ and $\epsilon^{V}_{k_{max}}|y^n| \geq \gamma_\alpha$. ■

Table 4.2: Datasets. #train/#test denotes the number of observations used for training and testing respectively; $V$ is the number of labels and $D$ the dimensionality of the features; Avg is the average number of labels per instance.

| dataset | domain | #train | #test | V | D | Avg |
|---|---|---|---|---|---|---|
| yeast | biology | 1500 | 917 | 14 | 103 | 4.23 |
| enron | text | 1123 | 579 | 53 | 1001 | 3.37 |

## 4.5 Experimental Results

To evaluate our multi-label learning method we applied it to real-world datasets and compared it to state-of-the art methods.

### 4.5.1 Datasets

For the sake of reproducibility we focused on publicly available datasets, and to ensure that the label dependencies have a reasonable impact on the results we restricted the experiments to datasets with a sufficiently large average number of labels per instance. We chose therefore two multilabel datasets from mulan:[4] *yeast* and *enron*. Table 4.2 describes them in more detail.

### 4.5.2 Experimental Setting

The datasets used have very informative unary features, so to better visualise the contribution of the label dependencies to the model we trained using varying amounts (1%, 10% and 100%) of the original unary features. We compared our proposed method to RML (the method proposed in Chapter 3, but without reversal[5]), which is essentially this chapter's model without the quadratic term, and to other state-of-the-art methods for which source code is publicly available – BM[41], RAkEL[44] and MLKNN[57].

### 4.5.3 Model Selection

Our model has two parameters: $\lambda$, the trade-off between data-fitting and good generalisation, and $c$, a scalar that multiplies $C$ to control the trade-off between

[4]http://mulan.sourceforge.net/datasets.html

[5]RML deals mainly with the reverse problem of predicting instances given labels, however it can be applied in the forward direction as well.

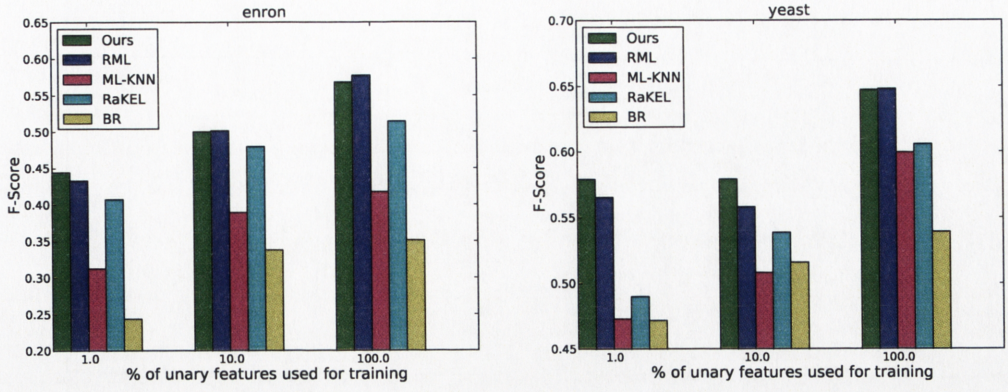



Figure 4.2: F-Score results on *enron* (left) and *yeast* (right), for different amounts of unary features. The horizontal axis denotes the proportion of the features used in training.

the linear and the quadratic terms. For each experiment we selected them with 5-fold cross-validation on the training data. We also control the sparsity of $C$ by setting $C_{ij}$ to zero for all except the top most frequent pairs – this way we can reduce the dimensionality of $\theta^2$, avoiding an excessive number of parameters for datasets with large values of $V$. In our experiments we used 50% of the pairs with *yeast* and 5% with *enron* (45 and 68 pairs, respectively). We experimented with other settings, but the results were very similar.

RML's only parameter, $\lambda$, was selected with 5-fold cross-validation. MLKNN's two parameters $k$ (number of neighbours) and $s$ (strength of the uniform prior) were kept fixed to 10 and 1.0, respectively, as was done in [57]. RAkEL's $m$ (number of models) and $t$ (threshold) were set to the library's default (respectively $2 * N$ and 0.5), and $k$ (size of the label set) was set to $\frac{V}{2}$ as suggested by [42]. For BM we kept the library's defaults.

### 4.5.4 Implementation

Our implementation is in C++, using the BMRM package [55]. Note that we had to modify BMRM to enforce positivity in $\theta^2$; the modifications, however, are relatively simple and are described in Appendix A. The max-flow computations needed for graph-cuts were done with the library of [70].

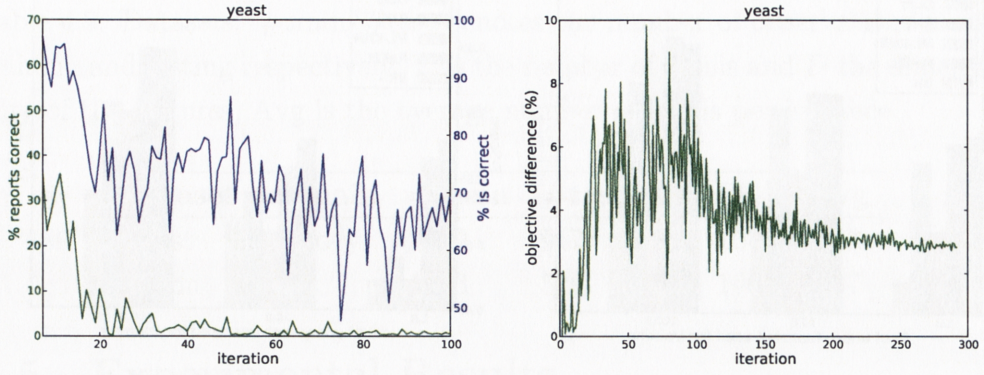



Figure 4.3: Empirical analysis of Algorithms 5 and 6 during training with the *yeast* dataset. Left: frequency with which Algorithm 5 is optimal at each iteration (blue) and frequency with which Algorithm 6 reports an optimal solution has been found by Algorithm 5 (green). Right: difference, at each iteration, between the objective computed using the results from Algorithm 5 and exhaustive enumeration.

### 4.5.5 Results: F-Score

In Figure 4.2 we plot the $F$-Score for varying-sized subsets of the unary features, for both *enron* (left) and *yeast* (right). The goal is to assess the benefits of explicitly modelling the pairwise label interactions, particularly when the unary information is deteriorated. As can be seen in Figure 4.2, when all features are available our model behaves similarly to RML. The results are slightly inferior due to two main reasons:

- setting: our parameters ($\lambda$ and $c$) are chosen with cross-validation from a small set of possible values, and therefore are not optimal;

- approximation: we are using an approximate constraint generation algorithm, whereas RML uses an exact one.

In this setting the unary features are very informative and the pairwise interactions are not helpful. As we reduce the number of available unary features (from right to left in the plots), the importance of the pairwise interactions increases, and our model demonstrates improvement over RML.

### 4.5.6 Results: Correctness

To evaluate how well our constraint generation algorithm performs in practice we compared its results against those of exhaustive search, which is exact but only feasible for a dataset with a small number of labels, such as *yeast*. We also assessed the strength of our test proposed in Algorithm 6. In Figure 4.3 (left) we plot, for the first 100 iterations of the learning algorithm, the frequency with which Algorithm 5 returns the exact solution (blue line) as well as the frequency with which the test given in Algorithm 6 guarantees that the solution is exact (green line). Overall we can see that in more than 50% of its executions Algorithm 5 produces an optimal solution. Our test effectively offers a lower bound which is informative in the sense that overall, its variations reflect legitimate variations in the real quantity of interest (as can be seen by the correlation between the two curves).

For the learning algorithm, however, what we are interested in is the objective $o_i$ and the gradient $g_i$ of line 7 of Algorithm 4, and both depend only on the compound result of $N$ executions of Algorithm 5 at each iteration of the learning algorithm. This is illustrated in Figure 4.3 (right), where we plot, for each iteration, the normalised difference between the objective computed with results from Algorithm 5 and the one computed with the results of an exact exhaustive search.[6] We can see that the difference is quite small – below 4% after the initial iterations.

### 4.5.7 Training Time

The time the algorithm takes to train depends on the average time per iteration and on the number of iterations.

At each iteration the running time is dominated by the $N$ calls to the constraint generation algorithm, where each one may require at most $V$ calls to the max-flow algorithm. The computational complexity of the whole learning algorithm is therefore $O(iNV^4)$, where $i$ is the number of iterations. In practice, however, this upper bound is loose.

In Table 4.3 we summarise training times along the variables mentioned above, for the experiments with all the features (rightmost bars in Figure 4.2). Note that our implementation is multi-threaded, and scales almost linearly with the number of available cpus, so wall-clock times can be much smaller.

[6] We repeated this experiment with several sets of parameters with similar results.

Table 4.3: Training times.

| Dataset | Training time | i | N | V | Time/call to Alg. 5 |
|---|---|---|---|---|---|
| **Yeast** | 47.8 cpu-seconds | 205 | 1500 | 14 | 155 us |
| **Enron** | 847.3 cpu-seconds | 116 | 1123 | 53 | 6504 us |

## 4.6 Summary

This chapter extended the multi-label learning method presented in Chapter 3 to explicitly models label dependencies in a submodular fashion. As an estimator we again used structured support vector machines optimised with constraint generation. Our key contribution is an algorithm for constraint generation which is proven to be partially optimal in the sense that all labels it predicts are included in some optimal solution. We also described an efficient test that if positive guarantees that the solution found is optimal.

We presented empirical results that corroborate the fact that the algorithm is very accurate, and we illustrate the gains obtained in comparison to other popular algorithms, particularly the algorithm we described in Chapter 3, which can be seen as a particular case when there are no explicit label interactions being modeled.

# Chapter 5

# Exponential Family Graph Matching and Ranking

Document ranking is a fundamental problem in IR [10, 11]. Finding a ranking of a set of documents can be seen as searching for an optimal permutation of the documents, which can be modeled as a bipartite matching problem. Bipartite graph matching, also known as linear assignment, is one of the best known combinatorial optimisation problems, which can be solved efficiently in polynomial time. This model is useful in a number of applications involving assignment of resources to tasks. The classical approach consists of hand-tuning the weights of the graph according to prior knowledge about the application at hand, and then proceeding to solve the matching problem. However in order to leverage data one should instead seek an estimator to produce the weights of the graph from training examples of good matches. Existing estimators for this problem are based on structured support vector machines. In this chapter we introduce an alternative estimator for this problem, namely maximum a posteriori estimation in exponential families. The driving motivation for seeking alternative estimators is the importance of having a fully probabilistic approach, which allows integration with other probabilistic models. We show that for small graphs maximum a posteriori estimation is computationally efficient. For larger graphs we point out the existence of a sampler that allows for approximations of maximum a posteriori estimation, albeit being slow in practice.

## 5.1 Introduction

The Maximum-Weight Bipartite Matching Problem (henceforth 'matching problem') is a fundamental problem in combinatorial optimisation [71]. This is the problem of finding the 'heaviest' perfect match in a weighted bipartite graph. An exact optimal solution can be found in cubic time by standard methods such as the Hungarian algorithm.

This problem is of practical interest because it can nicely model real-world applications. For example, in computer vision the crucial problem of finding a correspondence between sets of image features is often modeled as a matching problem [72, 73]. Ranking algorithms can be based on a matching framework [74], as can clustering algorithms [75, 76] (see Figure 5.1 for some example applications).

When modeling a problem as one of matching, one central question is the choice of the weight matrix. The problem is that in real applications we typically observe edge *feature vectors*, not edge weights. Consider a concrete example in computer vision: it is difficult to tell what the 'similarity score' is between two image feature points, but it is straightforward to extract feature vectors (e.g. SIFT) associated with those points.

In this setting, it is natural to ask whether we could parameterise the features, and use *labeled matches* in order to estimate the parameters such that, given graphs with 'similar' features, their resulting max-weight matches are also 'similar'.

[17] and [73] describe max-margin structured learning formalisms for this problem. As we discussed in Chapter 2, max-margin structured estimators are appealing in that they *try* to minimise the loss that one really cares about ('structured losses', of which the Hamming loss is an example). However structured losses are typically piecewise constant in the parameters, which eliminates any hope of using smooth optimisation directly. Max-margin estimators instead minimise a surrogate loss which is easier to optimise, namely a convex upper bound on the structured loss [16]. In practice the results are often good, but it is very challenging to integrate such models in larger systems since they have no probabilistic nature and therefore cannot be easily used as a module in a graphical model formulation for large systems, which is often of importance for computing expectations of relevant quantities in a decision-theoretic setting.

Goal: To design a probabilistic model for predicting bipartite matchings.

Motivated by the lack of a probabilistic interpretation for max-margin structured estimators, in this chapter we present a MAP estimator for the matching problem. The observed data are the edge feature vectors and the labeled matches provided for training. We then maximise the conditional posterior probability of matches given the observed data. We build an exponential family model where the sufficient statistics are such that the mode of the distribution (the prediction) is the solution of a max-weight matching problem. The resulting partition function is ♯P-complete to compute exactly. However, we show that for *learning to rank* applications the model instance is tractable, and can be solved by exact enumeration. We then compare the performance of our model instance against a large number of state-of-the-art ranking methods, including DORM [74], an approach that only differs from our model instance by using max-margin instead of a MAP formulation. We show very competitive results on standard document ranking datasets, and in particular we show that our model performs better than or on par with DORM. For intractable model instances, we show that the problem can be approximately solved using sampling and we provide experiments from the computer vision domain. However the fastest suitable sampler is still quite slow for large models, in which case max-margin matching estimators like those of [73] and [17] are likely to be preferable even in spite of their potentially inferior accuracy.

### 5.1.1 The Matching Problem

Consider a weighted bipartite graph with $m$ nodes in each part, $G = (V, E, w)$, where $V$ is the set of vertices, $E$ is the set of edges and $w : E \mapsto \mathbb{R}$ is a set of real-valued weights associated with the edges (please refer to Table 3.1 for the notation used throughout this chapter). $G$ can be simply represented by a matrix $(w_{ij})$ where the entry $w_{ij}$ is the weight of the edge $ij$. Consider also a bijection $y : \{1, 2, \ldots, m\} \mapsto \{1, 2, \ldots, m\}$, i.e., a permutation. Then the matching problem consists of computing

$$y^* = \operatorname*{argmax}_y \sum_{i=1}^{m} w_{iy(i)}. \tag{5.1}$$

This is a well-studied problem: it is tractable and can be solved in $O(m^3)$ time [77, 71]. This model can be used to match features in images [73], improve classification algorithms [76] and rank documents [74], to cite a few applications

Table 5.1: Notation used throughout this chapter.

| | |
|---|---|
| $G$ | weighted bipartite graph $G = (V, E, w)$ |
| $V$ | set of vertices of a graph |
| $E$ | set of edges of a graph |
| $w$ | set of real-valued weights associated with the edges of a graph ($m \times m$ matrix) |
| $m$ | number of nodes in each side of a bipartite graph |
| $y$ | a bijection $\{1, 2, \ldots, m\} \mapsto \{1, 2, \ldots, m\}$ |
| $x_e$ | feature vector associated with edge $e$ ($d \times 1$ vector) |
| $\theta$ | parameter vector ($d \times 1$ vector) |
| $N$ | number of training instances |
| $\phi(x, y)$ | feature map (5.6) |
| $q_k$ | query $k$: a list of documents $\{d_1^k, \ldots, d_{D(k)}^k\}$ with corresponding ratings $\{r_1^k, \ldots, r_{D(k)}^k\}$ |
| $D(k)$ | number of documents retrieved by query $k$ |
| $R$ | number of possible ratings for a document |
| $\psi_i^k$ | joint feature vector for document $d_i^k$ and query $q_k$. |

(see Figure 5.1 for some examples). The typical setting consists of engineering the score matrix $w_{ij}$ according to domain knowledge and subsequently solving the combinatorial problem.

### 5.1.2 Ranking

Ranking is a fundamental problem with applications in diverse areas such as document retrieval, recommender systems, product rating and others. In this chapter we focus, without loss of generality, on document ranking.

In this setting the ranking problem can be described as follows: given a query and a list of documents retrieved by the query, our task is to order these documents by relevance with respect to the query.

The traditional approach has been to use domain knowledge to engineer a document scoring function and use its outputs to order the documents – examples are BM25 [79] and language models for information retrieval (LMIR) [80]. The problem with this approach is that these functions have to be manually tuned, a complex task that becomes increasingly challenging as more sophisticated models are created.

We can, however, apply machine learning to this task. Assuming we are given

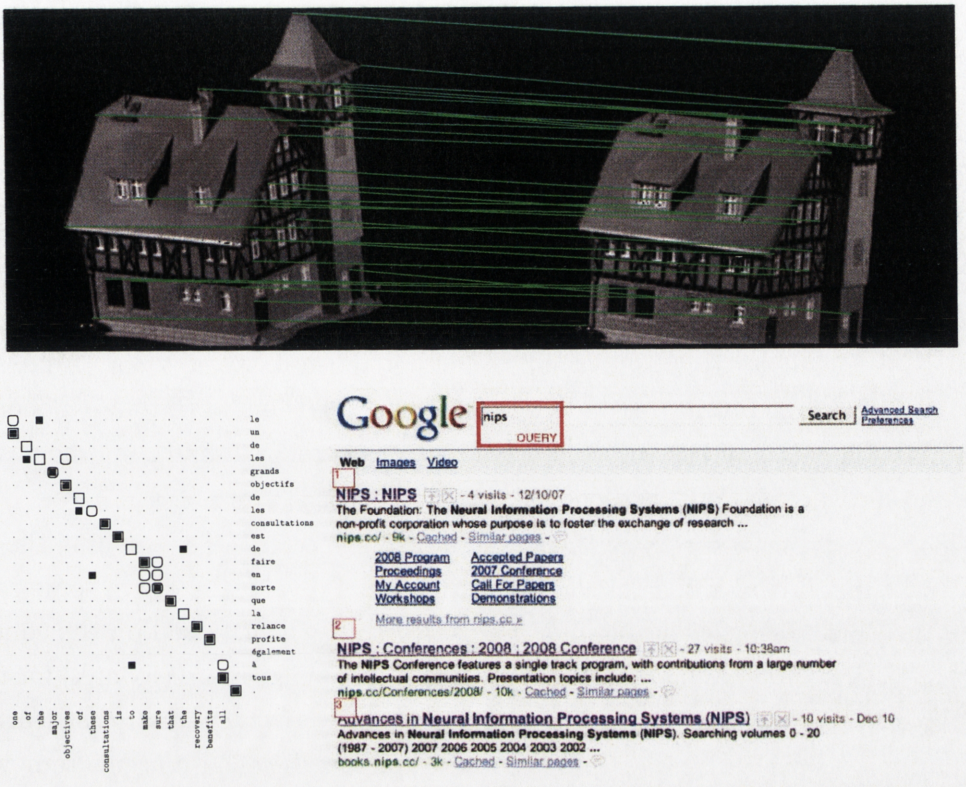



Figure 5.1: Some applications of graph matching. Top: image matching [73]. Bottom left: machine translation [78] (reproduced with permission of Ben Taskar). Bottom right: page ranking [74].

a list of queries and corresponding retrieved documents ordered by a human editor, we can use statistical tools to learn a ranking function that, given the documents, will order them by relevance.

### 5.1.3 Related Work

Initial learning to rank methods applied a *pairwise approach*: given a query and all its retrieved documents, pairs of documents are used as instances in learning. Given a pair there are only two possible orderings, and so this becomes a binary classification problem and standard methods can be applied. Support Vector Machines in this setting leads to RankSVM [81]. Similar approaches with boosting and neural networks lead to RankBoost [82] and RankNet [83] respectively.

The limitation of these methods is that they give the same penalty for incorrect orderings no matter what the position of the documents in the ranking is. This is undesirable, since we want to give more weight to errors in the first positions –

in document ranking applications, for example, usually only the 10 top positions are considered, since these correspond to the first retrieved page of results. This is also reflected in ranking measures, such as *normalised discounted cumulative gain* (NDCG). [84] therefore proposed a *listwise approach* where document lists are used in training instead of document pairs. To that end, a map from a list of scores to a probability distribution is defined, and a metric between two probability distributions is used as a listwise loss function. Neural Networks are the model used for this method, which is called ListNet. [85] (AdaRank) attacked the same problem by applying the boosting framework to a exponential loss function based on IR performance measures, therefore minimising the desired ranking measure (NDCG, for example).

[86] raised some issues with the loss used in the model of [83] – mainly the fact that it is unbounded and cannot achieve the minimal value of zero in some cases; they then proposed a new loss function to overcome them, based on the concept of the Fidelity distance measure from physics, and an algorithm (FRank) for minimising this loss using a generalised additive model. [74] (DORM) cast learning to rank as a graph matching problem and applied the max-margin framework in the structured learning setting, directly optimising the desired performance measure. [87] proposed a general boosting method (QBRank) that can handle complex losses and applied it to learning to rank by combining the use of preference data (i.e., document $i$ is preferred over document $j$) and label data (i.e., document $i$'s relevance level is $r$) in learning. [88] (IsoRank) proposed a 'minimum effort' optimisation method that takes into account the entire training data within a query at each iteration, applying functional iterative methods for learning. [89] (SortNet) applied Neural Networks in a pairwise manner, but with the training set selected by an iterative procedure that, at each iteration, adds the most informative training examples. [90] (StructRank) proposed using cumulative distribution networks (CDNs) to maximise a joint cumulative distribution function (CDF) over multiple pairwise preferences. [91] proposed using both content information and relations between objects to define the ranking model and used Continuous Conditional Random Fields (C-CRFs) to perform the learning task. [92] (BoltzRank) introduced an energy-based, listwise approach, defining a conditional probability distribution over rankings that combines potentials that depend on individual documents and pairs of documents. [93] proposed to use an ordered weighted average (OWA) of pairwise losses as a way to focus on top ranked elements.

In the years following the publication of our research, learning to rank contin-

ued to be a very active area of research. One line of work has been on reducing test time:[1] [95] addressed this issue by jointly optimising effectiveness and efficiency of linear ranking functions; [94] proposed to reduce the execution time of decision tree ensembles using early exits, showing considerable speed improvements with little or no degradation in the quality of the results.

Another recent line of research is on applying *transfer learning* to ranking, as in [96], which proposed a multi-task learning algorithm with boosted decision trees and applied it to web-search ranking, using data sets from several countries.

Finally, there has been increasing interest in online learning to rank: [97] proposed an online learning algorithm that can quickly refine the results of existing ranking methods based on real-time users' feedback; [98] is concerned with *recency ranking* – ranking documents while taking their recency into account – and proposed to improve ranking recency with a query classification algorithm that automatically detects recency queries with high precision, and apply a specialised ranker to them.

## 5.2 The Model

### 5.2.1 Basic Goal

In this chapter we assume that the weights $w_{ij}$ are to be *estimated* from training data. More precisely, the weight $w_{ij}$ associated with the edge $ij$ in a graph will be the result of an appropriate composition of a *feature vector* $x_{ij}$ (observed) and a *parameter vector* $\theta$ (estimated from training data). Therefore, in practice, our input is a *vector-weighted* bipartite graph $G_x = (V, E, x)$ $(x : E \mapsto \mathbb{R}^n)$, which is 'evaluated' at a particular $\theta$ (obtained from previous training) so as to attain the graph $G = (V, E, w)$. See Figure 5.2 for an illustration.

More formally, assume that a training set $\{X, Y\} = \{(x^n, y^n)\}_{n=1}^N$ is available, where $x^n := (x^n_{11}, x^n_{12} \ldots, x^n_{M(n)M(n)})$. Here $M(n)$ is the number of nodes in each part of the vector-weighted bipartite graph $x^n$. We then parameterise $x_{ij}$ as $w_{iy(i)} = f(x_{iy(i)}; \theta)$, and the goal is to find the $\theta$ which maximises the posterior probability of the observed data. We will assume $f$ to be bilinear, i.e., $f(x_{iy(i)}; \theta) = \langle x_{iy(i)}, \theta \rangle$.

[1] Search engines typically require that the document scoring phase does not exceed a few hundred milliseconds [94].

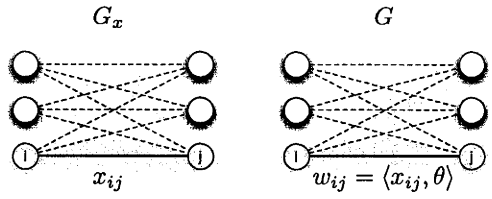



Figure 5.2: Left: Illustration of an input *vector-weighted* bipartite graph $G_x$ with $3 \times 3$ edges. There is a vector $x_e$ associated with each edge $e$ (for clarity only $x_{ij}$ is shown, corresponding to the solid edge). Right: weighted bipartite graph $G$ obtained by evaluating $G_x$ on the learned vector $\theta$ (again only edge $ij$ is shown).

### 5.2.2 Exponential Family Model

We assume an exponential family model, where the probability model is

$$p(y|x;\theta) = \exp\left(\langle \phi(x,y), \theta \rangle - g(x;\theta)\right), \text{ where} \tag{5.2}$$

$$g(x;\theta) = \log \sum_y \exp \langle \phi(x,y), \theta \rangle \tag{5.3}$$

is the log-partition function, which is a convex and differentiable function of $\theta$ [20].

That is exactly the same setting as we described in section 2.1.2, so our loss function is:

$$\ell(Y|X;\theta) = \frac{\lambda}{2} \|\theta\|^2 + \frac{1}{N} \sum_{n=1}^{N} \left(g(x^n;\theta) - \langle \phi(x^n, y^n), \theta \rangle\right) \tag{5.4}$$

where $\lambda$ is a regularisation constant. $\ell(Y|X;\theta)$ is a convex function of $\theta$ since the log-partition function $g(\theta)$ is a convex function of $\theta$ [20] and the other terms are clearly convex in $\theta$.

### 5.2.3 Feature Parameterisation

The critical observation now is that we equate the solution of the matching problem (5.1) to the prediction of the exponential family model (2.10), i.e., $\sum_i w_{iy(i)} = \langle \phi(x,y), \theta \rangle$. Since our goal is to parameterise features of individual pairs of nodes

(so as to produce the weight of an edge), the most natural model is

$$\phi(x, y) = \sum_{i=1}^{M} x_{iy(i)}, \text{ which gives} \tag{5.5}$$

$$w_{iy(i)} = \left\langle x_{iy(i)}, \theta \right\rangle, \tag{5.6}$$

i.e., linear in both $x$ and $\theta$ (see Figure 5.2, right). The specific form for $x_{ij}$ will be discussed in the experimental section. In light of (5.6), (5.1) now clearly means a *prediction* of the best match for $G_x$ under the model $\theta$.

## 5.3 Learning the Model

### 5.3.1 Basics

We need to solve $\theta^* = \text{argmin}_\theta \ell(Y|X;\theta)$. $\ell(Y|X;\theta)$ is a convex and differentiable function of $\theta$ [20], therefore gradient descent will find the global optimum. In order to compute $\nabla_\theta \ell(Y|X;\theta)$, we need to compute $\nabla_\theta g(\theta)$. It is a standard result of exponential families that the gradient of the log-partition function is the expectation of the sufficient statistics:

$$\nabla_\theta g(x;\theta) = \mathbf{E}_{y \sim p(y|x;\theta)}[\phi(x, y)]. \tag{5.7}$$

Therefore in order to perform gradient descent we need to compute the above expectation. Opening the above expression gives

$$\mathbf{E}_{y \sim p(y|x;\theta)}[\phi(x, y)] = \sum_y \phi(x, y) p(y|x;\theta) \tag{5.8}$$

$$= \frac{1}{Z(x;\theta)} \sum_y \phi(x, y) \prod_{i=1}^{M} \exp(\left\langle x_{iy(i)}, \theta \right\rangle), \tag{5.9}$$

which reveals that the partition function $Z(x;\theta)$ needs to be computed. The partition function is:

$$Z(x;\theta) = \sum_y \prod_{i=1}^{M} \underbrace{\exp(\left\langle x_{iy(i)}, \theta \right\rangle)}_{=:B_{iy(i)}}. \tag{5.10}$$

Note that the above is the expression for the *permanent* of matrix $B$ [99]. The permanent is similar in definition to the determinant, the difference being that for the latter $\text{sgn}(y)$ comes before the product. However, unlike the determinant, which is computable efficiently and exactly by standard linear algebra manipulations [100], computing the permanent is a $\sharp$P-complete problem [101]. Therefore we have no realistic hope of computing (5.7) exactly for general problems.

### 5.3.2 Exact Expectation

The exact partition function itself can be efficiently computed for up to about $M = 30$ using the $O(M2^M)$ algorithm by Ryser [102]. However for arbitrary expectations we are not aware of any exact algorithm which is more efficient than full enumeration (which would constrain tractability to very small graphs). However we will see that even in the case of very small graphs we find a very important application: learning to rank. In our experiments, we successfully apply a tractable instance of our model, small enough to be solved by exact enumeration, to benchmark document ranking datasets, obtaining very competitive results. For larger graphs, we have alternative options as indicated below.

### 5.3.3 Approximate Expectation

If we have a situation in which the set of feasible permutations is too large to be fully enumerated efficiently, we need to resort to some approximation for the expectation of the sufficient statistics. One option is to use the sampler proposed by Huber and Law, who recently presented an algorithm to approximate the permanent of dense non-negative matrices [103]. The algorithm works by producing *exact samples* from the distribution of perfect matches on weighted bipartite graphs. This is in precisely the same form as the distribution we have here, $p(y|x;\theta)$. We can use this algorithm for applications that involve larger graphs.[2] We generate $K$ samples from the distribution $p(y|x;\theta)$, and directly approximate (5.8) with a Monte Carlo estimate

$$\mathbf{E}_{y\sim p(y|x;\theta)}[\phi(x,y)] \approx \frac{1}{K}\sum_{i=1}^{K}\phi(x,y_i). \tag{5.11}$$

In our experiments, in order to illustrate the use of the sampler, we selected an image matching application, which requires larger graphs than the ranking problem.

## 5.4 Experimental Results

### 5.4.1 Ranking

Here we apply the general matching model introduced in previous sections to the task of *learning to rank*. Ranking is a fundamental problem with applications in

[2]The algorithm is described in appendix C.

diverse areas such as document retrieval, recommender systems, product rating and others. We focus on document ranking.

We are given a set of queries $\{q_k\}$ and, for each query $q_k$, a list of $D(k)$ documents $\{d_1^k, \ldots, d_{D(k)}^k\}$ with corresponding ratings $\{r_1^k, \ldots, r_{D(k)}^k\}$ (assigned by a human editor), measuring the relevance degree of each document with respect to query $q_k$. A rating or relevance degree is usually a nominal value in the list $\{1, \ldots, R\}$, where $R$ is typically between 2 and 5. We are also given, for every retrieved document $d_i^k$, a joint feature vector $\psi_i^k$ for that document and the query $q_k$.

### Training

At training time, we model each query $q_k$ as a vector-weighted bipartite graph (Figure 5.2) where the nodes on one side correspond to *a subset* of cardinality $M$ of all $D(k)$ documents retrieved by the query, and the nodes on the other side correspond to all possible ranking positions for these documents $(1, \ldots, M)$. The subset itself is chosen randomly, provided that at least one exemplar document of every rating is present. Therefore $M$ must be such that $M \geq R$.

The process is then repeated in a bootstrap manner: we resample (with replacement) from the set of documents $\{d_1^k, \ldots, d_{D(k)}^k\}$, $M$ documents at a time (conditioned on the fact that at least one exemplar of every rating is present, but otherwise randomly). This effectively boosts the number of training examples since each query $q_k$ ends up being selected many times, each time with a different subset of $M$ documents from the original set of $D(k)$ documents.

In the following we drop the query index $k$ to examine a single query. Here we follow the construction used in [74] to map matching problems to ranking problems (indeed the only difference between our ranking model and that of [74] is that they use a max-margin estimator and we use MAP in an exponential family.) Our edge feature vector $x_{ij}$ will be the product of the feature vector $\psi_i$ associated with document $i$, and a scalar $c_j$ (the choice of which will be explained below) associated with ranking position $j$

$$x_{ij} = \psi_i c_j. \tag{5.12}$$

$\psi_i$ is dataset specific (see details below). From (5.6) and (5.12), we have $w_{ij} = c_j \langle \psi_i, \theta \rangle$, and training proceeds as explained in Section 5.3.

### Testing

At test time, we are given a query $q$ and its corresponding list of $D$ associated documents. We then have to solve the prediction problem, i.e.,

$$y^* = \underset{y}{\operatorname{argmax}} \sum_{i=1}^{D} \langle x_{iy(i)}, \theta \rangle = \underset{y}{\operatorname{argmax}} \sum_{i=1}^{D} c_{y(i)} \langle \psi_i, \theta \rangle . \tag{5.13}$$

We now notice that if the scalar $c_j = c(j)$, where $c$ is a non-increasing function of rank position $j$, then (5.13) can be solved simply by sorting the values of $\langle \psi_i, \theta \rangle$ in decreasing order.[3] In other words, the matching problem becomes one of *ranking the values* $\langle \psi_i, \theta \rangle$. Inference in our model is therefore very fast (linear time).[4] In this setting it makes sense to interpret the quantity $\langle \psi_i, \theta \rangle$ as a *score* of document $d_i$ for query $q$. This leaves open the question of which non-increasing function $c$ should be used. We do not solve this problem here, and instead choose a fixed $c$. In theory it is possible to optimise over $c$ during learning, but in that case the optimisation problem would no longer be convex.

### Data sets

We describe the results of our method on LETOR 2.0 [106], a publicly available benchmark data collection for comparing learning-to-rank algorithms. It is comprised of three data sets: OHSUMED, TD2003 and TD2004.

OHSUMED contains features extracted from query-document pairs in the OHSUMED collection, a subset of MEDLINE, a database of medical publications. It contains 106 queries. For each query there are a number of associated documents, with relevance degrees judged by humans on three levels: *definitely*, *possibly* or *not relevant*. Each query-document pair is associated with a 25 dimensional feature vector, $\psi_i$. The total number of query-document pairs is 16,140.

TD2003 and TD2004 contain features extracted from the topic distillation tasks of TREC 2003 and TREC 2004, with 50 and 75 queries, respectively. Again, for each query there are a number of associated documents, with relevance degrees judged by humans, but in this case only two levels are provided: *relevant* or *not relevant*. Each query-document pair is associated with a 44 dimensional feature vector, $\psi_i$. The total number of query-document pairs is 49,171 for TD2003 and

[3] If $r(v)$ denotes the vector of ranks of entries of vector $v$, then $\langle a, \pi(b) \rangle$ is maximised by the permutation $\pi^*$ such that $r(a) = r(\pi^*(b))$, a theorem due to Polya, Littlewood, Hardy and Blackwell [104].

[4] Sorting the top $k$ items of a list of $D$ items takes $O(k \log k + D)$ time [105].

74,170 for TD2004. All datasets are already partitioned for 5-fold cross-validation. See [106] for more details.

### Evaluation Metrics

In order to measure the effectiveness of our method we use the NDCG measure [107] at rank position $k$, which is defined as

$$NDCG@k = \frac{1}{Z} \sum_{j=1}^{k} \frac{2^{r(j)} - 1}{\log(1+j)}. \tag{5.14}$$

here $r(j)$ is the relevance of the $j^{th}$ document in the list, and $Z$ is a normalisation constant so that a perfect ranking yields an NDCG score of 1.

### External Parameters

The regularisation constant $\lambda$ is chosen by 5-fold cross-validation, with the partition provided by the LETOR package. All experiments are repeated 5 times to account for the randomness of the sampling of the training data. We use $c(j) = M - j$ on all experiments.

### Optimisation

To optimise (5.4) we use a standard BFGS Quasi-Newton method with a backtracking line search, as described in [50].

### Results

For the first experiment training was done on subsets sampled as described above, where for each query $q_k$ we sampled $0.4 \cdot D(k) \cdot M$ subsets, therefore increasing the number of samples linearly with $M$. For TD2003 we also trained with all possible subsets ($M = 2(all)$ in the plots). In Figure 5.3 (left) we plot the results of our method (named RankMatch), for $M = R$, compared to those achieved by a number of state-of-the-art methods which have published NDCG scores in at least two of the datasets: RankBoost [82], RankSVM [81], FRank [86], ListNet [84], AdaRank [85], QBRank [87], IsoRank [88], SortNet [89], StructRank [90] and C-CRF [91]. We also included a plot of our implementation of DORM [74], using *precisely* the same resampling methodology and data for a fair comparison. RankMatch performs among the best methods on both TD2004 and OHSUMED, while on TD2003 it performs poorly (for low $k$) or fairly well (for high $k$).

In Figure 5.3 (right) we compare variants of our method for different values of $M$. Overall, we observe that our method does not seem to benefit from the use of larger $M$. This is intuitively plausible, since if $M > R$, documents with the same relevance appear in different orders during training, which may create unnecessary bias.

We notice that there are four methods which only report results in two of the three datasets: the two SortNet versions are only reported on TD2003 and TD2004, while StructRank and C-CRF are only reported on TD2004 and OHSUMED. RankMatch compares similarly with SortNet and StructRank on TD2004, similarly to C-CRF and StructRank on OHSUMED and similarly to the two versions of SortNet on TD2003. This exhausts all the comparisons against the methods which have results reported in only two datasets. A fairer comparison could be made if these methods had their performance published for the respective missing dataset.

When compared to the methods which report results in all datasets, RankMatch entirely dominates their performance on TD2004 and is second only to IsoRank on OHSUMED (and performing similarly to QBRank).

These results should be interpreted cautiously: [108] presents an interesting discussion about issues with these datasets. Also, benchmarking of ranking algorithms is still in its infancy and we don't yet have publicly available code for all of the competitive methods.

### Finite Sample Performance of the Estimator

In a second experiment we trained RankMatch with different training subset sizes, starting with $0.03 \cdot D(k) \cdot M$ and going up to $1.0 \cdot D(k) \cdot M$. Once again, we repeated the experiments with DORM using *precisely* the same training subsets. The purpose here is to see whether we observe a practical advantage of our method with increasing sample size. The results are plotted in Figure 5.4 (right), where we can see that, as more training data is available, RankMatch improves more saliently than DORM.

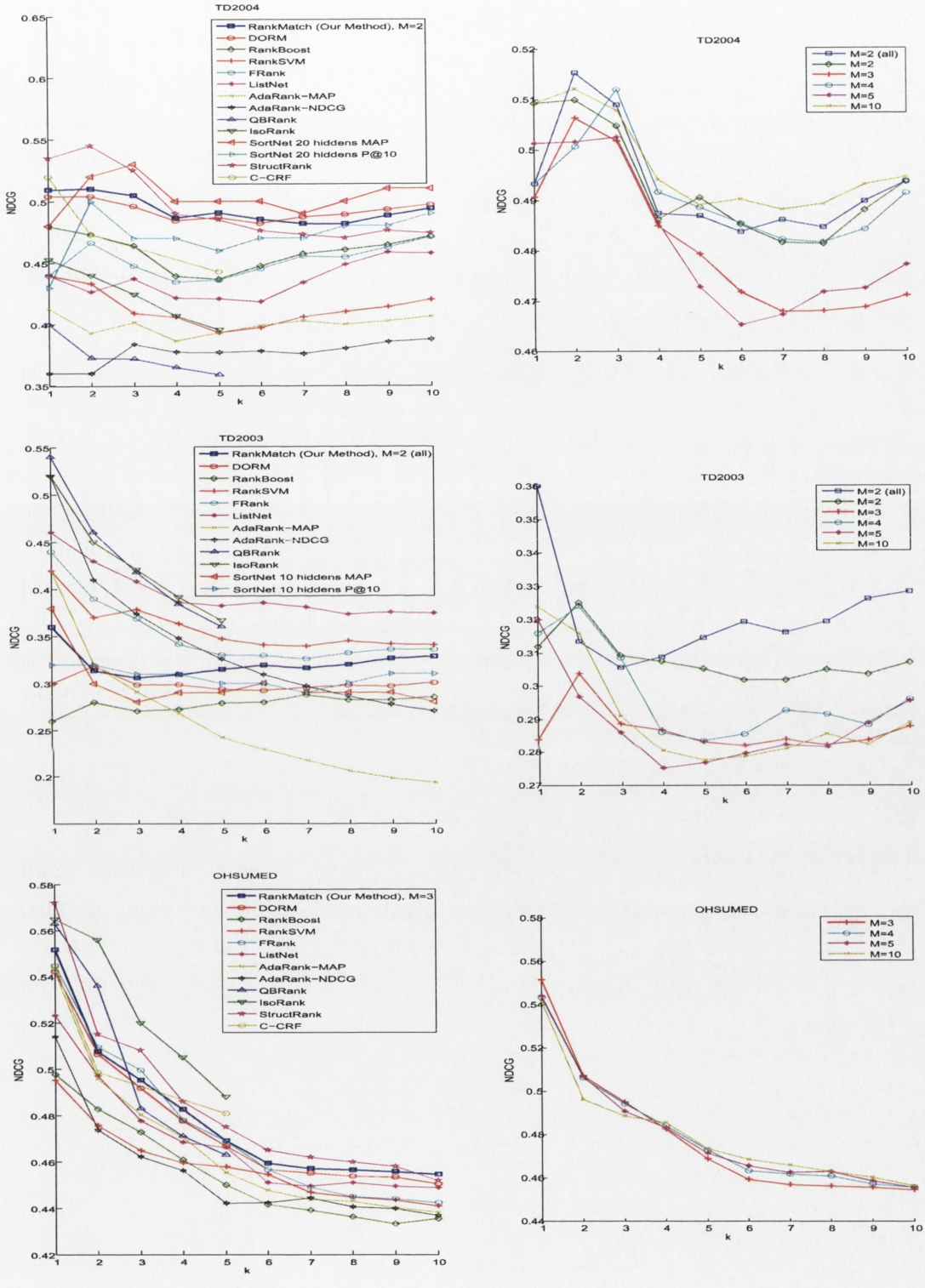



Figure 5.3: Results of NDCG@k. Left: comparing to state-of-the-art methods. Right: varying $M$. From top to bottom: TD2004, TD2003 and OHSUMED.

### 5.4.2 Image Matching

In order to illustrate the use of the sampler (see Section 5.3.3) to deal with larger graphs, we applied our matching model to a computer vision application. We took a silhouette image from the Mythological Creatures 2D database,[5] randomly selected 20 points on the silhouette as our interest points and applied shear to the image creating 200 different images. We then randomly selected $N$ pairs of images for training, $N$ for validation and 500 for testing, and trained our model to match the interest points in the pairs. In this setup,

$$x_{ij} = |\psi_i - \psi_j|^2, \tag{5.15}$$

where $|\cdot|$ denotes the element-wise difference and $\psi_i$ is the Shape Context feature vector [109] for point $i$.

For a graph of this size computing the exact expectation is not feasible, so we used the sampling method described in Section 5.3.3. The regularisation constant $\lambda$ was chosen by cross-validation, and once again we compared the performance of the MAP and max-margin estimators as the sample size grows. We present results with varying training set sizes in Figure 5.4 (left). The max-margin method is that of [73]. After a sufficiently large training set size, our model exhibits a slight advantage.

### 5.4.3 Runtime

The runtime of our algorithm is competitive with that of max-margin for small graphs, such as those that arise from the ranking application. For larger graphs, the use of the sampling algorithm will result in much slower runtimes than those typically obtained in the max-margin framework. Table 5.2 shows the runtimes for graphs of different sizes, both for exponential family and max-margin matching models.

[5] http://tosca.cs.technion.ac.il

Table 5.2: Training times (per observation, in seconds, on an Intel Core2 2.4GHz machine) for exponential model (EM) and max-margin (MM). Runtimes for $M = 3, 4, 5$ are from the ranking experiments, computed by full enumeration (exact); $M = 20$ corresponds to the shape image matching experiments, where we used the sampler from [103].

| M | EM (exact) | EM (sampling) | MM |
|---|---|---|---|
| 3 | 0.0006661 | - | 0.0008965 |
| 4 | 0.0011277 | - | 0.0016086 |
| 5 | 0.0030187 | - | 0.0015328 |
| 20 | - | 36.0300000 | 0.9334556 |

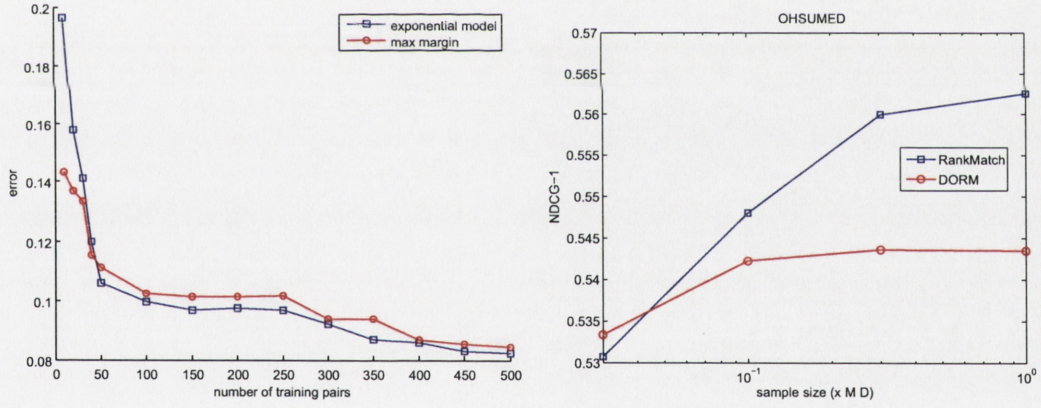



Figure 5.4: Performance with increasing sample size. Left: Hamming loss for different numbers of training pairs in the image matching problem (test set size fixed to 500 pairs). Right: results of NDCG@1 on the ranking dataset OHSUMED. The plots above seem to suggest that our estimator enjoys a better finite sample performance, since after a finite number of training instances it becomes superior to the max-margin estimator.

## 5.5 Summary

Document ranking is a fundamental problem in IR, and has been the subject of intensive research in recent years.[6] This chapter presented a method for a supervised structured learning problem – learning max-weight bipartite matching predictors – and applied it extensively to well-known document ranking datasets, obtaining state-of-the-art results. It also illustrated – with an image matching application – that larger problems can also be solved, albeit slowly, with a recently developed sampler. The method has some convenient features: it consists of performing MAP estimation in an exponential family model, which results in a simple unconstrained convex optimisation problem solvable by standard algorithms such as BFGS; and, being fully probabilistic, it can easily be integrated as a module in a Bayesian framework.

[6] See, for example, http://research.microsoft.com/en-us/um/beijing/projects/letor/paper.aspx.

# Part II

# Unsupervised Structured Learning

# Chapter 6

# Word Features for Latent Dirichlet Allocation

Topic models are widely used in IR for summarising documents [6], and have been studied in the context of IR for years [13, 14, 15]. In this chapter we propose an extension to a well known type of topic model, LDA, that explicitly allows for the encoding of side information in the distribution over words. This results in a variety of new capabilities, such as improved estimates for infrequently occurring words, as well as the ability to leverage thesauri and dictionaries in order to boost topic cohesion within and across languages. We present experiments on multi-language topic synchronisation where dictionary information is used to bias corresponding words towards similar topics. Results indicate that our model substantially improves topic cohesion when compared to the standard LDA model.

## 6.1 Introduction

LDA [30] assigns topics to documents and generates topic distributions over words given a collection of texts. In doing so, it ignores any side information about the similarity between words. Nonetheless, it achieves a surprisingly high quality of coherence within topics.

The inability to deal with word features makes LDA fall short on several aspects. The most obvious one is perhaps that the topics estimated for infrequently occurring words are usually unreliable. Ideally, for example, we would like the topics associated with *synonyms* to have a prior tendency of being similar, so that in case one of the words is rare but the other is common, the topic estimates for

the rare one can be improved. There are other examples. For instance, it is quite plausible that 'Germany' and 'German', or 'politics', 'politician', and 'political' should, by default, belong to the same topic. Similarly, we would like to be able to leverage dictionaries in order to boost topic cohesion across languages, a problem that has been researched but is far from being fully solved, especially for non-aligned corpora [110]. For example, we know that 'democracy' and 'democracia' are different words, but it is clear that not leveraging the fact they actually mean the same thing (and therefore should have aligned topics) reduces the statistical strength of a model. This is specially relevant for IR applications, where it is not uncommon to have a corpus composed of documents in different languages.

Goal: To extend Latent Dirichlet Allocation to allow for the encoding of side information on the words.

A possible solution, which we propose in this chapter, is to treat word information as *features* rather than as explicit constraints and to adjust a smoothing prior over topic distributions for words such that correlation is emphasised. In the parlance of LDA we do *not* pick a globally constant $\beta$ smoother over the word multinomials but rather we adjust it according to word similarity. In this way we are capable of learning the prior probability of how words are distributed over various topics based on how similar they are, e.g. in the context of dictionaries, synonym collections, thesauri, edit distances, or distributional word similarity features.

Unfortunately, in performing such model extension we lose full tractability of the setting by means of a collapsed Gibbs sampler. Instead, we use a hybrid approach where we perform smooth optimisation over the word smoothing coefficients, while retaining a collapsed Gibbs sampler to assign topics for a fixed choice of smoothing coefficients. The advantage of this setting is that it is entirely modular and can be added to existing Gibbs samplers without modification.

We present experimental results on multi-language topic synchronisation which clearly evidence the ability of the model to incorporate dictionary information successfully. Using several different measures of topic alignment, we consistently observe that the proposed model improves substantially on standard LDA, which is unable to leverage this type of information.

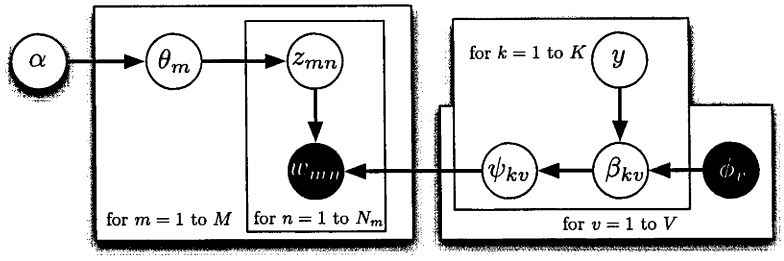



Figure 6.1: Our Extension: We assume that we observe side information $\phi_v$ (i.e. features) for each word $v$. The word-specific smoothing parameters $\beta_{kv}$ are governed by $\phi_v$ and a common parameter choice $y$.

### 6.1.1 Related Work

Loosely related works that use logistic models to induce structure in generative models are [111], which proposed a shared logistic normal distribution as a Bayesian prior over probabilistic grammar weights, and [112], which incorporated features into unsupervised models using locally normalised models. More related to our work is [113], which encodes correlations between synonyms, and [114] which encodes more general correlations. In fact, our proposed model can be seen as a generalisation of [114], where we can encode the strength of the links between each pair of words.

Previous work on multilingual topic models requires parallelism at either the sentence level [115] or document level [116, 117]. More recent work [118] relaxes that, but still requires that a significant fraction (at least 25%) of the documents are paired up.

Multilingual topic alignment without parallelism was recently proposed by [110]. Their model requires a list of matched word pairs $m$ (where each pair has one word in each language) and corresponding matching priors $\pi$ that encode the prior knowledge on how likely the match is to occur. The topics are defined as distributions over word pairs, while the unmatched words come from a unigram distribution specific to each language. Although their model could in principle be extended to more than two languages their experimental section was focused on the bilingual case.

One of the key differences between [110] and our method is that we do not hardcode word information, but we use it only as a prior – this way our method becomes less sensitive to errors in the word features. Furthermore, our model automatically generalises to multiple languages without any modification, align-

ing topics even for language pairs for which we have no information, as we show in the experimental section for the Portuguese/French pair. Finally, our model is conceptually simpler and can be incorporated as a module in existing LDA implementations.

Since our research was published there has been limited new work on the specific subject of multilingual topic models; one example is [119], which proposed a probabilistic model for bilingual documents; their method requires paired documents for training, but doesn't need a dictionary – in this sense it is complementary to ours. At the time of this writing, however, we couldn't find any new relevant work on incorporating features into LDA.

## 6.2 The Model

As we reviewed in Chapter 2, the LDA model of [30] (Figure 2.3) assumes that

$$\theta_m \sim \text{Dir}(\alpha), \qquad (6.1a) \qquad \psi_k \sim \text{Dir}(\beta), \qquad (6.1c)$$
$$z_{mn} \sim \text{Mult}(\theta_m), \qquad (6.1b) \qquad w_{mn} \sim \text{Multi}(\psi_{z_{mn}}). \qquad (6.1d)$$

Nonparametric extensions in terms of the number of topics can be obtained using Dirichlet process models [120] regarding the generation of topics. Our extension deals with the word smoother $\beta$. Instead of treating it as a constant for all words we attempt to infer its values for different words and topics. That is, we assume that (6.1c) is replaced by

$$\psi_k \sim \text{Dir}(\beta_k|\phi, y). \qquad (6.2)$$

We refer to this setting as *downstream conditioning*, in analogy to the *upstream conditioning* of [121] (which dealt with topical side information over documents). The corresponding graphical model is given in Figure 6.1. The above dependency allows us to incorporate features of words as side information. For instance, if two words (e.g. 'politics' and 'politician') are very similar then it is plausible that their topic distributions should also be quite similar. This can be achieved by choosing similar $\beta_{k,\text{politics}}$ and $\beta_{k,\text{politician}}$. For example, both of those coefficients might have great affinity to $\beta_{k,\text{scandal}}$ and we might estimate $y$ such that this is achieved.

Table 6.1: Notation used throughout this chapter.

| variable | description |
|---|---|
| $K$ | number of topics |
| $M$ | number of documents |
| $V$ | dictionary size |
| $\alpha_k$ | Dirichlet prior for $\theta$ (hyperparameter), for topic $k$ |
| $\alpha$ | vector $\alpha = [\alpha_1 \dots \alpha_K]$ |
| $\bar{\alpha}$ | $\bar{\alpha} := \|\alpha\|_1$ |
| $\beta_{kv}$ | Dirichlet prior for $\psi$ (hyperparameter), for topic $k$ and term $v$ |
| $\beta_k$ | vector $\beta_k = [\beta_{k1} \dots \beta_{kV}]$ |
| $\bar{\beta}_k$ | $\bar{\beta}_k := \|\beta_k\|_1$ |
| $\theta$ | distribution of topics per document |
| $\psi$ | distribution of words per topic |
| $z_{mn}$ | topic $(1..K)$ of word $n$ of document $m$ |
| $w_{mn}$ | term index $(1..V)$ of word $n$ of document $m$ |
| $n_{kv}^{KV}$ | number of times the term $v$ has been observed with topic $k$ |
| $n_k^K$ | number of times topic $k$ has been observed in all documents |
| $n_{km}^{KM}$ | number of times topic $k$ has been observed in a word of document $m$ |
| $n_m^M$ | number of words in document $m$ |
| $n_v^V$ | total number of times term $v$ has been observed in the corpus |
| $G$ | similarity graph $G = (V, E, \phi)$ |
| $V$ | set of vertices of a graph |
| $E$ | set of edges of a graph |
| $\phi_{uv}$ | weight for edge between vertices $u$ and $v$ |
| $y_{kv}$ | smoothing coefficient for topic $k$ and term $v$ |
| $\Gamma$ | gamma function: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt$ |
| $\Upsilon$ | digamma function: $\Upsilon(x) := \partial_x \log \Gamma(x)$ |

### 6.2.1 Detailed Description

We now discuss the directed graphical model from Figure 6.1 in detail. Whenever needed we use the *collapsed* representation of the model [33], that is we integrate out the parameters $\theta_m$ and $\psi_{kv}$ such that we only need to update $\alpha$ and $\beta$ (or indirectly $y$). We define the standard quantities (please refer to Table 6.1 for the notation used throughout this chapter):

$$n_{kv}^{\mathrm{KV}} = \sum_{m,n} \{z_{mn} = k \text{ and } w_{mn} = v\} \qquad n_k^{\mathrm{K}} = \sum_m n_{km}^{\mathrm{KM}} \qquad n_m^{\mathrm{M}} = \sum_k n_{km}^{\mathrm{KM}}$$
$$n_{km}^{\mathrm{KM}} = \sum_n \{z_{mn} = k\} \qquad n_v^{\mathrm{V}} = \sum_k n_{kv}^{\mathrm{KV}},$$

as well as:

**Topic distribution** $p(z_{mn}|\theta_m)$: We assume that this is a multinomial distribution specific to document $m$, that is $p(z_{mn}|\theta_m) = \theta_{m,z_{mn}}$.

**Conjugate distribution** $p(\theta_m|\alpha)$: This is a Dirichlet distribution with parameters $\alpha$, where $\alpha_k$ denotes the smoother for topic $k$.

**Collapsed distribution** $p(z_m|\alpha)$**:** Integrating out $\theta_m$ and using conjugacy yields

$$p(z_m|\alpha) = \frac{\prod_{k=1}^K \Gamma(n_{km}^{\mathrm{KM}} + \alpha_k)}{\Gamma\left(n_m^{\mathrm{M}} + \|\alpha\|_1\right)} \frac{\Gamma\left(\|\alpha\|_1\right)}{\prod_{k=1}^K \Gamma(\alpha_k)},$$

where $\Gamma$ is the gamma function: $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt$.

**Word distribution** $p(w_{mn}|z_{mn}, \psi)$: We assume that given a topic $z_{mn}$ the word $w_{mn}$ is drawn from a multinomial distribution $\psi_{w_{mn},z_{mn}}$. That is $p(w_{mn}|z_{mn}, \psi) = \psi_{w_{mn},z_{mn}}$. This is entirely standard as per the basic LDA model.

**Conjugate distribution** $p(\psi_k|\beta_k)$: As by default, we assume that $\psi_k$ is distributed according to a Dirichlet distribution with parameters $\beta_k$. The key difference is that here we do not assume that all coordinates of $\beta_k$ are identical.

**Collapsed distribution** $p(w|z, \beta)$: Integrating out $\psi_k$ for all topics $k$ yields the following

$$p(w|z, \beta) = \prod_{k=1}^K \frac{\prod_{v=1}^V \Gamma(n_{kv}^{\mathrm{KV}} + \beta_{kv})}{\Gamma\left(n_k^{\mathrm{K}} + \|\beta_k\|_1\right)} \frac{\Gamma\left(\|\beta_k\|_1\right)}{\prod_{v=1}^V \Gamma(\beta_{kv})}.$$

### 6.2.2 Priors

In order to better control the capacity of our model, we impose a prior on naturally related words, e.g. the ('Toyota', 'Kia') and the ('Bush', 'Cheney') tuples, rather than generally related words. For this purpose we design a similarity graph

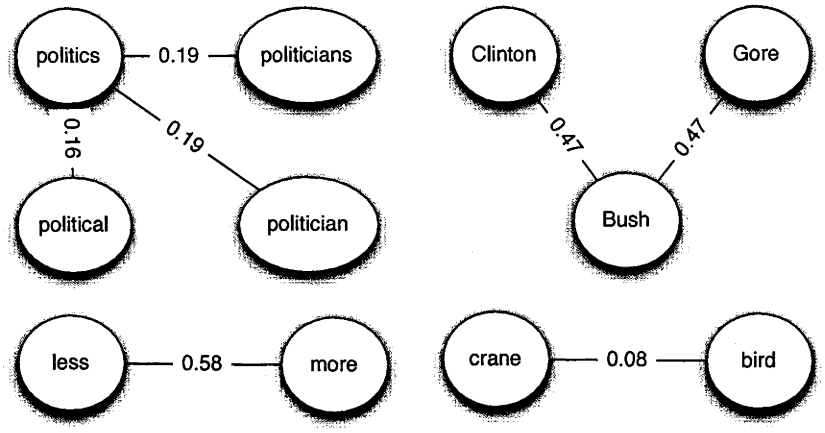



Figure 6.2: Example similarity graph: vertices represent words, and edges connect similar words. Edge weights can encode the degree of similarity.

$G(V, E)$ with words represented as vertices $V$ and similarity edge weights $\phi_{uv}$ between vertices $u, v \in V$ whenever $u$ is related to $v$. In particular, the magnitude of $\phi_{uv}$ can denote the similarity between words $u$ and $v$ (see Figure 6.2 for an example).

In the following we denote by $y_{kv}$ the topic dependent smoothing coefficients for a given word $v$ and topic $k$. We impose the smoother

$$\log \beta_{kv} = y_{kv} + y_v \text{ and } \log p(\beta) = \frac{-1}{2\lambda^2} \left[ \sum_{v,v',k} \phi_{v,v'} (y_{kv} - y_{kv'})^2 + \sum_v y_v^2 \right]$$

where $\log p(\beta)$ is given up to an additive constant and $y_v$ allows for multiplicative topic-independent corrections. A similar model was used by [122] to capture temporal dependence between topic models computed at different time instances, e.g. when dealing with topic drift over several years in a scientific journal. There the vertices are words at a given time and the edges are between smoothers instantiated at subsequent years.

## 6.3 Inference

In analogy to the collapsed sampler of [33] we also represent the model in a collapsed fashion. That is, we integrate out the random variables $\theta_m$ (the document topic distributions) and $\psi_{kv}$ (the topic word distributions), which leads to a joint likelihood in terms of the actual words $w_{mn}$, the side information $\phi$ about

words, the latent variable $y$, the smoothing hyperprior $\beta_{kv}$, and finally, the topic assignments $z_{mn}$.

### 6.3.1 Document Likelihood

The likelihood contains two terms: a word-dependent term which can be computed online while resampling data[1], and a model-dependent term involving the topic counts and the word-topic counts which can be computed by one pass through the aggregate tables respectively. Let us first write out the uncollapsed likelihood in terms of $z, \theta, \psi, \alpha, \beta$. We have

$$p(w, z, \theta, \psi | \alpha, \beta) = \prod_{m=1}^{M} \prod_{n=1}^{N_m} p(w_{mn}|z_{mn}, \psi) p(z_{mn}|\theta_m) \prod_{m=1}^{M} p(\theta_m|\alpha) \prod_{k=1}^{K} p(\psi_k|\beta).$$

Define $\bar{\alpha} := \|\alpha\|_1$ and $\bar{\beta}_k := \|\beta_k\|_1$. Integrating out $\theta$ and $\psi$ yields

$$p(w, z|\alpha, \beta) = \prod_{m=1}^{M} \frac{\Gamma(\bar{\alpha})}{\Gamma(\bar{\alpha} + n_m^{\mathrm{M}})} \prod_{k: n_{km}^{\mathrm{KM}} \neq 0} \frac{\Gamma(\alpha_k + n_{km}^{\mathrm{KM}})}{\Gamma(\alpha_k)} \prod_{k=1}^{K} \frac{\Gamma(\bar{\beta}_k)}{\Gamma(\bar{\beta}_k + n_k^{\mathrm{K}})} \prod_{v: n_{kv}^{\mathrm{KV}} \neq 0} \frac{\Gamma(\beta_{kv} + n_{kv}^{\mathrm{KV}})}{\Gamma(\beta_{kv})}.$$

The above product is obtained simply by cancelling out terms in the denominator and numerator where the counts vanish. This is computationally significant, since it allows us to evaluate the normalisation for sparse count tables with cost linear in the number of nonzero coefficients rather than in the number of entries in the dense count table.

### 6.3.2 Collapsed Sampler

In order to perform inference we need two components: a sampler which is able to draw from $p(z_i = k|w, z_{\neg i}, \alpha, \beta)$[2], and an estimation procedure for $(\beta, y)$. The sampler is essentially the same as in standard LDA. For the count variables $n^{\mathrm{KM}}, n^{\mathrm{KV}}, n^{\mathrm{K}}$ and $n^{\mathrm{M}}$ we denote by the subscript '$-$' their values after the word $w_{mn}$ and associated topic $z_{mn}$ have been removed from the statistics. Standard calculations yield the following topic probability for resampling:

$$p(z_{mn} = k|\text{rest}) \propto \frac{\left[\beta_{kv} + n_{kv_{mn}-}^{\mathrm{KV}}\right] \left[n_{km-}^{\mathrm{KM}} + \alpha_k\right]}{n_{k-}^{\mathrm{K}} + \bar{\beta}_k}. \tag{6.6}$$

[1]Note that this is not entirely correct – the model changes slightly during one resampling pass, hence the log-likelihood that we compute is effectively the averaged log-likelihood due to an ongoing sampler. For a correct computation we would need to perform one pass through the data without resampling. Since this is wasteful, we choose the approximation instead.

[2]Here $z_i$ denotes the topic of word $i$, and $z_{\neg i}$ the topics of all words in the corpus except for $i$.

In Section 6.7 we show how we can adapt the sampler of [36] to obtain faster sampling.

### 6.3.3 Topic Smoother for $\beta$

Optimising over $y$ is considerably hard since the log-likelihood does not decompose efficiently. This is due to the dependence of $\bar{\beta}_k$ on all words in the dictionary. The data-dependent contribution to the negative log-likelihood is

$$L_\beta = \sum_{k=1}^{K} \left[\log \Gamma(\bar{\beta}_k + n_k^{\mathrm{K}}) - \log \Gamma(\bar{\beta}_k)\right] + \sum_{k=1}^{K} \sum_{v: n_{kv}^{\mathrm{KV}} \neq 0} \left[\log \Gamma(\beta_{kv}) - \log \Gamma(\beta_{kv} + n_{kv}^{\mathrm{KV}})\right]$$

with gradients given by the appropriate derivatives of the $\Gamma$ function. We use the prior from section 6.2.2, which smooths between closely related words only. After choosing edges $\phi_{uv}$ according to these matching words, we obtain an optimisation problem directly in terms of the variables $y_{kv}$ and $y_v$. Denote by $N(v)$ the neighbours for word $v$ in $G(V, E)$, and $\Upsilon(x) := \partial_x \log \Gamma(x)$ the Digamma function. We have

$$\partial_{y_{kv}} \left[L_\beta - \log p(\beta)\right] = \frac{1}{\lambda^2} \sum_{v' \in N(v)} \phi_{v,v'} \left[y_{kv} - y_{kv'}\right] + \beta_{kv} \Big( \Upsilon(\bar{\beta}_k + n_k^{\mathrm{K}}) - \Upsilon(\bar{\beta}_k) + \\ + \left\{n_{kv}^{\mathrm{KV}} > 0\right\} \left[\Upsilon(\beta_{kv}) - \Upsilon(\beta_{kv} + n_{kv}^{\mathrm{KV}})\right] \Big).$$

The gradient with respect to $y_k$ is analogous:

$$\partial_{y_v} \left[L_\beta - \log p(\beta)\right] = \frac{1}{\lambda^2} y_v + \sum_{k=1}^{K} \beta_{kv} \Big( \Upsilon(\bar{\beta}_k + n_k^{\mathrm{K}}) - \Upsilon(\bar{\beta}_k) + \\ + \left\{n_{kv}^{\mathrm{KV}} > 0\right\} \left[\Upsilon(\beta_{kv}) - \Upsilon(\beta_{kv} + n_{kv}^{\mathrm{KV}})\right] \Big).$$

## 6.4 Experimental Results

To demonstrate the usefulness of our model we applied it to a multi-lingual document collection, where we can show a substantial improvement over the standard LDA model on the coordination between topics of different languages.

### 6.4.1 Dataset

Since our goal is to compare topic distributions on different languages we used a parallel corpus [123] with the proceedings of the European Parliament in 11 languages. We focused on two language pairs: English/French and English/Portuguese.

Note that a parallel corpus is *not* necessary for the application of the proposed model – it is being used here only because it allows us to properly evaluate the effectiveness of our model.[3]

We treated the transcript of each speaker in each session as a document, since in the European Parliament proceedings different speakers usually talk about different topics. We randomly sampled 1000 documents from each language, removed infrequent[4] and frequent[5] words and kept only the documents with at least 20 words. Finally, we removed all documents that lost their corresponding translations in this process. After this preprocessing we were left with 2415 documents, 805 in each language, and a vocabulary size of 23883 words.

### 6.4.2 Baselines

We compared our model to standard LDA, learning $\alpha$ and $\beta$, both asymmetric (that is, we don't assume all coordinates of $\alpha$ and $\beta$ are identical).

### 6.4.3 Prior

We imposed the graph-based prior mentioned in Section 6.2.2. To build our similarity graph we used the English-French and English-Portuguese dictionaries from `http://wiki.webz.cz/dict/`, augmented with translations from Google Translate for the most frequent words in our dataset. As described earlier, each word corresponds to a vertex, with an edge whenever two words match in the dictionary. Here all edges have a fixed weight of one.

In our model $\beta = \exp(y_{kv} + y_v)$, so we want to keep both $y_{kv}$ and $y_v$ reasonably low to avoid numerical problems, as a large value of either would lead to overflow. We ensure that by setting $\lambda$, the standard deviation of their prior, to one in all experiments. We did the same for the standard LDA model, where to learn an asymmetric beta we simply removed $y_{kv}$ to obtain $\beta = \exp(y_v)$.

[3]To emphasise this point, later in this section we show experiments with non-parallel corpora, in which case we have to rely on visual inspection to assess the outcomes.

[4]Words that occurred less than 3 times in the corpus.

[5]Words that occurred more than $M/10$ times in the corpus, where $M$ is the total number of documents.

### 6.4.4 Methodology

In our experiments we used all the English documents and a subset of the French and Portuguese ones – this is what we have in a real application, when we try to learn a topic model from web pages: the number of pages is English is far greater than in any other language.

We compared three approaches. First, we ran the standard LDA model with all documents mixed together – this is one of our baselines, which we call STD1.

Next we ran our proposed model, but with a slight modification to the setup: in the first half of the iterations of the Gibbs sampler we included only English documents; in the second half we added the French and Portuguese ones to the mix. We need to start with only one language so that an initial topic-word distribution is built; once that is done the priors are learned and can be used to guide the topic-word distributions in other languages.

Finally, as a control experiment we ran the standard LDA model in this same setting: first English documents, then all languages mixed. We call this STD2.

In all experiments we ran the Gibbs sampler for a total of 3000 iterations, with the number of topics fixed to 20, and keep the last sample. After a burn-in of 500 iterations, the optimisation over the word smoothing coefficients was done every 100 iterations, using an off-the-shelf L-BFGS [124] optimiser.[6]. We repeated every experiment 5 times with different randomisations.

### 6.4.5 Evaluation

Evaluation of topic models is an open problem – recent work [125] suggests that popular measures based on held-out likelihood, such as perplexity, do *not* capture whether topics are coherent or not. Furthermore, we need a set of measures that can assess whether or not we improved over the standard LDA model with respect to our goal – *to synchronise topics across different languages* – and there's no reason to believe that likelihood measures would assess that: a model where topics are synchronised across languages is not necessarily more likely than a model that is not synchronised. Therefore, to evaluate our model we compare the topic distributions of each English document with its corresponding French pair (and analogously for the other combinations: English/Portuguese and French/Portuguese), with these metrics:

[6] `http://www.chokkan.org/software/liblbfgs`

**Mean $\ell_2$ distance:**

$\frac{1}{|L_1|}\sum_{d_1\in L_1, d_2=F(d_1)}\left(\sum_{k=1}^{K}\left(\theta_k^{d_1}-\theta_k^{d_2}\right)^2\right)^{\frac{1}{2}}$,

where $L_1$ denotes the set of documents in the first language, $F$ a mapping from a document in the first language to its corresponding translation in the second language and $\theta^d$ the topic distribution of document $d$.

**Mean Hellinger distance:** $\frac{1}{|L_1|}\sum_{d_1\in L_1, d_2=F(d_1)}\sum_{k=1}^{K}\left(\sqrt{\theta_k^{d_1}}-\sqrt{\theta_k^{d_2}}\right)^2$.

**Agreements on first topic:** $\frac{1}{|L_1|}\sum_{d_1\in L_1, d_2=F(d_1)} I(\text{argmax}_k\,\theta_k^{d_1}, \text{argmax}_k\,\theta_k^{d_2}))$,

where $I$ is the indicator function – that is, the proportion of document pairs where the most likely topic is the same for both languages.

**Mean number of agreements in top 5 topics:**

$\frac{1}{|L_1|}\sum_{d_1\in L_1, d_2=F(d_1)} agreements(d_1, d_2)$,

where $agreements(d_1, d_2)$ is the cardinality of the intersection of the 5 most likely topics of $d_1$ and $d_2$.

### 6.4.6 Results

In Figure 6.3 we compare our method (DC) to the standard LDA model (STD1 and STD2, see section 6.4.4), for the English-French pair. In all metrics our proposed model shows a substantial improvement over the standard LDA model.

In Figures 6.4 and 6.5 we do the same for the English-Portuguese and Portuguese-French pairs, respectively, with similar results. Note that we did *not* use a Portuguese-French dictionary in any experiment.

In Figure 6.6 we plot the word smoothing prior for the English word *democracy* and its French and Portuguese translations, *démocratie* and *democracia*, for both the standard LDA model (STD1) and our model (DC), with 20% of the French and Portuguese documents used in training. In STD1 we don't have topic-specific priors (hence the horizontal line) and the word *democracy* has a much higher prior, because it happens more often in the dataset (since we have all English documents and only 20% of the French and Portuguese ones). In DC, however, the priors are topic-specific and quite similar, as this is enforced by the similarity graph.

To emphasise that we do not need a parallel corpus we ran a second experiment where we selected the same number of documents of each language, but assuring that for each document its corresponding translations are *not* in the dataset,

and trained our model (DC) with 100 topics. This could be done with any multilingual corpus, since no parallelisation is required. In this case, however, we cannot compute the distance metrics as before, since we have no information on the actual topic distributions of the documents. The best we can hope to do is to visually inspect the most likely words for the learned topics. This is shown in Table 6.2, for some selected topics, where the synchronisation amongst the different languages is clear.

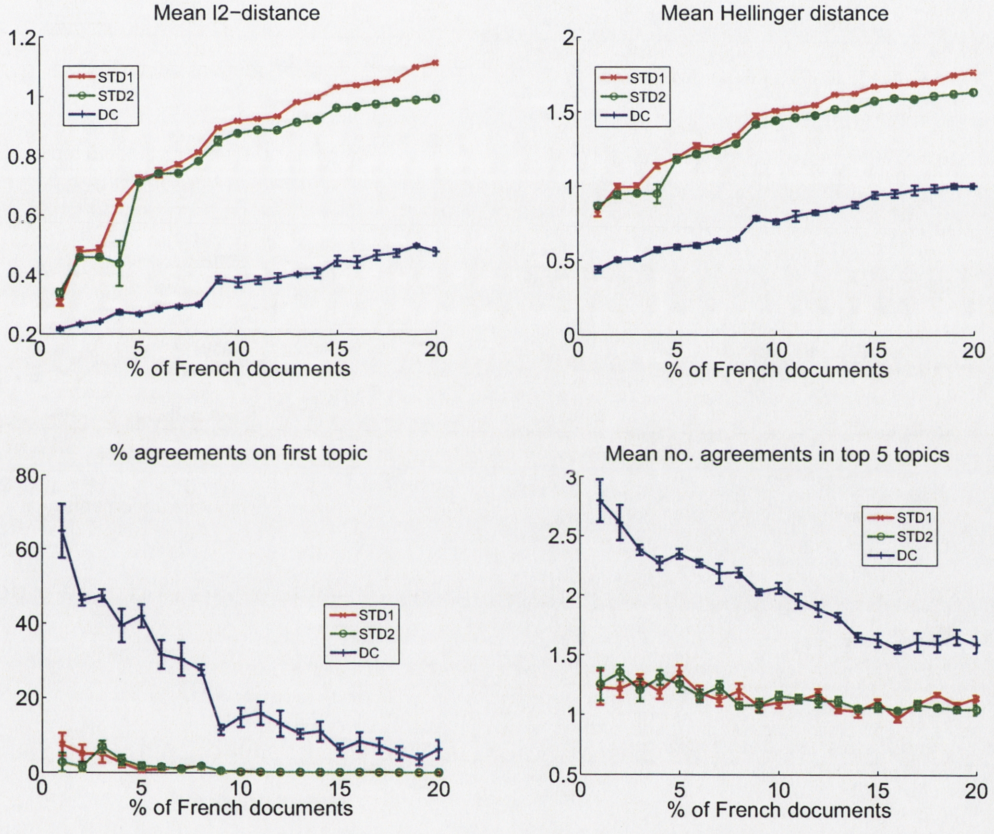



Figure 6.3: Comparison of topic distributions in English and French documents. See text for details.

## 6.5 Extensions: Other Features

Although we have implemented a specific type of feature encoding for the words, our model admits a large range of applications through a suitable choice of fea-

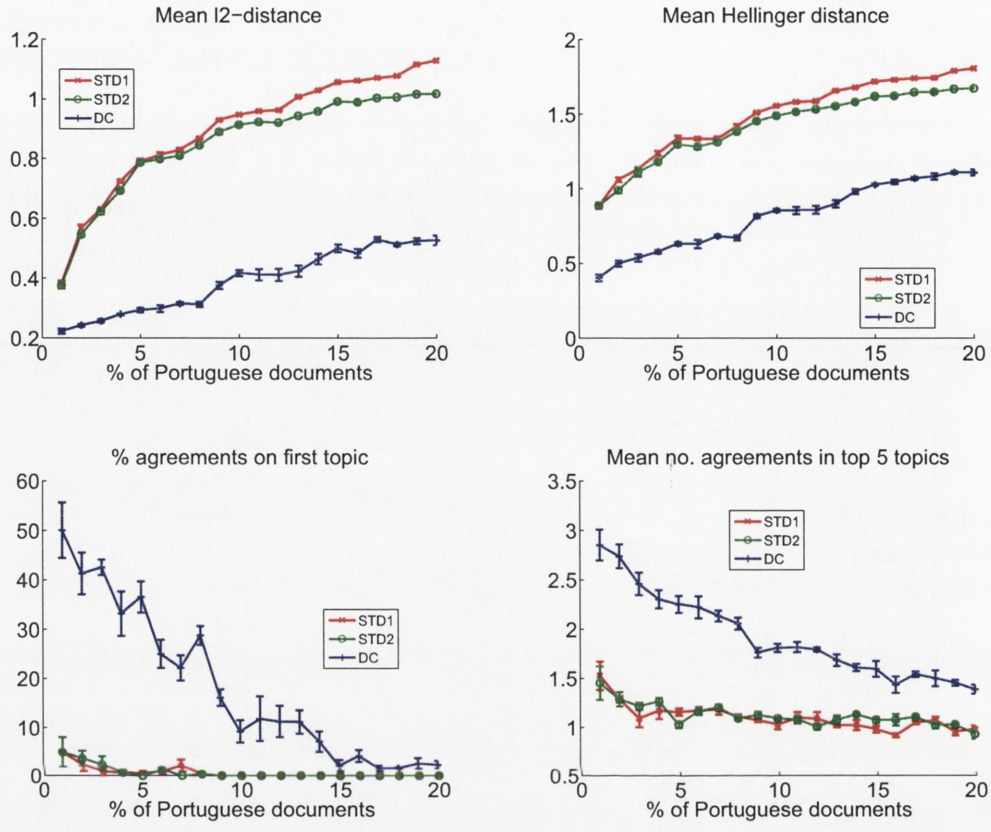



Figure 6.4: Comparison of topic distributions in English and Portuguese documents. See text.

tures. In the following we discuss a number of them in greater detail.

### 6.5.1 Single Language

#### Distributional Similarity

The basic idea is that words are similar if they occur in a similar context [126]. Hence, one could build a graph as outlined in Section 6.2.2 with edges only between words which exceed a level of proximity.

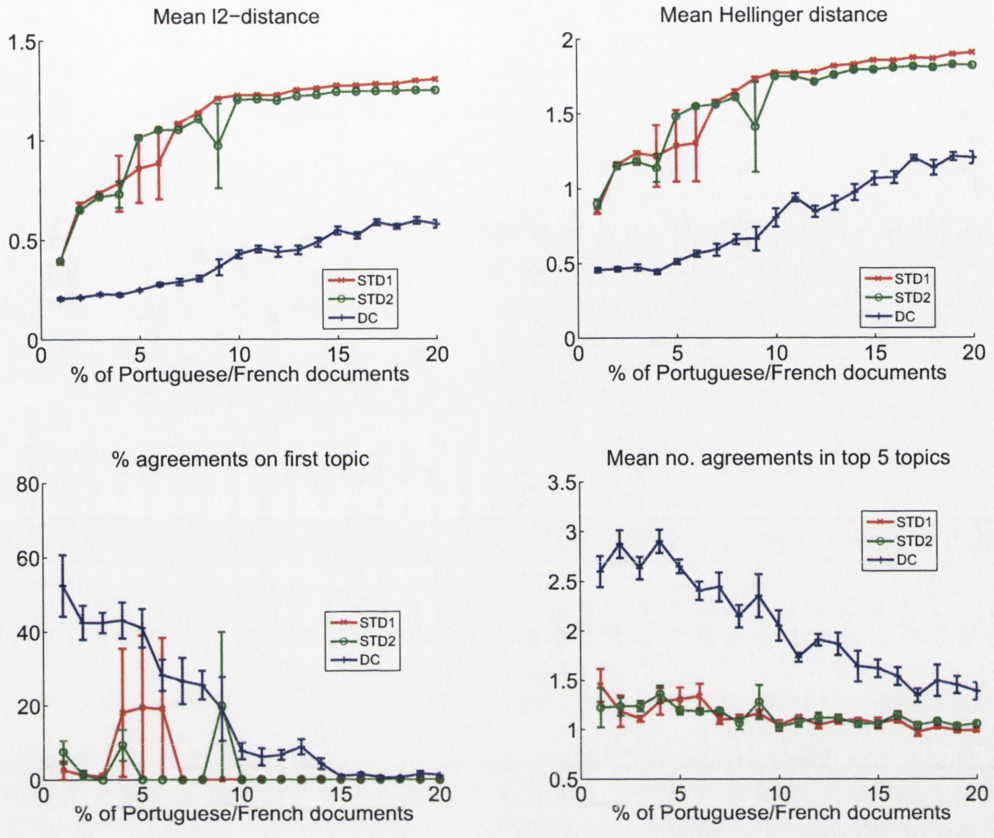



Figure 6.5: Comparison of topic distributions in Portuguese and French documents. See text.

### Lexical Similarity

For interpolation between words one could use a distribution over substrings of a word as the feature map. This is essentially what is proposed by [127]. Such lexical similarity makes the sampler less sensitive to issues such as stemming: after all, two words which reduce to the same stem will also have a high lexical similarity score, hence the estimated $\beta_{kv}$ will yield very similar topic assignments.

### Synonyms and Thesauri

Given a list of synonyms it is reasonable to assume that they belong to related topics. This can be achieved by adding edges between a word and all of its synonyms. Since in our framework we only use this information to shape a *prior*,

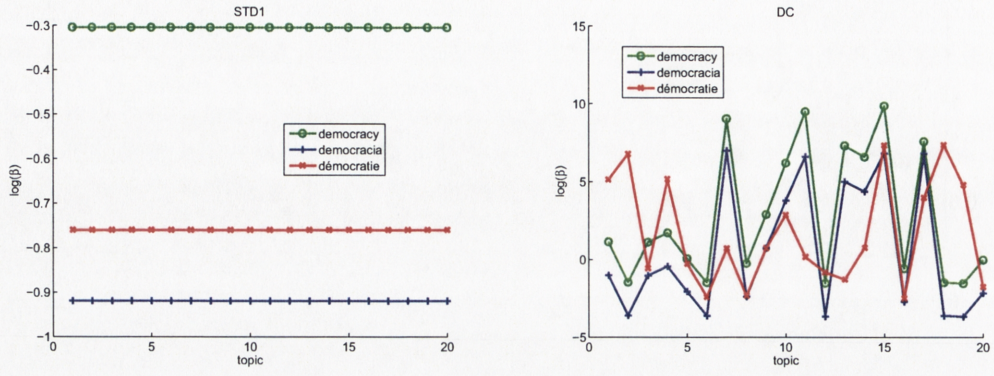



Figure 6.6: Word smoothing prior for two words in standard LDA and in our model. The x-axis is the index of the topic. See text for details.

errors in the synonym list and multiple meanings of a word will not prove fatal.

### 6.5.2 Multiple Languages

#### Lexical Similarity

Similar considerations apply for inter-lingual topic models. It is reasonable to assume that lexical similarity generally implies similarity in meaning. Using such features should allow one to synchronise topics even in the absence of dictionaries. However, it is important that similarities are not hardcoded but only imposed as a prior on the topic distribution (e.g., 'gift' has different meanings in English and German).

## 6.6 Scalability

In Figure 6.7 (left) we plot run times for the experiments in Figures 6.3-6.5, where we can see that DC is approximately 2.5 times slower than standard LDA.

We also plot run times for DC with different numbers of topics (Figure 6.7, centre) and different numbers of documents (Figure 6.7, right), and as can be seen it scales linearly with the number of topics and sub-linearly with the number of documents.

Table 6.2: Top 10 words for some of the learned topics. Words are coloured according to their language – English, Portuguese or French – except when ambiguous (e.g., *information* is a word in both French and English). See text for details.

| Topic 8 | Topic 17 | Topic 20 | Topic 32 | Topic 49 |
|---|---|---|---|---|
| amendments | élections | informação | stability | monnaie |
| alterações | electoral | information | coordination | consumers |
| amendment | elections | regiões | estabilidade | consumidores |
| amendements | députés | société | central | consommateurs |
| alteração | eleições | l'information | coordenação | l'euro |
| use | partis | acesso | plans | crois |
| substances | proportional | aeroplanes | objectivo | s'agit |
| règlement | eleitoral | prix | stabilité | moeda |
| l'amendement | transnational | régions | ue | pouvoir |
| accept | scrutin | comunicação | list | currency |

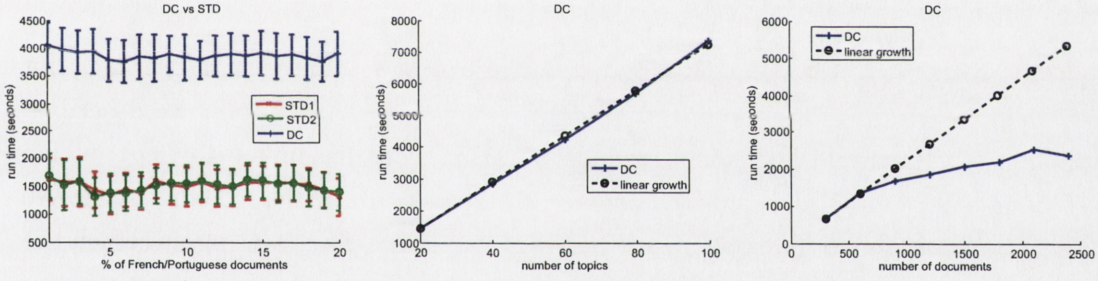



Figure 6.7: Left: run times for the experiments in Figures 6.3-6.5. Centre: run times versus number of topics for downstream conditioning. Right: run times versus number of documents for downstream conditioning.

## 6.7 Fast Sampling

As we mentioned in Section 2.2.3, recently [36] showed how to break LDA's sampling into three components and leverage the resulting sparsity in $k$ of some of them to significantly speed up the sampling process. We can do the same here, by decomposing (6.6) into three terms:

$$p(z_{mn} = k|\text{rest}) \propto \underbrace{\beta_{kv}\frac{\alpha_k}{n^{\mathrm{K}}_{k-} + \bar{\beta}_k}}_{:=A_k} + \underbrace{\beta_{kv}\frac{n^{\mathrm{KM}}_{km-}}{n^{\mathrm{K}}_{k-} + \bar{\beta}_k}}_{:=B_k} + \underbrace{\frac{n^{\mathrm{KV}}_{kv_{mn}-}\left[n^{\mathrm{KM}}_{km-} + \alpha_k\right]}{n^{\mathrm{K}}_{k-} + \bar{\beta}_k}}_{:=C_k}.$$

Unlike in the case of standard LDA [30], some of the normalisations are now dependent on the *word* in question (via the $\beta$ smoother). It is therefore not possible to apply the sampler of [36] directly, since all terms now depend on both the topics and the words via $\beta_{kv}$. We modify it as follows: $A := \sum_k A_k$ only depends on $w_{mn}$ in a *multiplicative* fashion via $\beta_{kv}$ and it is constant throughout the document otherwise. In $B := \sum_k B_k$ only two terms need updating whenever we reassign a word to a new topic (and we have a new multiplicative constant for different words via $\beta_{kv}$). Hence, the only term that needs to be fully recomputed for each word is $C := \sum_k C_k$. Fortunately, this is nonzero for only a small number of topics for a given word.

## 6.8 Summary

Estimating topic distributions for documents is a well-known IR problem, for which one of the most popular approaches is LDA. In this chapter we described a simple yet general formalism for incorporating word features into LDA, which among other things allows us to synchronise topics across different languages. We performed a number of experiments in the multiple-language setting, in which the goal was to show that our model is able to incorporate dictionary information in order to improve topic alignment across different languages. Our experimental results revealed substantial improvement over the LDA model in the quality of topic alignment, as measured by several metrics, and in particular we obtained much improved topic alignment even across languages for which a dictionary is not used (as described in the Portuguese/French plots, see Figure 6.5). We also showed that the algorithm is quite effective even in the *absence* of documents that are explicitly denoted as being aligned (see Table 6.2). This sets it apart from [118], which requires that a significant fraction (at least 25%) of documents are paired. Also, the model is not limited to lexical features. Instead, we could for instance also exploit syntactical information such as parse trees. For instance, noun / verb disambiguation or named entity recognition are all useful in determining the meaning of words and therefore it is quite likely that they will also aid in obtaining an improved topic mixture model.

# Chapter 7

# Conclusion

> In 2007, for the first time ever, more information was generated in one year than had been produced in the entire previous five thousand years - the period since the invention of writing.
>
> Jaap Bloem, Menno van Doorn and Sander Duivestein, 2009.

Information retrieval is a field comprised of several different methods and techniques which are often developed from sound but heuristic arguments, and therefore there is little understanding about why and when which methods work or not. In this thesis we took a fresh view into several information retrieval problems from the perspective of structured learning, providing principled strategies for attacking such problems.

In Chapters 3 and 4 we focused on the multi-label classification problem. The task there was to predict a set of labels associated with an instance – for example, predicting the tags associated with a document or image. Our main contribution in Chapter 3 was to propose a predictor that can be optimised for a specific evaluation measure, chosen from a set of possible measures that includes one of the most relevant ones for information retrieval: the F-score. In Chapter 4 we extended this predictor to make use of label interactions, improving its performance.

In Chapter 5 we shifted focus to graph matching predictors, addressing one particular application of them: page ranking. The task there was to, given a query and a set of documents related to it, rank the documents according to their relevance. Our main contribution in that chapter was a novel method for learning max-weight bipartite matching predictors based on an exponential family model which, unlike previous work, is fully probabilistic and therefore can easily be incorporated as a module in larger probabilistic models.

In Chapter 6 we addressed the problem of topic modeling. The task in that case was to find compact representations of documents by learning latent topics and associating each document with a topic distribution. Our main contribution was an extension of Latent Dirichlet Allocation to allow for the encoding of side information in the distribution over words. While this has several potential applications, we focused on the alignment of topics in multilingual data sets.

## 7.1 Future Directions

There are a number of possible extensions for the work described in this thesis.

On the multi-label classification problem we could try to extend the method to support other measures, such as area under curve (AUC) and the set of micro-measures: micro-precision, micro-recall, micro-$F_1$ and the generalisation of these, micro-$F_\beta$ (existing work on micro-measures [49] is restricted to binary labels). We could also extend the work to the interactive setting: that is, we assume a subset of the labels are known and have as a task to predict the remaining ones [7].

On the ranking problem, one possible focus would be on finding more efficient ways of solving large problems. Two possible and independent lines of research for that would be to exploit the data sparsity in the permutation group [128, 129] and to apply pseudo-likelihood estimators [130]. Another possibility would be to apply the so called *kernel trick* [131] to obtain a non-linear version of our method, potentially improving its performance.

On the topic modeling problem we could work on the application side: so far we have explored just one type of word feature – interlingual dictionaries – but there are many others that we could try, such as lexical similarity, synonyms and thesauri, each one with potentially many applications.

# Appendix A

# Adding Positivity Constraints to BMRM

BMRM ([55]) approximates the solution to an optimisation problem of the form

$$\underset{\theta}{\text{minimize}} \left[ R_{emp}(\theta) + \frac{\lambda}{2} \|\theta\|^2 \right], \tag{A.1a}$$

$$\text{where } R_{emp}(\theta) = \frac{1}{N} \sum_{n=1}^{N} l(x_n, y_n, \theta), \tag{A.1b}$$

by iteratively solving a linear program arising from a lower bound based on a first-order Taylor approximation of $l$. Denoting $a_{i+1} := \partial_\theta R_{emp}(\theta_i)$ and $b_{i+1} := R_{emp}(\theta_i) - \langle a_{i+1}, \theta_i \rangle$, where $i$ corresponds to the iteration number, the problem that is solved is

$$\underset{\theta,\xi}{\text{minimize}} \frac{\lambda}{2} \|\theta\|^2 + \xi \tag{A.2a}$$

$$\text{subject to } \langle a_j, \theta \rangle + b_j \leq \xi \text{ for all } j \leq i \text{ and } \xi \geq 0. \tag{A.2b}$$

This is done by, at each iteration, solving the dual of (A.2):

$$\underset{\alpha}{\text{maximize}} -\frac{1}{2\lambda} \alpha^T A^T A \alpha + \alpha^T b \tag{A.3a}$$

$$\text{s.t. } \mathbf{1}^T \alpha \leq 1, \quad \alpha \geq \mathbf{0}, \tag{A.3b}$$

where $A$ denotes the matrix $[a_1 a_2 \dots a_i]$, $b$ the vector $[b_1 b_2 \dots b_i]^T$ and $\theta = -\frac{1}{\lambda} A\alpha$.

Enforcing positivity in $\theta^2$ amounts to adding the additional constraints $\theta^2 \geq \mathbf{0}$ to (A.2), and the corresponding dual problem now becomes:

$$\underset{\alpha,\gamma}{\text{maximize}} -\frac{1}{2\lambda}\left(\alpha^T A^T A\alpha + \gamma^T\gamma - 2\alpha^T A_1^T\gamma\right) + \alpha^T b \tag{A.4a}$$

$$\text{s.t. } \mathbf{1}^T\alpha \leq 1, \quad \alpha \geq \mathbf{0}, \quad \gamma \geq \mathbf{0}, \tag{A.4b}$$

where $A_1$ is the subset of the rows of $A$ that correspond to the gradient with respect to $\theta^2$.

# Appendix B

# Graph-cuts

Here we describe a way of solving (4.2) via graph-cuts. We want to solve

$$\bar{y} \in \underset{y \in \mathcal{Y}}{\operatorname{argmax}} y^T A y \tag{B.1}$$

where $A$ is a $V \times V$ upper-triangular matrix with non-negative off-diagonal elements, and $y$ is a $V \times 1$ binary vector ($y \in \{0,1\}^V$). We first change this to the equivalent minimisation problem

$$\bar{y} \in \underset{y \in \mathcal{Y}}{\operatorname{argmin}} -y^T A y \tag{B.2}$$

and then, using subscripts to index elements of $y$ and $A$, we define the energy function

$$E(y) = \sum_{i=1}^{V} E^i(y_i) + \sum_{i=1}^{V-1} \sum_{j=i+1}^{V} E^{ij}(y_i, y_j), \text{ where} \tag{B.3}$$

$$E^i(y_i) = -y_i A_{ii} \text{ and} \tag{B.4}$$

$$E^{ij}(y_i, y_j) = -y_i y_j A_{ij} \tag{B.5}$$

and rewrite (B.2) as an energy minimisation problem:

$$\bar{y} \in \underset{y \in \mathcal{Y}}{\operatorname{argmin}} E(y). \tag{B.6}$$

The graph-cut technique amounts to constructing a specialised graph for this energy function such that the minimum cut[1] on the graph (which can be computed efficiently with max flow algorithms [132, 70]) also minimises the energy. For that end we make use of the work of Kolgomorov and Zabih [133], which provided

[1] A cut is a partition of the vertices of a graph into two disjoint subsets.

a characterisation of the energy functions that can be minimised as well as a general-purpose graph construction technique.

From eq. (7) of [133], (B.6) is graph-representable[2] if and only if each term of $E^{ij}$ satisfies the inequality

$$E^{ij}(0,0) + E^{ij}(1,1) \leq E^{ij}(0,1) + E^{ij}(1,0). \tag{B.7}$$

Substituting (B.5) in (B.7) we get the condition

$$\forall i,j : i < j, A_{ij} \geq 0. \tag{B.8}$$

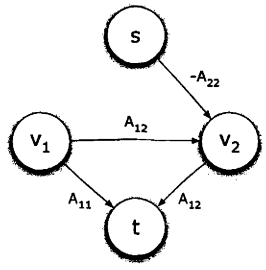
This is the reason why we need to enforce non-negativity on the off-diagonal of $A$. Following the construction technique of [133], we create a graph with $V+2$ nodes: a source $s$, a sink $t$ and one node $v_i$ for each one of the $V$ elements of $y$. We then add:[3]

- For each $i \in \{1 \ldots V\} : A_{ii} \leq 0$, an edge from $s$ to $v_i$ with weight $-A_{ii}$.
- For each $i \in \{1 \ldots V\} : A_{ii} > 0$, an edge from $v_i$ to $t$ with weight $A_{ii}$.
- For each $i,j : i > j$, an edge from $v_i$ to $v_j$ and another from $v_j$ to $t$, both with weight $A_{ij}$.

Note that by construction all edge weights are non-negative. As a minimal example, consider the following $2 \times 2$ example matrix $A$, with $A_{11} \geq 0$ and $A_{22} < 0$:

| | |
|---|---|
| $A_{11}$ | $A_{12}$ |
| 0 | $A_{22}$ |

the corresponding graph will be:




Having constructed the graph we find the minimum cut and assign:

- $y_i = 0$ for each $i$ such that $v_i$ is in the same side of the cut as the source $s$.
- $y_i = 1$ for each $i$ such that $v_i$ is in the same side of the cut as the sink $t$.

[2]For the definition of graph-representability please see [133].

[3]For the theory behind this construction we refer the reader to [133].

# Appendix C

# Approximating the Permanent

For completeness we include a description of the sampling algorithm presented in [103]. The algorithm is an *accept-reject* algorithm. The core idea of such an algorithm is very simple: assume we need to sample from a distribution $p$ in a given domain $\mathcal{M}$, but that such a task is intractable. Instead, we sample from a distribution $q$ in a *superset* $\mathcal{N}$ of the original domain (in which sampling is easier), whose restriction to the original domain coincides with the original distribution: $q|_{\mathcal{N}} = p$. We then only 'accept' those samples that effectively fall within the original domain $\mathcal{M}$. Clearly, the efficiency of such a procedure will be dictated by (i) how efficient it is to sample from $q$ in $\mathcal{N}$ and (ii) how much mass of $q$ is in $\mathcal{M}$. Roughly speaking, the algorithm presented in [103] manages to sample perfect matches of bipartite graphs such that both conditions (i) and (ii) are favourable.

The reasoning goes as follows: the problem consists of generating variates $y \in \mathcal{Y}$ ($y$ is a match) with the property that $p(y) = w(y)/Z$, where $w(y)$ is the non-negative score of match $y$ and $Z = \sum_y w(y)$ is the partition function, which in our case is a permanent as discussed in Section 5.3.1. We first partition the space $\mathcal{Y}$ into $\mathcal{Y}_1, \ldots, \mathcal{Y}_I$, where $\mathcal{Y}_i = \{y : y(1) = i\}$. Each part has its own partition function $Z_i = \sum_{y \in \mathcal{Y}_i} w(y)$. Next, a suitable upper bound $U(\mathcal{Y}_i) \geq Z_i$ on the partition function is constructed such that the following two properties hold:[1]

$$\text{(P1)} \qquad \sum_{i=1}^{M} U(\mathcal{Y}_i) \leq U(\mathcal{Y}).$$

$$\text{(P2)} \qquad \text{If } |\mathcal{Y}_i| = 1, \text{ then } U(\mathcal{Y}_i) = Z_i = w(y).$$

That is, (i) the upper bound is super-additive in the elements of the partition and (ii) if $\mathcal{Y}_i$ has a single match, the upper bound *equals* the partition function,

[1] See [103] for details.

which in this case is just the score of that match.

Now the algorithm: consider the random variable $\mathcal{I}$ where $p(\mathcal{I} = i) = U(\mathcal{Y}_i)/U(\mathcal{Y})$. By (P1), $\sum_{i=1}^{M} p(i) \leq 1$, so assume $p(\mathcal{I} = 0) = 1 - \sum_{i=1}^{M} p(i)$. Now, draw a variate from this distribution, and if $\mathcal{I} = i = 0$, reject and restart, otherwise recursively sample in $\mathcal{Y}_i$.[2] This algorithm either stops and restarts or it reaches $\mathcal{Y}_{\text{final}}$ which consists of a match, i.e., $|\mathcal{Y}_{\text{final}}| = 1$. This match is then a legitimate sample from $p(y)$. The reason this is the case is because of (P2), as shown below. Assuming the algorithm finishes after $k$ samples, the probability of the match is the telescopic product

$$\frac{U(\mathcal{Y}_{\mathcal{I}(1)})}{U(\mathcal{Y})} \frac{U(\mathcal{Y}_{\mathcal{I}(2)})}{U(\mathcal{Y}_{\mathcal{I}(1)})} \cdots \frac{U(\mathcal{Y}_{\mathcal{I}(k)})}{U(\mathcal{Y}_{\mathcal{I}(k-1)})} \overset{(P2)}{=} \frac{w(y)}{U(\mathcal{Y})}, \tag{C.1}$$

and since the probability of acceptance is $Z/U(\mathcal{Y})$, we have

$$p(y) = \frac{w(y)/U(\mathcal{Y})}{Z/U(\mathcal{Y})} = \frac{w(y)}{Z}, \tag{C.2}$$

which is indeed the distribution from which we want to sample. For pseudocode and a rigorous presentation of the algorithm, see [103].

[2]Due to the self-reducibility of permutations, when we fix $y(1) = i$, what remains is also a set of permutations. We then sample $y(2), y(3) \ldots y(M)$.

# Bibliography


[1] Youtube: Two days worth of video uploaded every minute. http://mashable.com/2011/05/25/youtube-6-birthday/. (Cited on page 1.)

[2] Netcraft's October/2011 Web Server Survey. http://news.netcraft.com/archives/2011/10/06/october-2011-web-server-survey.html. (Cited on page 1.)

[3] Wikipedia:modelling wikipedia's growth. http://en.wikipedia.org/wiki/Wikipedia:Modelling_Wikipedia's_growth. (Cited on page 1.)

[4] CNN fortune 500 companies (2010). http://money.cnn.com/magazines/fortune/fortune500/2010/performers/companies/biggest/. (Cited on page 1.)

[5] The PASCAL Visual Object Classes Homepage. http://pascallin.ecs.soton.ac.uk/challenges/VOC/. (Cited on page 2.)

[6] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval.* Cambridge University Press, 2008. (Cited on pages 2, 3, and 77.)

[7] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. TagProp: Discriminative Metric Learning in Nearest Neighbor Models for Image Auto-Annotation. In *Proceedings of the International Conference on Computer Vision*, 2009. (Cited on pages 2, 23, 38, and 96.)

[8] Pia Borlund. The concept of relevance in IR. *Journal of the American Society for Information Science and Technology*, 54(10):913–925, 2003. (Cited on page 2.)

[9] Peter Ingwersen. Cognitive perspectives of information retrieval interaction: elements of a cognitive IR theory. *Journal of Documentation*, 52(1):3–50, 1996. (Cited on page 2.)

[10] SIGIR 2009 Workshop on Learning to Rank for Information Retrieval. http://research.microsoft.com/en-us/um/beijing/events/lr4ir-2009/. (Cited on pages 2 and 57.)

[11] Yahoo! Learning to Rank Challenge Workshop. http://learningtorankchallenge.yahoo.com/workshop.php. (Cited on pages 2 and 57.)

[12] LETOR: Learning to Rank for Information Retrieval. http://research.microsoft.com/en-us/um/beijing/projects/letor/. (Cited on page 2.)

[13] Susan T. Dumais. Using LSI for information filtering: TREC-3 experiments. In *The third Text REtrieval Conference (TREC3)*, 1995. (Cited on pages 2 and 77.)

[14] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999. (Cited on pages 3 and 77.)

[15] Quan Wang, Jun Xu, Hang Li, and Nick Craswell. Regularized latent semantic indexing. In *Proceedings of the 34th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 685–694, 2011. (Cited on pages 3 and 77.)

[16] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005. (Cited on pages 8, 9, 10, 11, 24, 40, and 58.)

[17] Ben Taskar. *Learning Structured Prediction Models: A Large Margin Approach.* PhD thesis, Stanford University, 2004. (Cited on pages 8, 58, and 59.)

[18] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, volume 18, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann. (Cited on page 8.)

[19] Vladimir Vapnik. *Statistical Learning Theory.* John Wiley and Sons, New York, 1998. (Cited on page 9.)

[20] Martin Wainwright and Michael Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Department of Statistics, September 2003. (Cited on pages 12, 13, 64, and 65.)

[21] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009. (Cited on page 14.)

[22] J. Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the American Association of Artificial Intelligence National Conference on AI*, pages 133–136, 1982. (Cited on page 14.)

[23] Stuart Gem and Donald Gem. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984. (Cited on page 15.)

[24] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian data analysis.* CRC press, 2004. (Cited on page 15.)

[25] Fei-Fei Li and Pietro Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 524–531. Ieee, 2005. (Cited on page 15.)

[26] Gabriel Doyle and Charles Elkan. Financial topic models. Presented at the NIPS 2009 workshop on Applications for Topic Models: Text and Beyond, 2009. (Cited on page 15.)

[27] Simon Rogers, Mark Girolami, Colin Campbell, and Rainer Breitling. The Latent Process Decomposition of cDNA Microarray Data Sets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2:143–156, April 2005. (Cited on page 15.)

[28] Diane J. Hu and Lawrence K. Saul. A probabilistic topic model for music analysis. Presented at the NIPS 2009 workshop on Applications for Topic Models: Text and Beyond, 2009. (Cited on page 15.)

[29] Xing Wei and W. Bruce Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185. ACM, 2006. (Cited on page 15.)

[30] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003. (Cited on pages 16, 17, 77, 80, and 94.)

[31] Wray L. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994. (Cited on page 17.)

[32] Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 352–359, 2002. (Cited on page 17.)

[33] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004. (Cited on pages 17, 18, 81, and 83.)

[34] Gregor Heinrich. Parameter estimation for text analysis. Technical report, 2004. (Cited on pages 17 and 18.)

[35] Yee Whye Teh, David Newman, and Max Welling. Collapsed variational bayesian inference algorithm for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 19pp:1378–1385, 2006. (Cited on page 17.)

[36] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009. (Cited on pages 17, 19, 85, 93, and 94.)

[37] Andrew McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI Workshop on Text Learning*, 1999. (Cited on pages 23 and 38.)

[38] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Annual ACM Conference on Research and Development in Information Retrieval*, pages 274–281, 2005. (Cited on page 23.)

[39] Douglas W. Oard and Jason R. Baron. Overview of the TREC 2008 Legal Track. (Cited on page 23.)

[40] Linli Xu, Martha White, and Dale Schuurmans. Optimal reverse prediction. *Proceedings of the International Conference on Machine Learning*, pages 1137–1144, 2009. (Cited on page 24.)

[41] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. *Mining Multi-label Data.* Springer, 2009. (Cited on pages 24, 35, 36, and 52.)

[42] Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning and Knowledge Discovery in Databases*, pages 254–269, 2009. (Cited on pages 24, 25, 35, 36, and 53.)

[43] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In *Proceedings of the International Conference on Machine Learning*, 2010. (Cited on pages 24 and 36.)

[44] Grigorios Tsoumakas and Ioannis P. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the 18th European Conference on Machine Learning*, pages 406–417, Warsaw, Poland, 2007. (Cited on pages 25, 27, 35, 36, and 52.)

[45] Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 995–1000, Washington, DC, USA, 2008. IEEE Computer Society. (Cited on pages 25 and 35.)

[46] Piyush Rai and Hal Daume. Multi-Label Prediction via Sparse Infinite CCA. In *Advances in Neural Information Processing Systems 22*, pages 1518–1526. 2009. (Cited on pages 25 and 35.)

[47] Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30. Springer, 2004. (Cited on pages 25 and 35.)

[48] Martin Jansche. Maximum expected F-measure training of logistic regression models. *Proceedings of the conference on Human Language Technology*

*and Empirical Methods in Natural Language Processing*, pages 692–699, 2005. (Cited on page 25.)

[49] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning*, pages 377–384, San Francisco, California, 2005. Morgan Kaufmann Publishers. (Cited on pages 25 and 96.)

[50] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer, 1999. (Cited on pages 25 and 69.)

[51] Chuan Shi, Xiangnan Kong, Philip Yu, and Bai Wang. Multi-label Ensemble Learning. *Machine Learning and Knowledge Discovery in Databases*, 6913:223–239, 2011. (Cited on page 25.)

[52] Elean Montañés, José Ramón Quevedo, and Juan José del Coz. Aggregating Independent and Dependent Models to Learn Multi-label Classifiers. *Machine Learning and Knowledge Discovery in Databases*, pages 484–500, 2011. (Cited on page 25.)

[53] Wei Bi and James Kwok. Multi-Label Classification on Tree- and DAG-Structured Hierarchies. In *Proceedings of the International Conference on Machine Learning*, 2011. (Cited on pages 25 and 41.)

[54] Donald E. Knuth. *The Art of Computer Programming: Fundamental Algorithms*, volume 1. Addison-Wesley, Reading, Massachusetts, second edition, 1998. (Cited on page 30.)

[55] Choon Hui Teo, S. V. N. Vishwanathan, Alex J. Smola, and Quoc V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010. (Cited on pages 30, 34, 44, 45, 53, and 97.)

[56] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999. (Cited on page 35.)

[57] Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, July 2007. (Cited on pages 36, 52, and 53.)

[58] Thomas Mensink, Jakob Verbeek, and Gabriela Csurka. Learning structured prediction models for interactive image labeling. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, United States, June 2011. (Cited on page 39.)

[59] Andrew McCallum. Multi-label text classication with a mixture model trained by EM. In *AAAI Workshop on Text Learning*, 1999. (Cited on page 40.)

[60] Lijuan Cai and Thomas Hofmann. Exploiting known taxonomies in learning overlapping concepts. In *Proceedings of International Joint Conferences on Artificial Intelligence*, pages 714–719, 2007. (Cited on page 41.)

[61] Claudio Gentile and Luca Zaniboni. Incremental Algorithms for Hierarchical Classification. *Journal of Machine Learning Research*, 7:31–54, 2006. (Cited on page 41.)

[62] Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-taylor. Kernel-Based Learning of Hierarchical Multilabel Classification Models . *Journal of Machine Learning Research*, 7:1601–1626, 2006. (Cited on page 41.)

[63] Zafer Barutcuoglu, Robert E. Schapire, and Olga G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics (Oxford, England)*, 22(7):830–6, April 2006. (Cited on page 41.)

[64] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73:185–214, November 2008. (Cited on page 41.)

[65] Nadia Ghamrawi and Andrew Mccallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM, 2005. (Cited on page 41.)

[66] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems*, 2003. (Cited on page 41.)

[67] Bharath Hariharan, S. V. N. Vishwanathan, and Manik Varma. Large Scale Max-Margin Multi-Label Classification with Prior Knowledge about

Densely Correlated Labels. In *Proceedings of the International Conference on Machine Learning*, 2010. (Cited on page 41.)

[68] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 697–706, Washington, DC, USA, 2008. IEEE Computer Society. (Cited on page 45.)

[69] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary MRFs via extended roof duality. In *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, June 2007. (Cited on page 46.)

[70] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. (Cited on pages 53 and 99.)

[71] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, New Jersey, 1982. (Cited on pages 58 and 59.)

[72] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002. (Cited on page 58.)

[73] Tiberio Caetano, Julian J. McAuley, Li Cheng, Quoc V. Le, and Alex J. Smola. Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):1048–1058, 2009. (Cited on pages 58, 59, 61, and 72.)

[74] Quoc Le and Alex J. Smola. Direct optimization of ranking measures. *Journal of Machine Learning Research*, 2007. submitted. (Cited on pages 58, 59, 61, 62, 67, and 69.)

[75] Tony Jebara and Vlad Shchogolev. B-matching for spectral clustering. In *Proceedings of the European Conference on Machine Learning*, 2006. (Cited on page 58.)

[76] Bert Huang and Tony Jebara. Loopy belief propagation for bipartite maximum weight b-matching. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2007. (Cited on pages 58 and 59.)

[77] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987. (Cited on page 59.)

[78] Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 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. (Cited on page 61.)

[79] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference*, pages 109–126, 1994. (Cited on page 60.)

[80] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *ACM Special Interest Group in Information Retrieval (SIGIR)*, pages 275–281. ACM, 1998. (Cited on page 60.)

[81] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132, Cambridge, MA, 2000. MIT Press. (Cited on pages 61 and 69.)

[82] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003. (Cited on pages 61 and 69.)

[83] Chris Burges, Tai Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hulldender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning*, 2005. (Cited on pages 61 and 62.)

[84] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the International Conference on Machine Learning*, pages 129–136, New York, NY, USA, 2007. ACM. (Cited on pages 62 and 69.)

[85] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398, New York, NY, USA, 2007. ACM Press. (Cited on pages 62 and 69.)

[86] Ming feng Tsai, Tie yan Liu, Tao Qin, Hsin hsi Chen, and Wei ying Ma. Frank: A ranking method with fidelity loss. In *Proceedings of the 30th Annual International ACM SIGIR Conference*, 2007. (Cited on pages 62 and 69.)

[87] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in Neural Information Processing Systems 20*, pages 1697–1704. MIT Press, Cambridge, MA, 2008. (Cited on pages 62 and 69.)

[88] Zhaohui Zheng, Hongyuan Zha, and Gordon Sun. Query-level learning to rank using isotonic regression. In *Learning to Rank for Information Retrieval*, 2008. (Cited on pages 62 and 69.)

[89] Leonardo Rigutini, Tiziano Papini, Marco Maggini, and Franco Scarselli. Sortnet: Learning to rank by a neural-based sorting algorithm. In *Learning to Rank for Information Retrieval*, 2008. (Cited on pages 62 and 69.)

[90] Jim C. Huang and Brendan J. Frey. Cumulative distribution networks and the derivative-sum-product algorithm. In D.A. McAllester and P. Myllymäki, editors, *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, pages 290–297. AUAI Press, 2008. (Cited on pages 62 and 69.)

[91] Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, De-Sheng Wang, and Hang Li. Global ranking using continuous conditional random fields. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*. 2009. (Cited on pages 62 and 69.)

[92] Maksims N. Volkovs and Richard S. Zemel. Boltzrank: Learning to maximize expected ranking gain. In *Proceedings of the International Conference on Machine Learning*, 2009. (Cited on page 62.)

[93] Nicolas Usunier, David Buffoni, and Patrick Gallinari. Ranking with ordered weighted pairwise classification. In *Proceedings of the International Conference on Machine Learning*, 2009. (Cited on page 62.)

[94] B Barla Cambazoglu, Hugo Zaragoza, Olivier Chapelle, Jiang Chen, Ciya Liao, and Zhaohui Zheng. Early Exit Optimizations for Additive Machine Learned Ranking Systems. *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, pages 411–420, 2010. (Cited on page 63.)

[95] Lidan Wang and Jimmy Lin. Learning to efficiently rank. *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 138–145, 2010. (Cited on page 63.)

[96] Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Boosted multi-task learning. *Machine Learning*, December 2010. (Cited on page 63.)

[97] Taesup Moon, Lihong Li, Wei Chu, Ciya Liao, Zhaohui Zheng, and Yi Chang. Online learning for recency search ranking using real-time user feedback. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1501–1504. ACM, 2010. (Cited on page 63.)

[98] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. Towards recency ranking in web search. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 11–20. ACM, 2010. (Cited on page 63.)

[99] Henryk Minc. *Permanents*. Addison-Wesley, 1978. (Cited on page 65.)

[100] Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. Structured prediction models via the matrix-tree theorem. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2007. (Cited on page 65.)

[101] L. G. Valiant. The complexity of computing the permanent. In *Theoretical Computer Science*, volume 8, pages 189–201, 1979. (Cited on page 65.)

[102] Herbert J. Ryser. *Combinatorial Mathematics.* The Carus Mathematical Monographs, No. 14, Mathematical Association of America, 1963. (Cited on page 66.)

[103] Mark Huber and Jenny Law. Fast approximation of the permanent for very dense problems. In *Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms*, 2008. (Cited on pages 66, 73, 101, and 102.)

[104] S. Sherman. On a Theorem of Hardy, Littlewood, Polya, and Blackwell. *Proceedings of the National Academy of Sciences*, 37(12):826–831, 1951. (Cited on page 68.)

[105] Conrado Martínez. Partial quicksort. In L. Arge, G.F. Italiano, and R. Sedgewick, editors, *Proceedings of the 6th ACM-SIAM Workshop on Algorithm Engineering and Experiments and the 1st ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics*, pages 224–228. SIAM, 2004. (Cited on page 68.)

[106] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, 2007. (Cited on pages 68 and 69.)

[107] Kalervo Jarvelin and Jaana Kekalainen. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 20:2002, 2002. (Cited on page 69.)

[108] Tom Minka and Stephen Robertson. Selection bias in the letor datasets. In *Learning to Rank for Information Retrieval*, 2008. (Cited on page 70.)

[109] Serge Belongie and Jitendra Malik. Matching with shape contexts. *IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 20–26, 2000. (Cited on page 72.)

[110] Jordan Boyd-Graber and David M. Blei. Multilingual topic models for unaligned text. In *Proceedings of the 25th Conference in Uncertainty in Artificial Intelligence*, 2009. (Cited on pages 78 and 79.)

[111] Noah A. Smith and Shay B. Cohen. The Shared Logistic Normal Distribution for Grammar Induction. In *NIPS Workshop on Speech and Language:*

*Unsupervised Latent-Variable Models,*, pages 1–4, 2008. (Cited on page 79.)

[112] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. Painless Unsupervised Learning with Features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010. (Cited on page 79.)

[113] Jordan Boyd-Graber, David Blei, and Xiaojin Zhu. A Topic Model for Word Sense Disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1024–1033, 2007. (Cited on page 79.)

[114] David Andrzejewski, Xiaojin Zhu, and Mark Craven. Incorporating domain knowledge into topic modeling via Dirichlet Forest priors. In *Proceedings of the International Conference on Machine Learning*, pages 1–8. ACM Press, 2009. (Cited on page 79.)

[115] Bing Zhao and Eric P. Xing. BiTAM: Bilingual Topic AdMixture Models for Word Alignment. In *In Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, 2006. (Cited on page 79.)

[116] Woosung Kim and Sanjeev Khudanpur. Lexical triggers and latent semantic analysis for crosslingual language model adaptation. *ACM Transactions on Asian Language Information Processing*, 3, 2004. (Cited on page 79.)

[117] Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. Mining multilingual topics from wikipedia. In *18th International World Wide Web Conference*, pages 1155–1155, April 2009. (Cited on page 79.)

[118] David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 880–889, Singapore, August 2009. ACL. (Cited on pages 79 and 94.)

[119] Wim De Smet, Jie Tang, and Marie-Francine Moens. Knowledge Transfer across Multilingual Corpora via Latent Topics. *Advances in Knowledge Discovery and Data Mining*, pages 549–560, 2011. (Cited on page 80.)

[120] Charles E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 2:1152–1174, 1974. (Cited on page 80.)

[121] David M. Mimno and Andrew McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, pages 411–418. AUAI Press, 2008. (Cited on page 80.)

[122] David M. Blei and John D. Lafferty. Dynamic topic models. In W. W. Cohen and A. Moore, editors, *Proceedings of the International Conference on Machine Learning*, volume 148, pages 113–120. ACM, 2006. (Cited on page 83.)

[123] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit X*, pages 79–86, 2005. (Cited on page 85.)

[124] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989. (Cited on page 87.)

[125] Jonathan Chang, Jordan Boyd-Graber, Sean Gerrish, Chong Wang, and David Blei. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, pages 288–296. 2009. (Cited on page 87.)

[126] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619, New York, July 2002. ACM Press. (Cited on page 90.)

[127] S. V. N. Vishwanathan and Alex J. Smola. Fast kernels for string and tree matching. In *Advances in Neural Information Processing Systems 15*, pages 569–576. MIT Press, Cambridge, MA, 2003. (Cited on page 91.)

[128] Jonathan Huang. *Probabilistic Reasoning with Permutations: A Fourier-Theoretic Approach.* PhD thesis, Carnegie Mellon Universit, 2008. (Cited on page 96.)

[129] Risi Kondor and Marconi Barbosa. Ranking with kernels in fourier space. *Conference on Learning Theory*, 2010. (Cited on page 96.)

[130] Julian Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195, 1975. (Cited on page 96.)

[131] Bernhard Schölkopf and Alex Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002. (Cited on page 96.)

[132] Ender Boros, Peter L. Hammer, and Gabriel Tavares. Preprocessing of unconstrained quadratic binary optimization. *RUTCOR Research Report, RRR*, pages 10–2006, 2006. (Cited on page 99.)

[133] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, February 2004. (Cited on pages 99 and 100.)