Submodularity in Natural Language Processing: Algorithms and Applications

Hui Lin

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2012

Reading Committee:

Jeff Bilmes, Chair

Anna Karlin

Katrin Kirchhoff

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Submodularity in Natural Language Processing:
Algorithms and Applications

Hui Lin

Chair of the Supervisory Committee:
Professor Jeff Bilmes
Department of Electrical Engineering

Most natural language processing tasks can be seen as finding an optimal object from a finite set of objects. Often, the object of interest is structured, with combinatorial structures involving trees, matchings, and permutations. The combinatorial nature makes many natural language processing problems challenge to solve. On the other hand, submodularity, also known as the discrete analogous of convexity, makes many combinatorial optimization problems either tractable or approximable where otherwise neither would be possible. Whether submodularity is applicable to natural language processing problems, however, has never been studied before.

In this thesis, we fill this gap by exploring submodularity in natural language processing. We show that submodularity is practically useful for many natural language processing tasks since, in addition to giving high-quality approximate solutions to the intractable problems, it also completely captures the essence of many practical situations arises in natural language processing tasks. To do so, we demonstrate the applicability of submodular function optimization to three natural language processing tasks: word alignment for machine translation, optimal corpus creation, and document summarization.

In word alignment task, we show that submodularity naturally arises when modeling word fertility in word alignment tasks. We moreover cast word alignment problem as a submodular optimization problem over matroid constraints, which provides a brand new angle

of viewing this problem and essentially generalizes conventional matching based approaches.

In the task of optimal corpus creation, we first show that the state-of-art method corresponds to using greedy algorithm for supermodular maximization with cardinality constraint, which could perform arbitrarily poorly in theory. Alternatively, we express the problem as a minimization problem over a weighted sum of modular functions and submodular functions. We further study algorithms for general submodular function minimization, where we offer the first empirical study of the complexity of minimum-norm-point algorithm, which is widely accepted as the most practical algorithm for submodular minimization, in the scale of practical interest, and show that on a particular type of submodular functions that arises in practice, minimum-norm-point algorithm's empirical time complexity is as bad as that of the combinatorial algorithms for submodular function minimization. We moreover propose acceleration methods which speed up minimum-norm-point algorithm phenomenally in practise.

For the document summarization task, we reveal that many well-established approaches, as well as the evaluation methods, are all correspond to submodular function optimization, giving strong evidences on the fact that submodularity is a natural fit for summarization tasks. The document summarization task, therefore, can naturally be casted as budget submodular maximization problem. We propose efficient algorithm for this problem that scales well to the application, and theoretically show that the algorithm is guaranteed to find near-optimal solutions. We further introduce a class of submodular functions that is not only monotone but also models relevance and diversity simultaneously for document summarization. This class of submodular functions is then generalized to a mixture of submodular components, where each component either models the relevance or models the diversity, and differs either in function forms or in function parameters. The learning problem of submodular mixtures is also addressed, in which we show the risk of approximate learning is bounded by the risk of exact learning where exact inference is used. When evaluated on the standard benchmark task for document summarization, namely Document Understanding Conference (DUC), we achieve *best results ever reported* on DUC-2004, DUC-

2005, DUC-2006, and DUC-2007.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express sincere appreciation to my advisor, Jeff Bilmes, who continuously encouraged and supported me. He is a great mentor in many ways, one of which is his patience, which helped me out of the tough time when I lost my direction. His tireless pursuit of excellence in every aspect of academic work is truly inspirational.

Many thanks to other committee members, Anna Karlin and Katrin Kirchhoff, for their time and effort spent on this thesis. Their feedback and comments on my work are extremely helpful.

While most of the work in this thesis is published in papers coauthored with Jeff Bilmes, I would also like to express my appreciation and gratitude to all my other coauthors. Although the collaborated research does not become part of this thesis, it was indeed enjoyable and rewarding to collaborate with all of you: Shasha Xie, Alex Stupakov, Yang Liu, Li Deng, Dong Yu, Yifan Gong, Chin-Hui Lee, Jasha Droppo, Alex Acero, Stefanie Jegelka, Koby Crammer, Dimitra Vergyri, Katrin Kirchhoff, Jui-Ting Huang, Francoise Beaufays, Brian Strope, Yun-Hsuan Sung, Zhijian Ou, Ye Tian, Jian-lai Zhou, and Hui Jiang.

I would also like to thank all the former members of SSLI lab, Chris Bartels, Amar Subramanya, Jonathan Malkin, Xin Lei, Xiao Li, Kevin Duh, for the tremendous help in my early years of graduate school, and also the current members, Mei Yang, Bin Zhang, Wei Wu, Alex Marin, and Amittai Axelrod, who make SSLI a fun place to work in. Thank all the members of MELODI lab, Andrew Guillory, Galen Andrew, Ajit Singh, John Halloran, Rishabh Iyer, and Richard Rogers, for great discussions and many fun meetings with yummy food.

I would like to express my gratitude to the sources of my financial support. My

research was funded in part by an ONR MURI grant No. N000140510388, and later by an Intel Science and Technology Fellowship. Thank the Speech groups in Microsoft Research and Google Research for offering me two wonderful summer internships.

Last but not least, I would like to thank my parents for their everlasting understanding and support. To Shasha, thank you for your endless love, support and inspiration.

DEDICATION

To my parents, Daojun Lin and Jufang Song.

Chapter 1

INTRODUCTION

Natural language processing (NLP) seeks to extract, analyze, understand and predict information carried by natural language texts. Traditional NLP tasks include part-of-speech tagging, parsing, automatic document summarization, machine translation and so on.

As natural language conveys information at many different granularities, from simple word (or token-based) representations, to rich hierarchical syntactic representations, to high-level logical representations across document collections, outputs of most NLP tasks are fundamentally *discrete* mathematical structures. For instance, in part-of-speech tagging, the task is to assign parts of speech, such as noun, verb and adjective to each of the input word (and other tokens), and in this case, the discrete output space is all the possible tagging sequences for an input sequence (of words or other tokens). In parsing, a parser aims to find the best syntactic tree for a given sentence among all valid trees under some grammar rules. In extractive document summarization, the task is to find several sentences of a document that best represents the whole document, where all possible subsets of sentences that are within the summary length limit consist of the discrete space of interest. In word alignment for machine translation, the finite set of objects to explore is the set of all possible alignments between the source language sentence and target language sentence.

Due to the discrete and combinatorial nature of natural language, many NLP problems can be seen as problems of finding an optimal object from a finite set of objects, which can then be casted as *combinatorial optimization* problems.

Precisely, denote the finite set of objects of interest as $\mathcal{Y}$. An NLP problem can often be defined as finding $\boldsymbol{y}^* \in \mathcal{Y}$ such that

$$\boldsymbol{y}^* \in \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} s(\boldsymbol{x}, \boldsymbol{y})$$

where $\boldsymbol{x} \in \mathcal{X}$ is the input of the problem (e.g., $\boldsymbol{x}$ could be a document in document summarization, or $\boldsymbol{x}$ could be a sentence in part-of-speech tagging), and $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a score

function that measures how good $\boldsymbol{y}$ is given input $\boldsymbol{x}$.

Finding $\boldsymbol{y}^*$ in $\mathcal{Y}$ is often referred as the inference problem, or decoding problem. Since $\mathcal{Y}$ is finite, one way to find the optimal solution would be exhaustive search, which would only be possible when the size of the output space is limited. Example tasks of this sort include document categorization, where it is possible to enumerate all possible target categories, compute a score of the input document belongs to each category, and output the category with the highest score.

In most of other NLP problems, however, the size of $\mathcal{Y}$ typically grows in a high polynomial or even exponential rate as the size of the input grows, making an exhaustive search intractable or impractical. For example, in part-of-speech tagging, the size of $\mathcal{Y}$, $|\mathcal{Y}|$, would be $n^{|\boldsymbol{x}|}$ where n is the number of all possible tags, and $|\boldsymbol{x}|$ is the length (number of words) of the input sentence $\boldsymbol{x}$. In extractive document summarization, $|\mathcal{Y}|$ would be $\binom{|\boldsymbol{x}|}{k}$ where $|\boldsymbol{x}|$ is the number of sentences in a input document $\boldsymbol{x}$ and k is the largest allowed number of sentences in a summary with for example $|\boldsymbol{x}| \gg k > 10$ in a typical summarization task; not to mention in a more realistic case, k is unknown and the constraint is usually on the number of words (or bytes) in a summary, where characterization of the solution itself difficult.

Indeed, the combinatorial nature of the output space makes many NLP problems challenging to solve. There are two commonly used viewpoints to better understand the combinatorial natural of the output space $\mathcal{Y}$.

Vector representation of $\boldsymbol{y}$

NLP researchers usually view any element $\boldsymbol{y} \in \mathcal{Y}$ as a variable length vector over a finite alphabet set. In particular, let this finite set be $\mathcal{A}$; given an input $\boldsymbol{x}$, we can view the output as a vector: $\boldsymbol{y} = (y_1, \cdots, y_{L_{\boldsymbol{x}}})$ where $y_i \in \mathcal{A}, i = 1, \cdots, L_{\boldsymbol{x}}$, and the length of the output vector, $L_{\boldsymbol{x}}$, usually depends on the corresponding input. In part-of-speech tagging, for example, alphabet set $\mathcal{A}$ is the set of all possible tags, $L_{\boldsymbol{x}}$ is just the number of tokens contained in input sentence $\boldsymbol{x}$.

Decomposition into parts

In many structured prediction problems in NLP, the output can be decomposed into "parts", and any output $\boldsymbol{y} \in \mathcal{Y}$ can be seen as "assembling" of parts; in other words, let $n_{\boldsymbol{x}}$ be the number of parts available given input $\boldsymbol{x}$, we have $\mathcal{A} = \{0, 1\}$ and $\boldsymbol{y} \in \{0, 1\}^{n_{\boldsymbol{x}}}$ is just a binary vector where y_i indicates whether part i appears in the output. For instance, in word alignment, given an input that consists of two sentences $e_1, \cdots, e_I$ and $f_1, \cdots, f_J$, the output $\boldsymbol{y} = \{y_{i,j}\}_{i=1,\cdots,I, j=1,\cdots,J}$ is a binary vector with length $I \times J$ and $y_{i,j}$ indicates whether e_i is aligned to f_j. Take document summarization as another example, in which case the output would be binary vector with length equaling to the number of sentences in the input document and y_i being a binary variable that indicates whether sentence i appears in the summary.

Note that the size of $\mathcal{A}$ is usually much smaller than the size of $\mathcal{Y}$. Therefore if all the output variables y_i are independent of each other, inferring the best $\boldsymbol{y}$ can then be decomposed to $n_{\boldsymbol{x}}$ sub-problems over $\mathcal{A}$, which are relatively easy to solve. What makes many NLP problems challenging, on the other hand, is that such independence assumption is seldom true, and the complexity of $\mathcal{Y}$ is characterized by *strong* dependences among the output variables, often with constraints corresponding to a combinatorial structure involving matchings, permutations and trees.

Denote the feasible set of outputs with respect to $\boldsymbol{x}$ as $\mathcal{Y}_{\boldsymbol{x}}$, and we have

$$\mathcal{Y} = \bigcup_{\boldsymbol{x} \in \mathcal{X}} \mathcal{Y}_{\boldsymbol{x}},$$

and note that an output might not be able to contain all available parts due to constraints (e.g., parts in an output should be a tree in parsing, and a matching in word alignment) over the parts, we have

$$\mathcal{Y}_{\boldsymbol{x}} \subseteq \{0, 1\}^{n_{\boldsymbol{x}}}.$$

Denote these additional constraints as $c(\boldsymbol{x}, \boldsymbol{y}) \leq 0$ where $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, we have $\mathcal{Y}_{\boldsymbol{x}} = \{\boldsymbol{y} \in$

$\{0,1\}^{n_{\boldsymbol{x}}} | c(\boldsymbol{x}, \boldsymbol{y}) \leq 0\}$ and therefore, the problem of interest is then to find a y^* such that

$$\boldsymbol{y}^* \in \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}} s(\boldsymbol{x}, \boldsymbol{y}) \tag{1.1}$$

$$= \operatorname*{argmax}_{\boldsymbol{y} \in \{0,1\}^{n_{\boldsymbol{x}}}, c(\boldsymbol{x}, \boldsymbol{y}) \leq 0} s(\boldsymbol{x}, \boldsymbol{y}). \tag{1.2}$$

Problems of this sort with complex $\mathcal{Y}_{\boldsymbol{x}}$ are often referred as *structured classification* or *structured prediction* problems. This thesis will focus on structured prediction problem arises in NLP.

1.1 Inference Algorithms

In general, there are two ways to address the difficulty of intractable exact structured prediction: either restrict the models (forms of the score function) to those for which exact inference is feasible, or use approximate inference while allowing wider range of models. Both of these directions have been actively explored in the research of graphical models (and other related areas), where example of the first approach include conditional random field on trees (Lafferty *et al.* , 2001), and examples on the second direction include loopy belief propagation (Pearl, 1988), tree-reweighted message passing (Wainwright *et al.* , 2005) and linear programming relaxations (Roth & Yih, 2004). In this section, we briefly review some inference (decoding) algorithms used in NLP and motivate the proposed methods in the thesis.

1.1.1 Dynamic programming

For some of the structured prediction problems, dynamic programing sometimes offers a solution under the assumption of locality of the output variable dependence. Specifically, to apply dynamic programing to NLP decoding problems, one often assumes a specific structure of the score function s: that it can be computed based on "local" parts of the output space. In sequence labeling problem (e.g. part-of-speech tagging), for instance, the widely used Viterbi algorithm relies on certain factorizing properties of the score function

s:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{|\boldsymbol{x}|} f(\boldsymbol{x}, y_i, y_{i-1}),$$

where the score function is factorized to sums of functions that only dependents on adjacent output variables. With such a locality of interaction assumption, the combinatorial optimization problems can be broke down subproblems with into structures that are densely shared, and then can often be solved efficient by dynamic programming. From early work on Levenshtein distance (Levenshtein, 1966) for monotonic sequence alignment to more recent work on incremental parsing (Huang & Sagae, 2010), dynamic programming is one of the most widely used techniques in the decoding of NLP problems due to its efficiency. The efficiency, however, comes with the expense of expressive power of the score function. In many other combinatorial problems of interests, locality assumption is not adequate and dynamic programing is no longer applicable.

1.1.2 Integer linear programming

Recently, much attention has been devoted to *integer linear programming* (ILP) formulations of NLP problems, with interesting results in applications like semantic role labelling (Roth & Yih, 2005), dependency parsing (Martins *et al.* , 2009a), word alignment for machine translation (Lacoste-Julien *et al.* , 2006), summarization (McDonald, 2007; Clarke & Lapata, 2008) and joint information extraction (Martins & Smith, 2009). In general, solving arbitrary ILPs is an NP-hard problem. On the other hand, ILPs are a well studied optimization problem with branch and bound algorithms that sometimes finds the optimal solution in affordable time, making ILP applicable to some small or median scale NLP problems.

In the ILP approach, both the score function and constraints are linear in a set of integer variables, and the NLP problems are then formularized as a constrained optimization problem with linear objective function. Integer solutions to these optimization problems are usually obtained using off-the-shelf ILP solvers. Typically, the integer variables ranges from 0 to $|\mathcal{A}|-1$ indicating the value of an output variable takes. Precisely, given an output vector $\boldsymbol{y} = \{y_1, \cdots, y_{L_{\boldsymbol{x}}}\}$ and its alphabet $\mathcal{A} = \{a_1, \cdots, a_{|\mathcal{A}|}\}$, an ILP approach typically introduces an integer variable $z_i \in \mathbb{Z} \cap [0, |\mathcal{A}|]$ such that $z_i = j$ if $y_i = a_j$. The characterization of

the output space $\mathcal{Y}_{\boldsymbol{x}}$ (i.e. dependencies among output variables y_i) is then expressed via linear constraints on these integer variables z_i. In the document summarization example, $\mathcal{A} = \{0, 1\}$ and therefore z_i is a binary variable indicating whether sentence i is included in the summary. The summary length constraint can then be expressed as a single inequality: $\sum_{i=1}^{|\boldsymbol{x}|} c_i z_i \leq b$ where c_i is the cost for sentence i and b is the budget.

The efficiency of the ILP approach greatly dependents on the number of integer variables as well as the number of linear constraints required to fully characterize $\mathcal{Y}_{\boldsymbol{x}}$. Moreover, the score function is required to be linear, limiting its expressiveness. To model interactions among output variables, additional integer variables along with possible extra constraints are required. For example, in the ILP formulation of document summarization of (McDonald, 2007), to incorporate the similarity of two sentences such that redundancy can be removed in the score function, additional integer variables $z_{i,j}$ are introduced to indicate that *both* sentence i and j appears in the summary. Consequently, additional constraints on z_i, z_j and $z_{i,j}$ are required to preserve sanity. Therefore, the number of integer variables as well as the number of constraints grows quadratically as the number of sentences in the input document grows, making the ILP approach impractical for real-world document summarization task (e.g. multi-document summarization, where typically thousands of sentences are to be summarized. See Section 4.4.1 for more detail). When one want to model three-way interactions in ILP, moreover, numbers of variables and constraints typically grow cubically. In general, the number of variables and constraints grow exponentially in the degree of interaction, which limits ILP's applicability to problems where modeling of higher-order interactions is desired.

1.1.3 Relaxation

Both dynamic programming and ILP offer exact (optimal) solutions for certain types of decoding problems. For many other combinatorial problems of interest in which exact solutions are extremely expensive to obtain, people usually resort to approximation algorithms that approximately solve the decoding problem. One family of approximation algorithms is those algorithms that are based on relaxations. In the relaxation-based approach, the

original problem is relaxed to another problem where the target output space $\mathcal{Y}'$ contains every element in $\mathcal{Y}$ as well as some other elements that are not in $\mathcal{Y}$. The central idea is that finding the optimal output (with respect to the score function) in $\mathcal{Y}'$ is much easier than finding the optimal output in $\mathcal{Y}$. For example, a problem on $\mathcal{Y}$ might be NP-hard while the same problem on $\mathcal{Y}'$ is polynomial-time solvable, or a problem might be solvable in polynomial time on both $\mathcal{Y}$ and $\mathcal{Y}'$, but when on $\mathcal{Y}'$, algorithm with a significantly lower complexity is available. The solution found in $\mathcal{Y}'$ is then projected onto $\mathcal{Y}$ to get a solution $\bar{y}$ for the original decoding problem. Usually, the solution $\bar{y} \in \mathcal{Y}$ is not optimal; it is an approximate solution sometimes even without formal approximation guarantees. Linear programing relaxation is one commonly used relaxation methods, where the integral constraints in the original ILP problems are dropped. While ILP is NP-hard in general, the linear programing relaxed problem is relatively easy since linear programing can be solved in polynomial time (Khachiyan, 1979; Karmarkar, 1984). Linear programing relaxations have recently been applied to maximum a posterior inference in Markov random fields with some theoretical guarantees (Wainwright *et al.* , 2005; Ravikumar *et al.* , 2008) with some success in real-world applications such as computer vision (Szeliski *et al.* , 2006).

Recently, another family of relaxation methods, namely dual decomposition, has been introduced into NLP (Rush *et al.* , 2010; Koo *et al.* , 2010). The algorithms in dual decomposition is a particular instance of Lagrangian relaxation, where the latter is one of the classical methods for combinatorial optimization (Fisher, 1981). Similar to all other relaxation methods, the first key step in Lagrangian relaxation is to choose a finite set $\mathcal{Y}'$ such that $\mathcal{Y}' \supset \mathcal{Y}$. An assumption is then made such that

$$\mathcal{Y} = \{\boldsymbol{y} : \boldsymbol{y} \in \mathcal{Y}', A\boldsymbol{y} = \boldsymbol{b}\}$$

where $A \in \mathbb{R}^{p \times |\boldsymbol{y}|}$ and $b \in \mathbb{R}^p$. The implication is that linear constraints $A\boldsymbol{y} = \boldsymbol{b}$ are required in order to fully characterize $\mathcal{Y}$, but these constraints complicates the decoding problem. So instead of incorporating them as hard constraints, Lagrangian multipliers $\boldsymbol{\lambda}$ are introduced, and the dual objective

$$\max_{\boldsymbol{y} \in \mathcal{Y}'} s(\boldsymbol{x}, \boldsymbol{y}) + \boldsymbol{\lambda}^\top (A\boldsymbol{y} - \boldsymbol{b})$$

is considered. The key point is that the above objective is easy to evaluate (under the assumption that $s(\boldsymbol{x}, \boldsymbol{y})$ is linear over the parts $\boldsymbol{y}$), and subgradient algorithms can then be used to minimize the dual objective. Again, the linear assumption limits the expressive power of the score function.

1.1.4 Other approaches

Other approximate decoding approaches include variants of dynamic programming (e.g. dynamic programming with beam search), re-ranking (Collins, 2000) where the central idea is re-ranking a small set of results generated by a simpler model (e.g., model with locality assumption), and coarse-to-fine approaches (Charniak & Johnson, 2005; Petrov *et al.* , 2006) which basically generalizes the re-ranking idea. Most of these approximate decoding approaches, however, lack formal theoretical guarantees on their performances.

1.2 Proposed Approach

To make decoding in a complex output space affordable, all the aforementioned approaches in Section 1.1 pose assumptions either on the score function, or on the dependencies of output variables, or on both. Most of them use a *linear* score function, where the score of $\boldsymbol{y} \in \mathcal{Y}$ is a sum of scores of the parts it contains. However, simply summing scores on individual parts does not usually model the problem accurately because for a structured problem, parts usually *interact* and cannot be scored independently. For instance, when a sentence is included in a summary, other sentences that are similar to this sentence should be scored lower to reduce redundancy.

In this thesis, instead of viewing the output $\boldsymbol{y}$ as a binary vector, an output is regarded as a *subset* of all possible parts in the output. In particular, denote $V_{\boldsymbol{x}} = \{1, 2, \cdots, |n_{\boldsymbol{x}}|\}$ as the *ground set*, which is the collection of all available parts given input $\boldsymbol{x} \in \mathcal{X}$ (recall that $n_{\boldsymbol{x}}$ is the number of parts in a given $\boldsymbol{x}$). Since $\boldsymbol{y}$ is a binary vector with element y_i indicating whether a part i appears in the output, we can equivalently view the output as a subset of parts. In particular, denote an output subset as Y, we have $Y \subseteq V_{\boldsymbol{x}}$ and $Y = \{i : y_i = 1, i = 1, \cdots, |n_{\boldsymbol{x}}|\}$. Therefore, the decoding problem can be written as finding the best subset such that a score function is maximized, i.e.

Problem 1. *find*

$$Y^* = \underset{Y \subseteq V_{\boldsymbol{x}}, c(\boldsymbol{x}, Y) \leq 0}{\operatorname{argmax}} \; s(\boldsymbol{x}, Y), \tag{1.3}$$

and now for a given $\boldsymbol{x}$, the score function can be seen as a *set function* on ground set $V_{\boldsymbol{x}}$, i.e. $s(\boldsymbol{x}, \cdot) : 2^{V_{\boldsymbol{x}}} \rightarrow \mathbb{R}$. We also have another set function $c(\boldsymbol{x}, \cdot) : 2^{V_{\boldsymbol{x}}} \rightarrow \mathbb{R}$ that expresses any additional constraints posed on the parts (ground set). The key point here is by using a general set functions, which essentially generalize linear functions, we increase the expressive power of the score function, and therefore interactions among parts could be properly modeled.

Note that to model the same degree of interactions using vector representation of outputs, the score function in Eqn 1.1 would need $O(2^{n_{\boldsymbol{x}}})$ non-linear terms. Similarly, to model the same degree of interactions while still having a linear score function (as required in the ILP approach), one would need to expand the dimension of the representative vector to $2^{n_{\boldsymbol{x}}}$ (i.e., to use a pseudo Boolean function).

Of course, with great expressive power often comes with great computation complexity. In general, using a set function as an objective makes it nearly hopeless to solve the already complex large-scale decoding problems. On the other hand, there is a family of set functions that is nice in the sense that it makes optimization in Eqn. 1.3 either tractable or approximable where otherwise neither would be possible. This family of set functions is called *submodular functions*.

This thesis proposes to use submodular functions as score functions for NLP problems, by which not only the modeling power is increased, but also the corresponding decoding problem can be solved near-optimally with strong theoretical guarantee. Of course, one could always force submodularity upon an application, leading to an artificial and poorly performing score function even if it can be optimized well. As we will see in this thesis, we are fortunate in that submodularity naturally arises in various NLP problems, making submodular function a natural choice for scoring structured outputs.

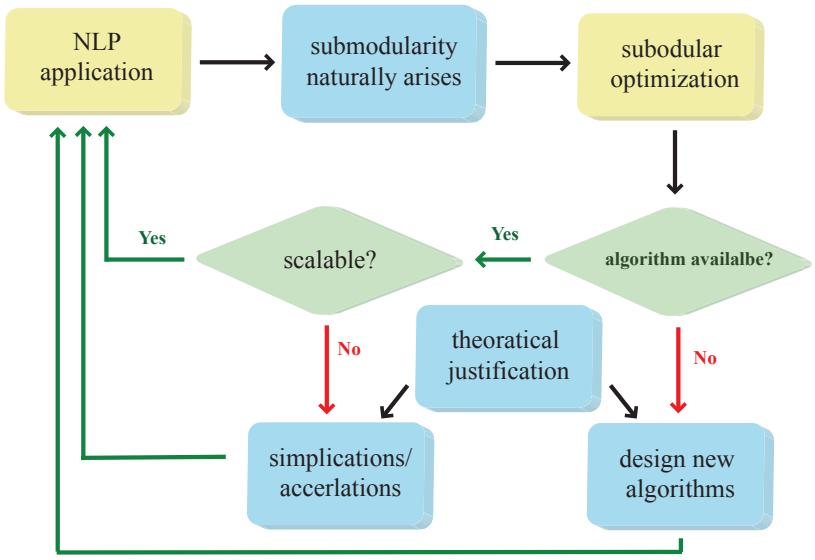


Figure 1.1: Methodology

1.3 Methodology and Contributions

This thesis aims to bridge the theory of submodular function optimization to the practise of natural language processing. In general, the following methodology is used. We always start with an NLP application, which is formulated as a combinatorial optimization problem (e.g, Problem 1, Problem 5, Problem 9, Problem 14, and Problem 15). We further show that how submodularity naturally arises and how submodular function lends itself in as a score function (e.g., see Section 4.6 and Section 6.3). The problem of interest is then a combinatorial optimization problem involving submodular functions. If the corresponding submodular optimization problem has not been studied before, we then analysis its hardness and design algorithms for it (e.g., the problem of optimization over the submodular knapsack in Section 3.5). If this problem has been studied before, we study whether the available algorithms are practical for problems of NLP scale. Often, it is the case that although polynomial time algorithms, either exact or approximate with rigours guarantees, have been shown by prior theoretical research on submodular optimization, they cannot be applied to the NLP problems due to their high polynomial complexities. For example, sub-

modular minimization is polynomial time solvable but the fastest algorithm still requires about $O(n^5)$ function evaluations (see Section 2.4.1 for more details), which is obviously not an option for problem with n at the scale of millions (e.g., applications in Chapter 7); for monotone submodular maximization with budget constraint, algorithms with rigours performance guarantees are available (see Section 2.4.2), but again, fail to be applicable to applications where efficiency is crucial (e.g., applications in Chapter 4). Indeed, there are gaps between theory and application when applying submodular optimization to NLP problems. In this thesis, we either propose acceleration methods (Section 3.3) or directly modify existing algorithms to make them practical. In both cases, we perform theoretical analysis accordingly, showing the acceleration does not violate optimality of the original algorithm, or proving performance guarantees still exist for the modified algorithms (see for example, Section 3.4). With theory and algorithms in place, we get back to the application, design rich class of submodular functions that suit the application (Section 4.7 and Section 5.6.1), evaluate our new approaches on benchmark tasks, and show by bridging the gap between submodular optimization and NLP applications, we can achieve performance beating the state-of-the-arts.

With this methodology in mind, there are three main threads in the development of this thesis, initiated by three different NLP applications.

- Summarization (Chapter 4 and Chapter 5).

  Document summarization is one of the classical NLP problems, and starts to receive increasing attention since the rise of "big data" problem in nowadays.

  The task of summarization is essentially a subset selection problem: given some budget, select a compact subset that best represents the ground set. One contribution of this thesis is that we show submodularity naturally arises in document summarization (Section 4.6). In particular, we show that many well-established methods for document summarization, including the widely used maximum marginal relevance (Carbonell & Goldstein, 1998) method, are actually corresponding to submodular function optimization (Theorem 20), without being realized for decades before this thesis. Moreover, we prove that the widely-used ROUGE score (Lin, 2004) for automatic summarization

evaluation, and the score used in recently popular Pyramid evaluation (Nenkova & Passonneau, 2004), are both monotone submodular (Theorem 21 and Theorem 22), giving further evidence that submodular functions are natural fit for the score function of document summarization.

It is therefore natural to model summarization problem as budgeted submodular maximization with budget constraint. When the submodular function is monotone, it is known that a greedy algorithm with partial enumeration (Sviridenko, 2004) can achieve $(1 - \frac{1}{e})$ approximation factor (see Section 2.4.2 for details). This algorithm, however, is not directly applicable to summarization due to the complexity in partial enumeration. We thus present a modified greedy algorithm (Algorithm 4) for submodular maximization with knapsack constraints that is cheap enough to be applicable to large scale summarization problems. Moreover, we generalize Khuller *et al.* (1999)'s theoretical result on optimizing set cover function under knapsack constraint to budgeted optimization problem with any monotone submodular objectives, and show that our modified algorithm is a $(1 - \frac{1}{\sqrt{e}})$-approximation algorithm for budgeted maximization of a monotone submodular function (Theorem 14). A scaling parameter is also introduced to account for the uncalibrated issue in the ratio-based greedy heuristic (ratio of objective function gain over the cost), and we also show that with non-unity scaling parameter, performance of the proposed algorithm is still bounded (Lemma 4). We moreover show that the modified greedy algorithm always finds solutions that achieve over 98% of the optimal value, performing near-optimally in practice for summarization tasks.

The remain issue is then how to design monotone submodular functions as score functions for summarization. While theoretical analysis favors monotone submodular function, most of the well-established greedy methods for summarization, although related to submodular optimization, optimizes a *non-monotone* submodular function. We then contribute in this thesis by introducing a class of submodular functions that is not only monotone but also models relevance and diversity simultaneously for document summarization (Section 4.7). This class of submodular functions is further

generalized to a mixture of submodular components, where each component might either models the relevance or models the diversity, and might differs either in function forms or in function parameters (Section 5.6.1). As mixture models have been shown to provide effective generalizations of classical methods in many simpler probabilistic density estimation settings (McLachlan & Basford, 1988), we show powerful score function can be established for summarization by using mixture of simple submodular functions.

We further investigate how to learn the submodular mixtures in a supervised learning setting. In such learning, we inevitably need to use approximate learning since by using a wider and more expressive class of score functions, the learning problem is intractable. On the hand, we of course want to leverage the rigorous approximation guarantees of submodular optimization such that the approximate learning of submodular mixtures can be bounded in performance in some way. On the other hand, as shown in (Kulesza *et al.* , 2007), if the learning and inference are not compatible, learning could fail even with approximate inference with rigorous guarantees. We therefore carefully choose the learning algorithm (Algorithm 11) such that it is compatible to our approximate inference (Section 5.4). We moreover theoretically analysis the performance of submodular mixtures with approximate learning (Section 5.5), showing the risk of approximate learning can be bounded in terms the empirical risk of exact learning (Theorem 23).

Empirically, we evaluated our approaches in various summarization tasks, not limited to document summarization but also including speech summarization (Section 4.2) as well training data summarization (Section 4.3), and show that our novel approach outperforms many well-known methods, e.g., MMR (Carbonell & Goldstein, 1998), LexRank (Erkan & Radev, 2004) and ILP approaches (McDonald, 2007; Gillick *et al.* , 2009; Xie *et al.* , 2009), for summarization. In particular, when evaluated on the standard benchmark task for document summarization, namely Document Understanding Conference (DUC), we achieve **best results ever reported** on DUC-2004, DUC-2005, DUC-2006, and DUC-2007.

- Optimal corpus creation (Chapter 7).

  A challenge working with large amount of learning data in NLP is determining how to quickly test a new algorithm on such data. One way to address this problem is to produce a smaller version of the data. Often, the complexity of an iteration of a learning system is linear in the number of samples but *polynomial* in the number of classes. For instance, in language models, decoding using a trigram language model with size-N vocabulary (number of *classes* or *types*) has, in the worst case, a complexity of at least $O(N^3)$.

  The goal of this application therefore is to improve the turnaround time for developing and testing novel algorithms but still exploit the utility in large data. One approach is to select a subset of the data where the number of types is limited and the total number of samples is maximized. Existing method (King *et al.* , 2005) for this purpose is basically a greedy approach where a type is added if the amount of data (number of samples) containing nothing other than the already selected types and this new type is maximized. The greedy algorithm, although conceptually simple, could perform arbitrarily poorly for the corpus creation problem since it essentially maximizes a supermodular function (see Section 7.2).

  To address this issue, we propose to view the corpus creation problem as a combinatorial problem, where we want to maximizes a modular function with submodular knapsack constraint (Problem 5). Intuitively, the number of types contained in a subset of data is a submodular function, and therefore when limiting the number of types, we have a submodular knapsack constraint.

  This problem has not been studied before as far as we know. We therefore contribute to perform a pilot study for this problem. We show the hardness result, with a theorem (Theorem 16) saying that there is no (ρ, σ)-bicriteria approximation algorithm for submodular knapsack problem, even with monotone submodular function, for any ρ and σ with $\frac{\rho}{\sigma} = o\left(\sqrt{\frac{\ln n}{n}}\right)$ where n is the ground set size. We further show that the submodular knapsack problem can be reduced to submodular minimization with

cardinality lower-bound. An approximation algorithm (Algorithm 5) is then immediate based on the bi-criteria algorithm for submodular minimization with cardinality lower-bound (Svitkina & Fleischer, 2008).

Since submodular knapsack problem is hard, we alternatively treat the corpus creation problem as finding a subset of data that simultaneously minimizes the vocabulary size (number of types) and maximizes the total amount of data (Problem 15). Quantities such as number of classes (or types) in a set of samples, or quality of a bundle of classes are submodular functions, making finding the optimal solutions possible. We then apply the principal partition to our problem such that solutions for all possible trade-offs between a modular function and a submodular function can be found efficiently (Section 7.5).

When submodular functions are graph-representable, we can convert our submodular minimization problem to the problem of finding minimum s-t cuts in a graph, and therefore a fast parametric flow algorithm (Gallo *et al.* , 1989) can be used to find all the solutions for the corpus creation problem for all possible values of the trade-off coefficient, while only requiring the computational complexity of running a single minimum s-t cut, which can be solved very efficiently even on very large graphs. Using this approach, we show results for speech recognition on the Switchboard-I speech recognition corpus, demonstrating improved results over previous techniques for this purpose. We also demonstrate the variety of the resulting corpora that may be produced using our method.

Ideally, we would like a process that can take a large corpus and produce a subset that satisfies a particular purpose. For example, when vocabulary size is the key attribute hindering the rapid evaluation of novel method, we might choose a limited vocabulary subset of data of maximal size. On the other hand, we may wish to correct for some other quality, such as imposing a bias against certain word forms. In some cases, the function used for expressing such bias can not be represented by a graph (e.g, Eqn 7.5). Therefore, the submodular function we want to minimize is no longer graph-representable, and the effient (parametric) s-t cut algorithm no longer applies,

which then draws our attension to general submodular function mininization.

Although combinatorial polynomial algorithms are available for general submodular function minimization, they are useless for this application due to their high polynomial approxima time complexity. On the other hand, minimum-norm-point (MN) algorithm (Fujishige & Isotani, 2011) is the widely accepted as the most efficient algorithm for general submodular function minimization (see Section 3.1 for more details). In this thesis, we offer the first empirical study of the complexity of MN algorithm when applied to a real-world application scale problem of practical interest. We show that the empirical complexity of MN algorithm on a particular type of submodular functions that arises in practice is not as good as the complexity of the combinatorial algorithms for submodular functions minimization (Section 3.2). To make MN algorithm applicable to our application, we moreover propose acceleration methods, ensure that it does not violate the optimality (Theorem 12), and show that it speeds up MN algorithm phenomenally in practise (Section 3.3).

- Word alignment for machine translation (Chapter 6).

  Word alignment is a key component in most statistical machine translation systems. In this thesis, we provide a new angle of viewing word alignment problem by casting it as an optimization problem over matroid constraints. We further show that submodularity naturally arises when modeling word fertility (Section 6.3). Therefore, the word alignment problem can then be seen as submodular maximization problem over matroid constraints (Problem 14). This new framework extends previous matching-based frameworks in two respects: submodular objective functions generalize modular (linear) objective functions, and matroid constraints generalize matching constraints. Moreover, such generalizations do not incur a prohibitive computational price since submodular maximization over matroids can be efficiently solved with performance guarantees (Algorithm 14). Experiments on the English-French Hansards alignment task show that our approach achieves lower alignment error rates compared to conventional matching based approaches as well as an ILP based approach.

A list of contributions of this thesis is summarized in Section 8.1.

1.4 Organizations

The organization of this thesis somehow does not follow the flow of aforementioned development. In particular, most of the theoretical and algorithmic contributions of this thesis are extracted and introduced in a single chapter (Chapter 3) such that it is self-contained while other chapters mostly focus on the applications. We also use a separate chapter (Chapter 5) for learning of submodular mixtures since it is applicable to general structured problems.

The rest of this thesis is organized as follows. In Chapter 2, we briefly review the background on matroid theory and submodular functions, providing basic material to understand the rest of this thesis. Chapter 3 presents theory and algorithm contributions of this thesis on budgeted submodular function maximization, submodular function minimization and submodular knapsack problems. Chapter 4 shows our submodular paradigm for document summarization, with evidences of the fitness of submodular functions for document summarization, descriptions of submodular functions that model the quality of summary, and empirical results on speech summarization, training-data summarization, query-independent document summarization, as well as query-focused document summarization. Chapter 5 reviews learning algorithms for structured decoding problems, introduces the proposed approach for learning a mixture of submodular functions, and analysis the learning algorithm's performance when approximate inference is used. Applications of submodular functions to word alignment in machine translation and corpus subset selection are presented in Chapter 6 and Chapter 7 respectively. Conclusions and future work are given in the last chapter.

1.5 Collaborations and Publications

Some of the work described in this thesis has been published in conference proceedings.

In Chapter 3, the theoretical analysis of the modified greedy algorithm is published in (Lin & Bilmes, 2010b) coauthored with Jeff Bilmes, and some of the empirical complexity results of minimum-norm-point algorithm is published in (Jegelka *et al.* , 2011), coauthored with Stefanie Jegelka and Jeff Bilmes.

The submodular summarization (Chapter 4) idea was first proposed in (Lin & Bilmes,

2009), coauthored with Jeff Bilmes, in the context of training data subset selection. The idea was later applied to speech summarization in (Lin *et al.* , 2009) coauthored with Jeff Bilmes and Shasha Xie, and was further studied in (Lin & Bilmes, 2010b, 2011a) in the application of multi-document summarization, both coauthored with Jeff Bilmes.

Submodular maximization over matroids with application to word alignment (Chapter 6) is published in (Lin & Bilmes, 2011c), optimal selection of limited vocabulary speech corpora (Chapter 7) is published in (Lin & Bilmes, 2010a, 2011b), all of which are coauthored with Jeff Bilmes.

Chapter 2

THEORY

Submodularity plays an important role in many areas, such as economics (Vives, 2001), game theory (Topkis, 1998; Shapley, 1971), combinatorial optimization (Edmonds, 1970; Lovász, 1983; Schrijver, 2003), and operations research (Conn & Cornuejols, 1990). More recently, submodular functions have started receiving attention in the machine learning and computer vision community (Kempe *et al.* , 2003; Narasimhan & Bilmes, 2005a; Krause & Guestrin, 2005b; Narasimhan & Bilmes, 2007; Krause *et al.* , 2008; Kolmogorov & Zabin, 2004). Submodularity arises naturally in applications in natural language processing, but its presence has not been that widely recognized. This thesis aims to bring more attention to exploring submodularity in NLP, by showing how submodularity naturally and frequently arises in NLP and how we can make submodular function optimizations practical and utilize them to better address real applications in NLP (e.g., the tasks of document summarization (Lin & Bilmes, 2010b) and word alignment (Lin & Bilmes, 2011c)).

In this chapter, we provide a background on submodularity that is necessary to understand the rest of this thesis. We start with definitions of submodular functions along with some examples that are potentially useful for NLP tasks. Concepts of matroid, as well its connections to submodularity, are then introduced. We further show several continuous extensions of submodular function that are useful for submodular analysis, and conclude this chapter with a literature survey of submodular function optimization algorithms.

2.1 Submodular Functions: Definition and Examples

Functions of interest here are set functions. A set function maps subsets of a ground set into real values. Set functions naturally arise when modeling natural language processing problem since natural language is essentially discrete in a mathematical sense. One simple example of a set function can be a function that counts the number of distinct words

(vocabulary) in a set of documents, where the ground set is the collection of all documents, and the set function maps all the subsets of documents to their vocabulary sizes. There could be many properties of a set function. For instance, we say a set function $f : 2^V \to \mathbb{R}$ is

- **non-negative** if $f(S) \geq 0$ for all $S \subseteq V$.
- **normalized** if $f(\emptyset) = 0$.
- **monotone** if $f(S) \leq f(T)$ whenever $S \subseteq T$.
- **symmetric** if $f(S) = f(V \setminus S)$ for all $S \subseteq V$.

Amongst all the possible forms of set functions, there is one family of set functions that is nice in a way that is not only mathematically beautiful, but also that it is practically useful and arises naturally in many real world applications. This family of set functions is called *submodular functions*.

Definition 1 (Submodular functions)**.** *Consider a set function $f : 2^V \to \mathbb{R}$, which maps subsets $S \subseteq V$ of a finite set V to real numbers. $f(\cdot)$ is called* submodular *if for any $S, T \subseteq V$,*

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T) \tag{2.1}$$

Submodular function can also be defined in terms of a property on increment values:

Definition 2 (Submodular functions and diminising return property)**.** *For any $R \subseteq S \subseteq V$ and $s \in V$, $f(\cdot)$ is submodular if*

$$f(S \cup \{s\}) - f(S) \leq f(R \cup \{s\}) - f(R) \tag{2.2}$$

Definition 2 states a property of *diminishing returns*, which means that the return (the increment in function value) diminishes when adding a new element in a larger context. In other words, a set function is submodular if adding a new element s to a set R always increases the function value at least as much as the increment if we add s to the superset

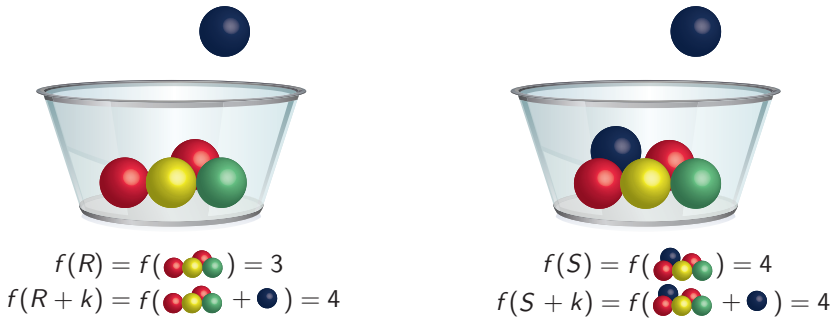


Figure 2.1: Submodular function and diminishing returns

S. Intuitively, this means that adding a new element helps less if we already have many elements, and more if we have fewer elements.

The law of diminishing return is a well-known fundamental principle of economics, where a common sort of example is adding more workers to a job, such as assembling cars, and the inputs are used less and less effectively as adding more and more workers. A more abstract example is illustrated in Figure 2.1, where we consider counting the number of colors in an urn. On the left of Figure 2.1, there are 4 balls in the urn with 3 colors, while on the right, there are 5 balls with 4 colors. Now consider adding a new ball. On the left, the increment of color number is 1, while on the right the increment is 0. In other words, the return, or the marginal benefit, diminishes in a larger context (the one on the right), and therefore the function that counts the number of ball colors is submodular. Actually, this balls-in-the-urn example is analogues to words-in-the-document example, where balls can be seen as words and urn can be seen as document, and thus the vocabulary size function is submodular as well.

Submodularity arises naturally in a variety of discrete domains including graph cuts (Goemans & Williamson, 1995), set covering (Feige, 1998a), facility location (Cornuejols *et al.* , 1977) and game theory (Shapley, 1971) problems. In the following, we discuss some examples of submodular functions that are potentially useful for NLP tasks. Before doing so, we define supermodular functions and modular functions.

Definition 3 (Supermodular functions)**.** *A set function $f : 2^V \to \mathbb{R}$ is called supermodular if $-f$ is submodular.*

Definition 4 (Modular functions)**.** *A set function $f : 2^V \to \mathbb{R}$ is called modular if it is both submodular and supermodular.*

2.1.1 Submodular functions in information theory

Information theory can be viewed as a way to measure and reason about the complexity of messages. It has been widely applied to natural language processing. Submodularity arises naturally in information theory.

First of all, Shannon entropy (Cover & Thomas, 2006) can be seen as a submodular function (Kelmans *et al.* , 1983). Precisely, let $X_1, ..., X_n$ be random variables, and $V = \{X_1, ..., X_n\}$. Then the entropy $f(S) = H(\{X_v\}_{v \in S})$ is a submodular function.

Proof. $\forall S, T \subseteq V$ and $X_k \notin T$,

$$H(S \cup X_k) - H(S) = H(X_k|S) \geq H(X_k|T) = H(T \cup X_k) - H(T),$$

where the inequality follows from the fact that conditioning never increases the entropy (Cover & Thomas, 2006). Therefore $H(\cdot)$ is submodular by Definition 2. □

The submodularity of a multivariate normal distribution with covariance matrix Σ (positive definite symmetric matrix) gives us that $f(S) = \log\det\Sigma_S$ for nonempty set S is a submodular function, where Σ_S denotes the principal submatrix of Σ indexed by S, and $S \subseteq V$ where V is the index set of the row/column of Σ.

Mutual information is not submodular in general. Fortunately, under some weak conditional independence assumptions, mutual information can be proved to be submodular (Krause & Guestrin, 2005b). Precisely, let $A, B \subseteq V$, $A \cap B = \emptyset$ and $\forall X_1, X_2 \in A, X_1 \perp\!\!\!\perp X_2|B$. Then for $S \subseteq A \cup B$, $f(S) = I(B; S)$ is submodular.

Proof.

$$\begin{aligned} f(S) &= H(B) - H(B|S) = H(B) + H(S) - H(B \cup S) \\ &= H(B) + H(S) - H\left(B \cup (A \cap S)\right) \\ &= H(B) + H(S) - H(B) - H(A \cap S|B) \\ &= H(S) - \sum_{X \in A \cap S} H(X|B) \end{aligned}$$

where the second and fourth equalities are based on the chain rule and the last equality follows from the conditional independence assumption. The first term $H(S)$ is submodular. The second term is modular, and therefore $f(S)$ is submodular. □

Also, symmetric mutual information is submodular, i.e. $f(S) = I(\{X_v\}_{v \in S}; \{X_v\}_{v \in V \setminus S})$ is submodular.

2.1.2 Submodular functions on bipartite graphs

Bipartite graphs are those graphs whose vertices can be divided into two disjoint sets, say V and F, such that every edge connects a $v \in V$ to a f in F. See Figure 2.2 for an example.

Bipartite graphs are useful in practise since they can represent and model interactions between two groups in many real world problems. Well-known applications include matching problems where, for instance, V could a set of papers and F could be a set of paper reviewers. In NLP applications, V could be a set of sentences, F could be a set of words, and a $v \in V$ connects to a $f \in F$ if the sentence represented by v contains the word represented by f.

We denote a bipartite graph $G = (V, F, E)$ where V are the left vertices, F are the right vertices, and $E \subseteq V \times F$ are the edges where each $(v, f) = e \in E$ is an edge connecting a node $v \in V$ and $f \in F$. Define a mapping $\gamma : 2^V \to 2^F$ such that $\gamma(X)$ is the set of nodes $f \in F$ that have a neighbor in X. That is

$$\gamma(X) \triangleq \{f \in F : \exists v \in X \text{ s.t. } (v, f) \in E\}. \tag{2.3}$$

An immediate submodular function can then be the cardinality of $\gamma(X)$, i.e.,

$$f(X) = |\gamma(X)|$$

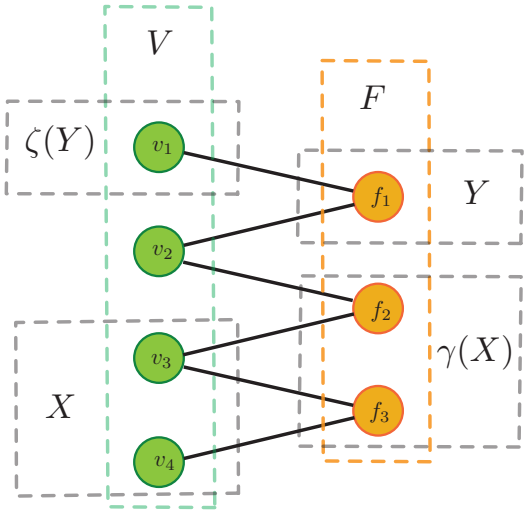


Figure 2.2: Submodular functions on bipartite graphs.

is submodular.

Actually, $|\gamma(X)|$ is the function that abstracts the aforementioned balls-in-the-urn and vocabulary functions. In particular, let the left vertex set V represent the set of sentences, the right vertex set F represent the set of words, and a left vertex v is connected to a right vertex f if sentence v contains word f. Then $|\gamma(X)|$ basically counts number of vertices in F that connects to X, and therefore is exactly the number of distinct words contained in X.

This can be generalized to any non-negative modular function over γ. I.e., given a non-negative modular function $m : 2^F \to \mathbb{R}_+$,

$$f(X) = m(\gamma(X))$$

is also submodular. This function could be seen as the weighted version of the vocabulary size function, which is practically useful as it could, for instance, measure the "importance" of the words contained in a collection of sentences. Note that when posing a non-positive modular function over γ, we get a supermodular function, and when m is an arbitrary modular function, $m(\gamma(X))$ is neither submodular nor supermodular in general.

Submodular functions can also be defined using vertices on the right as the ground set.

Define a mapping $\zeta : 2^F \rightarrow 2^V$:

$$\zeta(Y) \triangleq \{v \in V : \exists f \in Y \text{ s.t. } (v, f) \in E \text{ and } (v, f') \notin E, \forall f' \in F \setminus Y\}. \tag{2.4}$$

In other words, $\zeta(Y)$ is the set of nodes $v \in V$ that have neighbors *only* in Y and not in $F \setminus Y$. Then for $g : 2^F \rightarrow \mathbb{R}$ such that

$$g(Y) = m(\zeta(Y))$$

where $m : 2^V \rightarrow \mathbb{R}^+$ is a modular function, g is supermodular (see Appendix for a proof). $g(Y)$ could also be an useful function in practise where for instance, it can represent the number of sentences that *only* contain a certain set of words, that is, the number of sentences that do not have any out-of-vocabulary word (Lin *et al.* , 2007).

2.2 Matroids and Polymatroids

Matroid was first introduced by Whitney (1935), and it is a fundamental concept in combinatorial optimization. As we use matroid in the task of word alignment (Chapter 6), and also since matroid is closely related to submodular functions, we briefly introduce its background here.

Matroids are combinatorial structures that generalize the notion of linear independence in matrices. A pair $(V, \mathcal{I})$ is called a *matroid* if V is a finite ground set and $\mathcal{I}$ is a nonempty collection of subsets of V that are *independent*. In particular, $\mathcal{I}$ must satisfy three conditions (I), (II) and (III):

(I) $\emptyset \in \mathcal{I}$,

(II) if $X \subset Y$ and $Y \in \mathcal{I}$ then $X \in \mathcal{I}$,

(III) if $X, Y \in \mathcal{I}$ and $|X| < |Y|$ then $X \cup \{e\} \in \mathcal{I}$ for some $e \in Y \setminus X$.

We typically refer to a matroid by listing its ground set and its family of independent sets: $\mathcal{M} = (V, \mathcal{I})$. Each $X \in \mathcal{I}$ is called an *independent set* of matroid $(V, \mathcal{I})$, and I is called the *family of independent sets* of matroid $(V, \mathcal{I})$.

In words, condition (III) above says that if X is independent and there exists a larger independent set Y then X can be extended to a larger independent by adding an element of $Y \setminus X$, which implies that every maximal (inclusion-wise) independent set is maximum. In other words, all maximal independent sets have the same cardinality. A maximal independent set is called a *base* of the matroid.

Matroids arise frequently in many areas. One trivial example of matroid is a uniform matroid in which

$$\mathcal{I} = \{X \subseteq V : |X| \leq k\},$$

for a given $k \in \mathbb{N}$. It is easy to verify that the above $\mathcal{I}$ satisfies aforementioned three conditions for independence, and therefore $\mathcal{I}$ is a collection of independent subsets and $(V, \mathcal{I})$ is a matroid. The base, i.e. a maximal independent set, of this matroid is then any set with cardinality k (or the ground set V is $k > |V|$). This matroid is known as *uniform matroid*.

Another example is a partition matroid where the ground set V is partitioned into disjoint sets $V_1, \cdots, V_l$, i.e. $\bigcup_{i=1}^{l} V_k = V$ and $V_i \cap V_j = \emptyset$ for all $i \neq j$. And we have independent sets:

$$\mathcal{I} = \{X \subseteq V, X \cap V_i \leq k_i, \forall i\},$$

where $k_i \in \mathbb{N}, i = 1, \cdots, l$. The first two conditions are trivial to verify. To check the third condition of independent sets, consider $X, Y \in \mathcal{I}$ and $|X| < |Y|$, then there must exits i such that $|Y \cap V_i| > |X \cap V_i|$, which implies adding any element in $V_i \cap (Y \setminus X)$ to X with maintain the independence. This type of matroid is called *partition matroid*, which is exactly the matroid that we applied in the task of word alignment (Chapter 6).

There are many other matroids. For example, *linear matroids* are those defined from a matrix, where a set of column indices is independent if the corresponding columns of the matroid are linear independent. Another very important class of matroids in combinatorial optimization is *graphic matroid* in which given a graph, independent sets are defined to be those subsets of edges which are forests (no cycles). A graphic matroid is also a linear matroid. Refer to (Welsh, 1976; Oxley, 2011) for more details.

2.2.1 Greedy algorithm and matroids

The greedy algorithm is one of the most commonly used algorithmic paradigms. A greedy algorithm tries to solve an optimization problem by always choosing a next step that is locally optimal. This will generally lead to a locally optimal solution. However, in some cases, a greedy algorithm will also lead to global optimal solutions. A well-known example is Kruskal's algorithm (Kruskal, 1956): although greedy, it finds the global optimal solution for the minimum spanning tree problem. A natural question then arises: when do greedy algorithms find the optimal solution?

It turns out that this question can be answered for a wide class optimization problems, and this class of problems is those that involves matroids.

Precisely, define a *subset system*, $(V, \mathcal{I})$, where V is the ground set, and $\mathcal{I}$ is a collection of subsets of V such that $\mathcal{I}$ is closed under inclusion, that is, if $X \subseteq Y$ and $Y \in \mathcal{I}$, then $X \in \mathcal{I}$. The optimization problem of interest is as follows: given any weight function $\boldsymbol{w} : V \to \mathbb{R}$, we want to find a set $X \in \mathcal{I}$ maximizing $\boldsymbol{w}(X)$. The *greedy algorithm* consists of setting $X = \emptyset$, and next repeatedly adding $v \in V \setminus X$ to X with $X \cup \{v\} \in \mathcal{I}$ and with w_v as large as possible. The algorithm stops if no such v is available. The algorithm is outlined in Algorithms 1.

Algorithm 1: The greedy algorithm for linear optimization over a subset system.

Input : A subset system $(V, \mathcal{I})$, and $\boldsymbol{w} \in \mathbb{R}^{|V|}$.

Output: $X \in \mathcal{I}$ such that $\boldsymbol{w}(X)$ is maximized.

$X \leftarrow \emptyset$;

Sort the elements in V as $v_1, \cdots, v_{|V|}$ such that $w(v_1) \geq \cdots \geq w(v_{|V|})$;

for $i = 1, \cdots, |V|$ **do**

  if $X \cup \{v_i\} \in \mathcal{I}$ **then**

    $X \leftarrow X \cup \{v_i\}$;

We then have the following theorem.

Theorem 1 (Edmonds, 1970)**.** *For any subset system* $(V, \mathcal{I})$*, the greedy algorithm solves the optimization problem for* $(V, \mathcal{I})$ **if and only if** $(V, \mathcal{I})$ *is a matroid.*

This theorem is bizarre since it indicates that matroids are exactly those structures where the greedy algorithm yields an optimum solution. Actually, an even stronger algorithmic property is that optimization over intersections of two matroids can also be done efficiently. Developments and discoveries matroid intersection methods, as well as matroid union methods, consequently result in algorithms for a wide range of important combinatorial optimization problem, including transversals (Ford & Fulkerson, 1958), tree packing and covering (Nash-Williams, 1961), and etc (Schrijver, 2003).

2.2.2 Submodularity in matroid rank function

Matroids are closely related to submodular functions. In fact, the rank function of any matroid is submodular. The rank function, $r_{\mathcal{M}} : 2^V \rightarrow \mathbb{N}$ of a matroid $\mathcal{M} = (V, \mathcal{I})$ is defined as

$$r_{\mathcal{M}}(X) = \max\{|Y| : Y \subseteq X, Y \in \mathcal{I}\}. \tag{2.5}$$

For the partition matroid introduced above, its rank function is given by:

$$r_{\mathcal{M}}(X) = \sum_{i=1}^{l} \min\{|V_i \cap X|, k_i\}.$$

and the rank of a linear matroid is precisely the rank in the linear algebra sense.

Theorem 2 (Welsh, 1976; Schrijver, 2003; Fujishige, 2005a)**.** *The rank function of any matroid is submodular.*

Connections between matroids and submodular functions not only lie in the matroid rank functions, but also in their associated polytopes.

2.2.3 From matroid polytope to polymatroids

The matroid polytope of a matroid is defined as the convex hull of the incidence vectors all independent sets in that matroid. Precisely, let $\mathcal{A}$ be the set of the characteristic (incidence)

vectors of all independent sets of a matroid $\mathcal{M} = (V, \mathcal{I})$, i.e.,

$$\mathcal{A}_{\mathcal{M}} = \{\mathbf{1}_Y \in \{0,1\}^{|V|} : Y \in \mathcal{I}\},$$

where $\mathbf{1}_: 2^V \to \{0,1\}^{|V|}$ is the characteristic vector operator such that $\mathbf{1}_Y$ is a binary vector where the ith dimension equals to 1 if $i \in Y$ and 0 otherwise. We then denote the matroid polytope of matroid $\mathcal{M}$ as $\mathbf{conv}(\mathcal{A}_{\mathcal{M}})$. The interesting part is that the matroid polytope can be fully characterized by its rank function:

Theorem 3 (Edmonds, 1970; Schrijver, 2003)**.** *Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid with rank function $r_{\mathcal{M}}$, and define polytope*

$$P = \{\boldsymbol{x} \in \mathbb{R}^{|V|}, \boldsymbol{x} \geq \mathbf{0}, \boldsymbol{x}(Y) \leq r(Y), \forall Y \subseteq V\},$$

where $\boldsymbol{x}(Y) = \sum_{i \in Y} x_i$. Then

$$\mathbf{conv}(\mathcal{A}_{\mathcal{M}}) = P.$$

In other words, a matroid polytope can be completely characterized by linear inequalities defined on its rank function. Since rank function is submodular, a generalization of the matroid polytope is then a new type of polytope that is defined on linear inequalities, where the rank function is replaced by an arbitrary submodular function:

$$P_f = \{\boldsymbol{x} \in \mathbb{R}^{|V|}, \boldsymbol{x} \geq \mathbf{0}, \boldsymbol{x}(Y) \leq f(Y), \forall Y \subseteq V\},$$

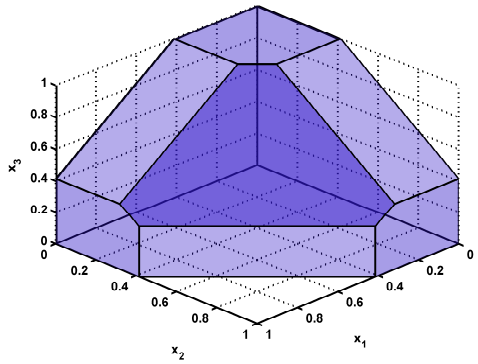
where $f : 2^V \to \mathbb{R}$ is a submodular function. This polytope is called the *submodular polytope* associated with submodular function f. Note P_f is non-empty only when f is non-negative.

We also define a *base polytope*:

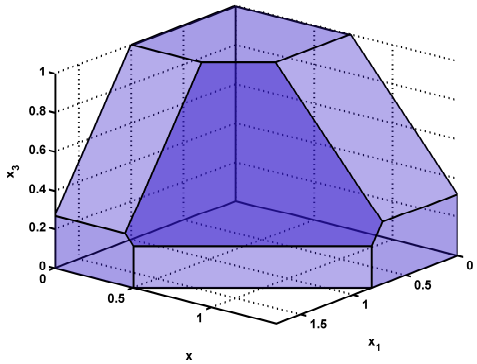
$$B_f = \{\boldsymbol{x} \in P_f, \boldsymbol{x}(V) = f(V)\}.$$

When f is normalized non-decreasing submodular, P_f is known as a *polymatroid*, and a normalized non-decreasing submodular function is also known as a *polymatroid function* (Edmonds, 1970). Figure 2.3 illustrates some examples of polymatroids and the associated base polytopes.

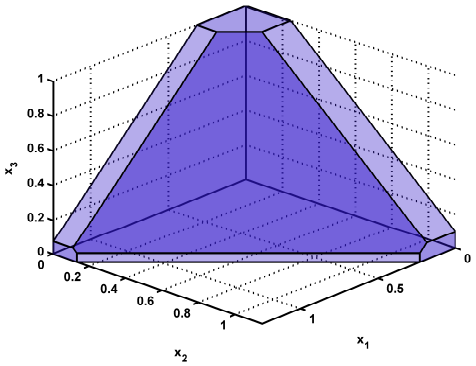
Figure 2.3: Examples of polymatroids and base polytopes. The colored polytope is the polymatroid corresponding to the submodular function indicated in the subcaption, and its darker facet is the associated base polytope. All submodular functions are defined on a ground set of size 3, and $\boldsymbol{x} = (3, 2, 1)$.



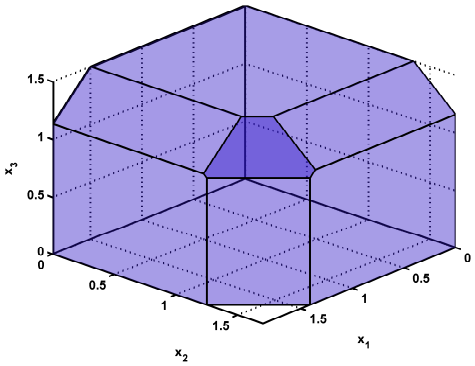
(a) $f(Y) = \sqrt{|Y|}$.



(b) $f(Y) = \sqrt{\boldsymbol{x}(Y)}$.



(c) $f(Y) = (\boldsymbol{x}(Y))^{\frac{1}{5}}$.



(d) $f(X) = |Y| + \frac{1}{2}\sqrt{\boldsymbol{x}(Y)}$.

Polymatroids are the polyhedral generalization of matroids, and it turns out that most nice properties in matroid theory can be generalized to polymatroids and submodular functions. In particular, an extension of the greedy algorithm can solve linear optimization problems over polymatroids (as well as submodular polytopes and base polytopes) exactly. This is amazing since there are an exponentially large number of linear inequalities that characterizes the polytope, yet the optimization can still be done in a simple and greedy

way.

Algorithm 2: The greedy algorithm for linear optimization over polymatroids.

Input : A set function $f : 2^V \to \mathbb{R}$, a polytope of the form

$$P_f = \{\boldsymbol{x} \in \mathbb{R}_+^{|V|}, \boldsymbol{x}(Y) \leq f(Y), \forall Y \subseteq V\}, \text{ and } \boldsymbol{w} \in \mathbb{R}^{|V|}.$$

Output: $\boldsymbol{x} \in P_f$ such that $\boldsymbol{w}^\top \boldsymbol{x}$ is maximized.

Sort the elements in V as $v_1, \cdots, v_{|V|}$ such that $w(v_1) \geq \cdots \geq w(v_{|V|})$;

$X_0 \leftarrow \emptyset$;

for $i = 1$ *to* $|V|$ **do**

  $X_i \leftarrow \{v_1, \cdots, v_i\}$;

  $x(i) = f(X_i) - f(X_{i-1})$;

The greedy algorithm is outlined in Algorithm 2. Similar to the matroid case, we have the following theorem.

Theorem 4 (Edmonds, 1970)**.** *Given a set function $f : 2^V \to \mathbb{R}$, a polytope of the form $P_f = \{\boldsymbol{x} \in \mathbb{R}_+^{|V|}, \boldsymbol{x}(Y) \leq f(Y), \forall Y \subseteq V\}$. Then the greedy algorithm (Algorithm 2) solves problem $\max_{\boldsymbol{x} \in P_f} \boldsymbol{w}^\top \boldsymbol{x}$ for any $\boldsymbol{w} \in \mathbb{R}^{|V|}$* **if and only if** *$f$ is normalized non-decreasing submodular (i.e. P_f is a polymatroid).*

2.3 Continuous Extensions of Submodular Functions

In this subsection, we show several continuous functions that extend submodular functions, which reveal the connection of submodularity to convexity as well as concavity.

2.3.1 Lovász extension

Submodularity can be seen as the discrete analog of convexity (Lovász, 1983). An important connection between convexity and submodularity concerns the continuous extension of a general set function, namely the *Lovász extension.*

Definition 5 (Lovász extension (Lovász, 1983; Bach, 2011))**.** *Given any real vector $\boldsymbol{x} \in \mathbb{R}^{|V|}$, we denote the sorted elements of ground set as $v_1, \cdots, v_{|V|}$ such that $x(v_1) \geq \cdots \geq$*

$x(|V|)$. *Define* $X_0 = \emptyset$ *and* $X_i = \{v_1, \cdots, v_i\}$ *for* $i = 1, \cdots, |V|$. *Then the* Lovász extension $f^L : \mathbb{R}^{|V|} \to \mathbb{R}$ *corresponding to a general set function* $f : 2^V \to \mathbb{R}$, *which is not necessarily submodular, can be equivalently written as follows:*

$$f^L(\boldsymbol{x}) = \sum_{k=1}^{|V|-1} (x(v_k) - x(v_{k+1}) f(X_k) + x(|V|) f(X_{|V|}), \tag{2.6}$$

$$f^L(\boldsymbol{x}) = \sum_{k=1}^{|V|} x(v_k)(f(X_k) - f(X_{k-1})), \tag{2.7}$$

$$f^L(\boldsymbol{x}) = \int_{\min_{i=1,\cdots,|V|} x(i)}^{+\infty} f(\{i|x(i) \geq y(i)\}) d\boldsymbol{y} + f(V) \min_{i=1,\cdots,|V|} x(i) \tag{2.8}$$

$$f^L(\boldsymbol{x}) = \int_{0}^{+\infty} f(\{i|x(i) \geq y(i)\}) d\boldsymbol{y} + \int_{-\infty}^{0} (f(\{i|x(i) \geq y(i)\}) - f(V)) d\boldsymbol{y}. \tag{2.9}$$

Let $\mathbf{1}_Y$ be the characteristic vector of set Y. Obviously, there is a one-to-one correspondence between $\mathbf{1}_Y$ and Y. The Lovász extension f^L is a natural extension of f in the sense that it satisfies the following:

$$f^L(\mathbf{1}_Y) = f(Y), \forall Y \subseteq V.$$

Moreover, the following relationship between the submodularity of f and the convexity of f^L is given:

Theorem 5 (Lovász, 1983)**.** *For a set function* $f : 2^V \to \mathbb{R}$ *and its Lovász extension* $f^L : \mathbb{R}^n \to R$, f *is submodular if and only if* f^L *is convex.*

This theorem builds a bridge between submodularity and convexity. It allows us to reduce discrete optimization to continuous optimization. For instance, instead of minimizing f over the subsets of V, it suffices to minimize f^L over the cube $[0, 1]^{|V|}$. I.e.,

$$\min_{X \subseteq V} f(X) = \min_{\boldsymbol{x} \in \{0,1\}^{|V|}} f^L(\boldsymbol{x}) = \min_{\boldsymbol{x} \in [0,1]^{|V|}} f^L(\boldsymbol{x}).$$

Since f^L can be evaluated efficiently, it yields an efficient algorithm for minimizing f using the standard techniques of convex optimization.

For a submodular function, its Lovśz extension can be written as

$$f^L(\boldsymbol{x}) = \max_{\boldsymbol{y} \in P_f} \boldsymbol{x}^\top \boldsymbol{y}. \tag{2.10}$$

Moreover, the Lovśz extension of a set function is equivalent to its convex closure (Dughmi, 2009) only when it is submodular.

Definition 6 (Convex and concave closures)**.** *Given a set function $f : 2^V \to \mathbb{R}$, let $\mathcal{D}_{\boldsymbol{x}}$ be a distribution over 2^V (all the subsets of ground set) such that $\mathbb{E}_{Y\sim\mathcal{D}}\mathbf{1}_Y = \boldsymbol{x}$. Then the convex closure $f^- : [0,1]^{|V|} \to \mathbb{R}$ is*

$$f^-(\boldsymbol{x}) = \min_{\mathcal{D}_{\boldsymbol{x}}} \mathbb{E}_{Y\sim\mathcal{D}_{\boldsymbol{x}}}[f(Y)]$$

or equivalently,

$$f^-(\boldsymbol{x}) = \min\left\{ \sum_{Y\subseteq V} p_Y f(Y) \;\middle|\; p_Y \in \mathbb{R}_+, \sum_{Y\subseteq V} p_Y \mathbf{1}_Y = \boldsymbol{x}, \sum_{Y\subseteq V} p_Y = 1 \right\}.$$

Similarly, the concave closure $f^+ : [0,1]^{|V|} \to \mathbb{R}$ is

$$f^+(\boldsymbol{x}) = \max_{\mathcal{D}_{\boldsymbol{x}}} \mathbb{E}_{Y\sim\mathcal{D}_{\boldsymbol{x}}}[f(Y)]$$

or equivalently,

$$f^+(\boldsymbol{x}) = \max\left\{ \sum_{Y\subseteq V} p_Y f(Y) \;\middle|\; p_Y \in \mathbb{R}_+, \sum_{Y\subseteq V} p_Y \mathbf{1}_Y = \boldsymbol{x}, \sum_{Y\subseteq V} p_Y = 1 \right\}.$$

Intuitively, the convex closure of a set function f is the point-wise highest continuous function such that it always lowerbounds f in $[0,1]^{|V|}$.

Theorem 6 (Vondrák, 2010)**.** *For a set function f, its Lovász extension f^L and convex closure f^- are identical if and only if f is submodular.*

2.3.2 Multilinear extension

Definition 7. *For a set function $f : 2^V \to \mathbb{R}$, its multilinear extension, $f^M : [0,1]^{|V|} \to \mathbb{R}$, is*

$$f^M(\boldsymbol{x}) = \sum_{Y\subseteq V} f(Y) \prod_{i\in Y} x_i \prod_{j\in V\setminus Y} (1 - x_j). \tag{2.11}$$

Alternatively, f^M can be seen as the expectation of $f(Y_{\boldsymbol{x}})$ where $Y_{\boldsymbol{x}}$ is a random set such that $i \in Y_{\boldsymbol{x}}$ with probability x_i. I.e., $f^M(\boldsymbol{x}) = \mathbb{E}[f(Y_{\boldsymbol{x}})]$. Note that $\mathbf{1}_{Y_{\boldsymbol{x}}} = \boldsymbol{x}$. Therefore we have $f^+(\boldsymbol{x}) \geq f^M(\boldsymbol{x}) \geq f^-(\boldsymbol{x})$ by definition.

The multilinear extension can be derived for any set function but it acquires particularly nice properties for submodular functions. In particular,

Theorem 7. *Given a set function $f : 2^V \to \mathbb{R}$ and its multilinear extension $f^M : [0,1]^{|V|} \to \mathbb{R}$. f is submodular if and only if $\frac{\partial^2 f^M}{\partial x_i \partial x_j} \leq 0$ for all $i, j \in V$.*

It implies that the multilinear extension of a submodular function is concave along any non-negative direction vector, although it is not necessarily concave in all directions.

Multilinear extension of a submodular function has been used in (Calinescu *et al.* , 2007; Kulik *et al.* , 2009) where the submodular optimization problem with matroid constraints is solved by resorting to its multilinear extension followed by pipage rounding (a technique introduced by (Ageev & Sviridenko, 2004)). The evaluation of f^M, however, requires $2^{|V|}$ queries to the value oracle of f, which is obviously affordable. Sampling methods have to be used to approximately evaluate f^M in practise.

2.3.3 Are submodular functions more analogous to convex or to concave functions?

Is a submodular function more "convex" or more "concave"? On the one hand, submodular functions are analogous to convex functions. The Lovśz extension of a submodular function is convex, and the minimum of a submodular function can be found in polynomial time, similar to the amenability of minimization of a convex function. On the other hand, submodular functions also exhibit some aspects of concavity. The property of diminishing returns can be seen as a concave behavior in which gain is compressed more in a larger context. For instance, the function $f(X) = g(|X|)$ is submodular if and only if g is a concave function (Lovász, 1983). Indeed, the analogy is nebulous.

Nevertheless, submodular functions share a number of properties in common with convex and concave functions (Lovász, 1983), including their wide applicability, their generality, their multiple options for their representation, and their closure under a number of common operators (including mixtures, truncation, complementation, and certain convolutions). For

example, if a collection of functions $\{f_i\}_i$ is submodular, then so is their weighted sum $f = \sum_i \alpha_i f_i$ where α_i are nonnegative weights. It can be shown that submodular functions also have the following composition property with concave functions:

Theorem 8. *Given functions $f : 2^V \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$, the composition $h = g \circ f : 2^V \to \mathbb{R}$ (i.e., $h(S) = g(f(S))$) is nondecreasing submodular, if g is non-decreasing concave and f is nondecreasing submodular.*

Proof. See Appendix A. □

This property will be quite useful when constructing submodular functions for real-world applications.

2.4 Submodular Function Optimization

2.4.1 Minimizing submodular functions

As convexity makes continuous functions more amenable to optimization, many combinatorial optimization problems can be solved optimally or near-optimally in polynomial time only when the underlying function is submodular. It has been shown that any submodular function can be *minimized* in polynomial time.

The first polynomial-time algorithm for submodular function minimization is due to (Grötschel *et al.* , 1981). The algorithm employs the ellipsoid method, which was later used to develop the first polynomial-time algorithm for linear programming. In spite of the polynomial time complexity, the ellipsoid method is not very efficient in practice.

Combinatorial strongly polynomial algorithms have been developed by (Schrijver, 2000a) and (Iwata *et al.* , 2001a). Both of the algorithms are based on the same framework due to (Cunningham, 1984) where combinatorial strongly polynomial algorithm was developed for solving the membership problem for matroid polyhedra (a special case of submodular function minimization). The Iwata-Fleischer-Fujishige Algorithm (Iwata *et al.* , 2001a) employs a scaling scheme that was originated from the submodular flow problem (Fujishige, 2005a; Iwata *et al.* , 2006), while (Schrijver, 2000a) directly achieves a strongly polynomial bound by introducing a novel subroutine of lexicographic augmentation. Note that the

strongly polynomial Iwata-Fleischer-Fujishige algorithm runs in $O(\gamma n^7 \log n)$ time while the complexity of Schrijver's algorithm is $O(n^8 + \gamma n^7)$, where γ is the time of invoking the function evaluation oracle. Iwata (2003) combined the scaling scheme and a push/relabel technique using Shrijver's subroutine and achieves $O((\gamma n^4 + n^5) \log M)$ running time for general submodular function minimization, where M is the maximum absolute value of the function. The currently fastest known *strongly* polynomial time algorithm for submodular minimization is Orlin's algorithm (Orlin, 2009), which runs in $O(n^5\gamma + n^6)$ time. All these combinatorial algorithms, although with polynomial-time complexity, do not scale well to most real world problems (e.g., problems of interests in this thesis).

Algorithms that are more practically useful for large real world problem are also available. In particular, when the submodular function has special structure, faster algorithms can be applied. One well-known example is the graph cut function, which is submodular and can be minimized efficiently. For symmetric submodular function, Queyranne (1998) showed that the minimization problem can be solved in $O(n^3)$ time. For *general* submodular function minimization, a minimum-norm-point algorithm (Fujishige, 2005a; Fujishige *et al.* , 2006) is currently widely accepted as the most practical algorithm. In Chapter 3, we will discuss this algorithm in detail and study how practical it is in the context of problems with scales of the sizes needed for a typical NLP problem.

Submodular minimization with additional constraints has also been recently studied (Svitkina & Fleischer, 2008; Goel *et al.* , 2009; Iwata & Nagano, 2009; Nagano *et al.* , 2011).

2.4.2 Maximizing Submodular Functions

Minimizing a submodular function is an important problem whose solution implies the solution of many very general combinatorial optimization problems. On the other hand, the problem of maximizing a submodular set function is also important with many practical applications. Unlike submodular minimization, maximizing a submodular function turns out to be much harder. Actually, quite a few NP-hard problems (for example, the maximum graph cut problem) are instances of submodular function maximization. Therefore, we have

to resort to approximation algorithms for submodular maximization.

Non-monotone submodular maximization

Maximizing a non-monotone submodular function is NP-hard in general. Feige *et al.* (2007) gave the first constant approximation algorithms for maximizing non-negative submodular functions. In particular, they gave a deterministic local search $\frac{1}{3}$-approximation and a randomized $\frac{2}{5}$-approximation algorithms for maximizing non-negative submodular functions. Gharan & Vondrák (2011) recently improves the approximation factor from $\frac{2}{5}$ to 0.41 for submodular maximization by using simulated annealing.

Note that submodular maximization is still an active research area. Recently, approximation algorithms have been designed for non-monotone submodular maximization subject to various constraints. Lee *et al.* (2009a) developed the first approximation algorithm for non-monotone maximization subject to k matroid constraints, or k knapsack constraints, where the algorithms are based on the local search with k-swaps. Vondrák (2009) showed a $(\frac{1}{6} - o(1))$-approximation algorithm for maximizing a non-monotone submodular function over the bases of a matroid. In (Vondrák, 2009), a randomized algorithm is also shown to have approximation factor of 0.309 for the problem of maximizing non-monotone submodular function over a matroid independence constraint, which improves the 0.25−approximate algorithm in (Lee *et al.* , 2009a), and is recently further improved by Gharan & Vondrák (2011), who show a 0.325−approximation algorithm. Gupta *et al.* (2010) gave algorithms that work for k-independence systems (which generalize constraints given by the intersection of k matroids). The algorithms essentially reduce the non-monotone maximization problem to multiple runs of the greedy algorithm previously used in the monotone case. Fadaei *et al.* (2011) recently show that using the continuous greedy process, a 0.13−factor approximation algorithm can be obtained for non-monotone submodular maximization over any solvable down-monotone polytope.

While maximizing non-monotone submodular functions with or without additional constraints is hard in general, maximizing monotone submodular function with certain constraints is more amenable, where efficient and effective algorithms are available.

Maximizing a monotone submodular function under a cardinality constraint

Maximization of a monotone submodular function under a cardinality constraint can be solved using a greedy algorithm (Nemhauser *et al.* , 1978) within a constant factor (0.63) of being optimal. In particular, the greedy algorithm starts with $S = \emptyset$, and iteratively adds the element $s^* \in V \setminus S$ that yields the greatest increment of the objective function value:

$$s^* \in \text{argmax}_{s \in V \setminus S} f(S \cup \{s\}) - f(S). \tag{2.12}$$

This repeats until either $|S| = K$ or no further increment occurs. The following theorem gives the performance guarantee for the greedy algorithm.

Theorem 9 (Nemhauser *et al.* , 1978)**.** *For normalized monotone submodular function f, the set S_G^* obtained by the greedy algorithm is no worse than a constant fraction $(1 - 1/e)$ away from the optimal value:*

$$f(S_G^*) \geq (1 - 1/e) \max_{S \subseteq V : |S| \leq K} f(S). \tag{2.13}$$

The greedy algorithm, moreover, is likely to be the best we can do in polynomial time, unless $P = NP$.

Theorem 10 (Feige, 1998a)**.** *Unless P=NP, there is no polynomial-time algorithm that guarantees a solution S^* with*

$$f(S^*) \geq (1 - 1/e + \epsilon) \max_{S \subseteq V : |S| \leq K} f(S), \epsilon > 0. \tag{2.14}$$

The submodular maximization problem with cardinality constraint can be seen as a selection problem: we want to select a good subset S of the whole set V that maximizes some objective function, given the constraint that the size of S is no larger than K (our budget). Such selection problems arise in many applications. For instance, in active learning, we wish to acquire labels only for the most informative subset of the unlabeled data (which is usually abundant) given limited budget and/or time for labeling. As we will see, the greedy algorithm has been successfully applied to such data selection problem about how to select a good subset for word-level transcription under a given fixed transcription budget labeling speech data (Lin & Bilmes, 2009), as well as document summarization tasks (Lin *et al.* , 2009).

Maximizing a monotone submodular function under matroid constraints

Let $\mathcal{M}_1, ..., \mathcal{M}_k$ be k arbitrary matroids on the common ground set V. For each matroid $\mathcal{M}_j$, denote its independent sets by $\mathcal{I}_j$. I.e., $\mathcal{M}_j = (V, \mathcal{I}_j)$. The maximization of submodular function under matroid constraints can then be written as:

Problem 2.

$$\max \left\{ f(S) : S \in \bigcap_{j=1}^{k} \mathcal{I}_j \right\}. \tag{2.15}$$

Early work in (Nemhauser *et al.* , 1978; Fisher *et al.* , 1978) showed that the greedy and local search algorithms give a $(k+1)$-approximation of the above problem. Lee *et al.* (2009b) showed a local search algorithm with approximation guarantee $1/(k + \epsilon)$. When $k = 1$, Vondrák (2008) shows a continuous greedy algorithm followed by pipage rounding with approximation factor $1 - 1/e$ (≈ 0.63). The algorithm, however, is based on the multilinear extension (see Definition 7). As discussed in Section 2.3.2, evaluation of the multilinear extension of a set function requires exponentially large number of function queries. Thus the algorithm in (Vondrák, 2008) is not practical for problems with NLP scale (e.g., the word alignment problem as we will see in Chapter 6), even with approximate evaluation of the multilinear extension.

Maximizing a monotone submodular function under knapsack constraints

Let $c^1, ...c^k$ be k cost vectors corresponding to knapsacks having capacities $b_1, ..., b_k$ respectively. The knapsack constraint problem can then be expressed as:

Problem 3.

$$\max \left\{ f(S) : \sum_{j \in S} c_j^i \leq b_i, \forall 1 \leq i \leq k, S \subseteq V \right\}. \tag{2.16}$$

(Sviridenko, 2004) showed that a modified greedy algorithm with partial enumeration gives an $(1 - 1/e)$-approximation algorithm to monotone submodular maximization subject to a single knapsack constraint. (Kulik *et al.* , 2009) more recently showed that the same approximation bound can be obtained when there are constant number of knapsacks. These

algorithms, again do not scale well to the problem sizes of practical interest. In Section 3.4, we will further discuss this problem, showing a simplified and practical algorithm, along with its theoretical justification and applications in summarization (Chapter 4).

Chapter 3

ALGORITHMS

In this chapter, we introduce the algorithm aspect of this thesis's work related to submodular function optimizations. In particular, three problems are studied: general submodular function minimization, budgeted submodular maximization, and modular maximization with submodular knapsack constraint.

For the study on submodular function minimization, the goal is to find an algorithm that is practical for problems at the scale of typical problems. Although combinatorial polynomial algorithms are available for submodular function minimization, they are practically useless for large problem instances due to their approximate $O(n^5)$ time complexity. On the other hand, the minimum-norm point algorithm (Fujishige-Wolfe algorithm) is widely regarded as a practical algorithm for submodular minimization. Its theoretical complexity, however, is still an open problem. In this chapter, in contrast to what has been previously reported about minimum norm's practical performance, we empirically show that the complexity of minimum-norm-point algorithm can be as bad as the combinatorial algorithms. To accelerate the minimum-norm-point algorithm, we further introduce two heuristics for early termination of the algorithm without violating optimality. Experiment results show that the accelerated minimum-norm-point algorithm achieves phenomenal (as large as $75,000$ times) speedup compared to the algorithm without using the early stopping heuristic based on a duality gap.

The second problem investigated in this chapter is the budgeted submodular maximization problem. Similar to the case of submodular minimization, although algorithms with rigorous performance guarantees are available, they are not practical for NLP problems. We show that a modified greedy algorithm is not only efficient but also performs near-optimally with constant approximation factor. We also address the un-calibration issue that arises in practise by scaling the cost in the greedy heuristic, and show that performance of the

algorithm is still bounded.

Lastly, we study the problem of modular maximization with submodular knapsack constraint, which we show can be reduced to submodular minimization with cardinality lower-bound. An approximation algorithm is then immediate based on the bi-criteria algorithm for submodular minimization with cardinality lower-bound (Svitkina & Fleischer, 2008). We also show the hardness of this problem.

Note that all the problems studied here are motivated by real-world applications. Submodular minimization arises when we create a speech corpus with limited vocabulary (Chapter 7), budgeted submodular maximization is used in document summarization (Chapter 4), and submodular knapsack problem is also motivated by the speech corpus creation task.

3.1 Minimum-norm-point Algorithm for Submodular Minimization

Submodular functions can be minimized in polynomial time. The first strong polynomial time algorithm for submodular function minimization (SFM) resulted from an observation of Grötschel *et al.* (1981), who realized the implications of the ellipsoid method. However, ellipsoid methods are slow and impractical. Combinatorial algorithms for SFM that are strongly polynomial were then discovered by Iwata *et al.* (2001b) and Schrijver (2000b) simultaneously and independently. The currently fastest strongly polynomial combinatorial algorithm achieves a running time of $O(n^5T + n^6)$ (Orlin, 2009) (where T is the time for function evaluation), which is far from practical for large problem instances.

It has been reported in (Fujishige & Isotani, 2011) that the minimum-norm (MN) algorithm is much faster in practise compared to the combinatorial algorithms. However, MN algorithm's worst-case complexity is still an open question. Moreover, experiments in (Fujishige & Isotani, 2011) only cover a limited number of types of submodular functions at a limited range of problem sizes. In particular, only two types of submodular functions were tested, one is the Iwata function, which turns out to be trivially easy for MN, and the other is the graph cut function, where fast min-cut/max-flow algorithms are applicable.

In this section, we test MN algorithm on larger and richer families of submodular functions that are not limited to graph-representable functions. We study the empirically complexity of MN on a family of submodular functions based on bipartite graphs. We show

that, in certain cases, the complexity of minimum-norm-point algorithm can be as bad as $O(n^6)$ on running time and $O(n^4)$ on the number of function oracle calls, which implies that the theoretical lower bound of the complexity of MN algorithm might have high exponents.

Another contribution shown in this section is the study of early stopping criteria for MN algorithm. Early stopping criteria have been shown to be very effective in improving the runtime performance of many convex optimization problems including the max-flow problem. Such potential heuristic improvements, however, has not yet been fully explored for MN algorithm. In (Fujishige & Isotani, 2011), an early stopping criteria for integral functions is suggested and a possible problem reduction is introduced. In this paper, we introduce two early stopping criteria for MN, one is based on the subgradient of the Lovász extension, and the other is based on the duality gap of the min-max theorem of submodular minimization. Both of these criteria do not violate the optimality of the MN algorithm. Our experimental results show that MN with duality gap acceleration achieves $75,000$ speedup for a type of submodular functions.

3.1.1 Minimum-norm point Algorithm

We assume $f(\emptyset) = 0$ (i.e., f is normalized) without loss of generality.

The are two polyhedra that are closely related to submodular function optimization. One is the *submodular polyhedra*:

$$P(f) = \{x | x \in \mathbb{R}^E, x(S) \leq f(S) \forall S \subseteq E\},$$

and the *submodular base polytope*:

$$B(f) = \{x | x \in P(f), x(E) = f(E)\}.$$

Interestingly, the problem of SFM is equivalent to an optimization problem over the corresponding submodular polyhedra, where the latter can be solved by finding the minimum-norm point in the base polytope (Fujishige, 2005b), i.e., $\min_{x \in B(f)} ||x||^2$. There are several ways to solve the minimum-norm point problem in a polytope, and MN algorithm for SFM is based on Wolfe's algorithm (Wolfe, 1976) back in 1976. One of the most expensive components in Wolfe's algorithm is to solve a linear optimization problem over the polytope,

which can be done by examining all the extreme points in the polytope. In general, if the number of extreme points in the polytope is exponential in the dimension of the space, it is hard to perform such primitive examinations. Fortunately, although the number of extreme points of submodular base polytope is exponentially large, linear optimization over the base polytope $B(f)$ can be done in $O(n \log n)$ time, thanks to submodularity. Therefore Wolfe's algorithm is then ideal for the minimum-norm point problem in base polytope. MN algorithm is also known as Fujishige-Wolfe algorithm. Details of the algorithm can be found in (Fujishige & Isotani, 2011) and (Wolfe, 1976).

We give a comprehensive review of MN algorithm (Fujishige-Wolfe algorithm) below.

3.1.2 Wolfe's algorithm on finding minimum norm point in a polytope

Preliminaries

Given a set of points $P = \{p_1, \cdots, p_m\}$ where $p_i \in \mathbb{R}^n$, the problem of interest is to find the minimum norm point in the convex hull of P:

$$\min_{x \in \mathbf{conv}(P)} \|x\| \tag{3.1}$$

where

$$\mathbf{conv}(P) \triangleq \left\{ x : x = \sum_{i=1}^{m} w_i p_i, \sum_i w_i = 1, w_i \geq 0, i = 1, \cdots, m \right\}. \tag{3.2}$$

While general purpose algorithms is available for this problem, most of them are convergent, non-terminating. Wolfe proposed an algorithm that is terminating, and explicitly uses the representation of x (i.e. convex combination of points in P). The algorithm maintains a set of points $Q \subseteq P$, which is *affinely independent*. Note that when Q are affinely independent, the minimum norm point in the affine hull of Q can easily be found, that is, close form solution for $\min_{x \in \mathbf{aff}(Q)} \|x\|$ is available, where $\mathbf{aff}(Q)$ denotes the affine hull of Q, i.e.,

$$\mathbf{aff}(Q) \triangleq \left\{ x : x = \sum_{p_i \in Q} w_i p_i, \sum_{i : p_i \in Q} w_i = 1 \right\}. \tag{3.3}$$

Wolfe's algorithm repeatedly produces minimum-norm-point x^* for selected set Q. If it turns out that $w_i \geq 0, i = 1, \cdots, m$ for the minimum-norm-point, then x^* of course belongs to $\mathbf{conv}(Q)$ and also a minimum-norm-point there. If $Q \subseteq P$ is suitably chosen, x^* may even be the minimum-norm-point over $\mathbf{conv}(P)$ and thus solve the original problem.

Geometry and correctness

Algorithm 3: Geometric description of Wolfe's algorithm

Input : $P = \{p_1, \cdots, p_m\}, p_i \in \mathbb{R}^n, i = 1, \cdots, m.$

Output: x^*: the minimum-norm-point in $\mathbf{conv}(P)$.

$x^* \longleftarrow p_{i^*}$ where $p_{i^*} \in \text{argmin}_{p \in P} \|p\|$;

$Q \longleftarrow \{x^*\}$;

while 1 **do** `/* major loop */`

1 **if** $x^* = 0$ *or* $H(x^*)$ *separates* P *from origin* **then**

  return : x^*

2 **else**

3 Choose $\hat{x} \in P$ on the near side of $H(x^*)$;

4 $Q = Q \cup \{\hat{x}\}$;

5 **while** 1 **do** `/* minor loop */`

6 $x_0 \longleftarrow \min_{x \in \mathbf{aff}(Q)} \|x\|$;

7 **if** $x_0 \in \mathbf{conv}(Q)$ **then**

8 $x^* \longleftarrow x_0$;

9 **break**;

10 **else**

11 $y \longleftarrow \min_{x \in \mathbf{conv}(Q) \cap [x^*, x_0]} \|x - x_0\|$;

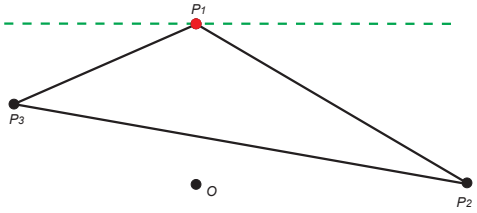
12 Delete from Q points not on the face of $\mathbf{conv}(Q)$ where y lies;

13 $x^* \longleftarrow y$;

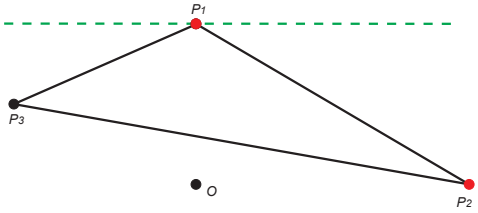
Define $H(x)$ as the hyperplane that is orthogonal to x while containing x, i.e.

$$H(x) \triangleq \left\{ y \in \mathbb{R}^n | x^\top y = \|x\|^2 \right\} \tag{3.4}$$

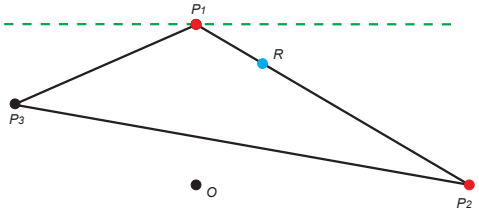
Figure 3.1: An example illustrating how MN algorithm works.



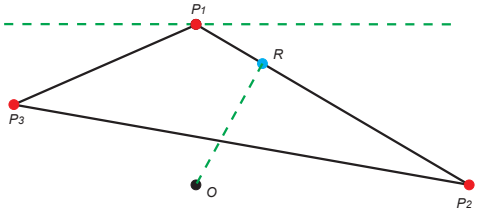
(a) $Q = \{p_1\}, x^* = p_1$. $H(p_1)$ is not a supporting hyperplane of $\mathbf{conv}(P)$.



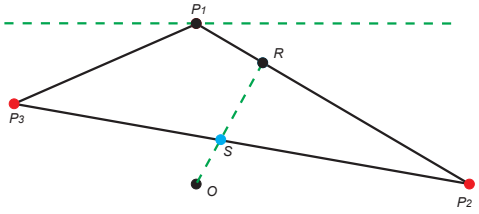
(b) Add p_2 to Q. $Q = \{p_1, p_2\}$.



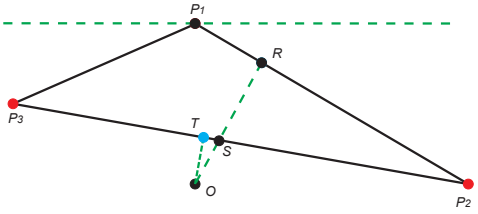
(c) R is the min-norm point in $\mathbf{aff}(\{p_1, p_2\})$.



(d) $H(R)$ is not a supporting hyperplane of $\mathbf{conv}(P)$. Add p_3. $Q = P = \{p_1, p_2, p_3\}$. The origin O is the min-norm point in $\mathbf{aff}(Q)$, and it is not in the interior of $\mathbf{conv}(Q)$.



(e) $S = \min_{x \in \mathbf{conv}(Q) \cap [x^*, x_0]} \|x - x_0\|$ where x_0 is O and x^* is R here. Remove p_1 from Q since it is not on the face where S lies. $Q = \{p_2, p_3\}$



(f) T is the min-norm point in $\mathbf{aff}(Q)$, $H(T)$ separates P from the origin, and therefore is the min-norm point in $\mathbf{conv}(P)$.

Algorithm 13 gives a geometric description of Wolfe's algorithm. Figure 3.1 illustrates an example. More algebra details are given in the next section. We show herein its correctness and finite termination. First of all, Q is always affinely independent.

Lemma 1. *Q in Algorithm 13 is always affinely independent.*

Proof. Q is of course affinely independent when there is at most one point in it. After the initialization, it changes only by deletion of single point or adding a single point. Deletion does not change the independence. Before adding $\hat{x}$ at Line 4, we know x^* is the minimum norm point in $\mathbf{aff}(Q)$. Therefore, x^* is normal to $\mathbf{aff}(Q)$, which implies $\mathbf{aff}(Q) \subseteq H(x^*)$. Since $\hat{x} \notin H(x^*)$, we have $\hat{x} \notin \mathbf{aff}(Q)$. Thus $Q \cup \{\hat{x}\}$ is affinely independent if Q is. □

By Lemma 1, we have for any $x \in \mathbf{aff}(Q)$ such that $x = \sum_i w_i q_i, \sum_i w_i = 1$, the weights w_i are uniquely determined.

Theorem 11. *Algorithm 13 finds the minimum norm point in* $\mathbf{conv}(P)$ *after finite number of iterations of the major loop.*

Proof. Sketch: Note that in the minor loop, we always have $x^* \in \mathbf{conv}(Q)$, since whenever Q is modified, x^* is updated as well (Line 13) such that the updated x^* remains in the updated $\mathbf{conv}(Q)$ Therefore, every time x^* is updated, its norm never increases, i.e., before Line 8, $\|x_0\| \leq \|x^*\|$ since $x^* \in \mathbf{aff}(Q)$ and $x_0 = \min_{x \in \mathbf{aff}(Q)} \|x\|$; similarly, before Line 13, $\|y\| \leq \|x^*\|$. Moreover, there can be no more iterations than the dimension of $\mathbf{conv}(Q)$ in a minor loop begin with a given Q, as when Q reduces to a singleton, minor loop always terminates. Thus a minor loop terminates in finite number of iterations.

Next, note that each time Q is updated at Line 4 and followed by updating x^* with x_0 at Line 8 (i.e. minor loop returns with only one iteration), $\|x^*\|$ strictly decreases. To see that, consider $x^* + \theta(\hat{x} - x^*)$ where $0 \leq \theta \leq 1$. Since both $\hat{x}, x^* \in \mathbf{conv}(Q)$, $x^* + \theta(\hat{x} - x^*) \in \mathbf{conv}(Q)$. Therefore, we have $\|x^* + \theta(\hat{x} - x^*)\| \geq \|x_0\|$, which implies

$$\|x^* + \theta(\hat{x} - x^*)\|^2 = \|x^*\|^2 + 2\theta\left((x^*)^\top \hat{x} - \|x^*\|^2\right) + \theta^2 \|\hat{x} - x^*\|^2 \geq \|x_0\|^2 \tag{3.5}$$

$\hat{x}$ is on the same side of $H(x^*)$ as the origin, i.e. $(x^*)^\top \hat{x} < \|x^*\|^2$. Therefore, for sufficiently small θ ($\theta < \frac{2\left(\|x^*\|^2 - (x^*)^\top \hat{x}\right)}{\|\hat{x} - x^*\|^2}$), we have $\|x^*\|^2 > \|x_0\|^2$. For the same reason, we have $\|x^*\|$ strictly decreases each time Q is updated at Line 4 and followed by updating x^* with y at Line 13. Therefore, in each iteration of major loop, $\|x^*\|$ strictly decreases, and Algorithm 13 must terminate and it can only do so when the optimal is found. □

Finding $\hat{x}$ on the near side of $H(x^*)$

In Algorithm 13, there could be several ways to find an $\hat{x}$ such that it is on the near side of $H(x^*)$ (i.e., the half plane where the origin lies). Ideally, we would like to find an $\hat{x}$ such that the reduction on $\|x^*\|$ can be maximized and thus reduce the number of major iterations. As we see in Eq. 3.5, the reduction on the norm is lower-bounded:

$$\Delta = \|x^*\|^2 - \|x_0\|^2 \geq 2\theta \left(\|x^*\|^2 - (x^*)^\top \hat{x} \right) - \theta^2 \|\hat{x} - x^*\|^2 \triangleq \underline{\Delta} \tag{3.6}$$

$$\tag{3.7}$$

When $0 \leq \theta < \frac{2\left(\|x^*\|^2 - (x^*)^\top \hat{x}\right)}{\|\hat{x} - x^*\|^2}$, we can get the maximal value of the lower bound, i.e.

$$\max_{0 \leq \theta < \frac{2\left(\|x^*\|^2 - (x^*)^\top \hat{x}\right)}{\|\hat{x} - x^*\|^2}} \underline{\Delta} = \left(\frac{\|x^*\|^2 - (x^*)^\top \hat{x}}{\|\hat{x} - x^*\|} \right)^2 \tag{3.8}$$

To maximize the lower bound the norm reduction in each major iteration, we want to find an $\hat{x}$ such that the above lower bound is maximized. In other words, ideally we want to find

$$\hat{x} = \operatorname*{argmax}_{x \in P} \left(\frac{\|x^*\|^2 - (x^*)^\top x}{\|x - x^*\|} \right)^2 \tag{3.9}$$

such that larger reduction can be guaranteed.

This problem, however, is at least as hard as the MN problem itself as we have a quadratic term in the denominator. Alternatively, we could resort to maximizing the numerator, i.e. find

$$\hat{x} = \operatorname*{argmax}_{x \in P} \|x^*\|^2 - (x^*)^\top x = \operatorname*{argmin}_{x \in P} (x^*)^\top x, \tag{3.10}$$

Intuitively, by solving the above, we find $\hat{x}$ such that it has the largest distance to the hyperplane $H(x^*)$, and this is exactly the strategy used in Wolfe's algorithm (Wolfe, 1976).

Also, the solution $\hat{x}$ here can be used to determine whether the hyperplane $H(x^*)$ separates $\mathbf{conv}(P)$ from the origin. Indeed, if the point that has the greatest distance to $H(x^*)$ is not on the side where origin lies, then $H(x^*)$ separates $\mathbf{conv}(P)$ from the origin. Formally, we terminate the algorithm if

$$(x^*)^\top \hat{x} \geq \|x^*\|^2,$$

where $\hat{x}$ is the solution of Eq. 3.10. In practice, without any provision, the above optimality test might never be passed. As suggested in (Wolfe, 1976), we introduce a tolerance parameter $\epsilon > 0$, and terminates the algorithm if

$$(x^*)^\top \hat{x} > \|x^*\|^2 - \epsilon \max_{x \in Q} \|x\|^2 \tag{3.11}$$

Notice that when $\mathbf{conv}(P)$ is a submodular base polytope, problem in Eqn 3.10 can be solved efficiently by a greedy algorithm despite that the number of the extreme points could be exponentially large, and this is one of the major reasons that why MN algorithm is applicable to submodular function minimization.

3.2 On Complexity of Minimum-norm-point Algorithm

The complexity of MN algorithm is still an open problem to date. Two obvious facts, however, is that each major iteration requires $O(n)$ function oracle calls, and the complexity of each major iteration could be at least $O(n^3)$ due to the affine projection step (solving a linear system). Therefore, the complexity of each major iteration is

$$O(n^3 + n^{1+p})$$

where we assume that each function oracle call requires $O(n^p)$ time.

Since the number of major iteration is unknown, the complexity of MN is unknown either. In this section, we empirically evaluate the number of major iterations required for various submodular functions. In particular, based on extensive experiments of submodular minimization on various submodular functions, we show two findings. First, the number of major iterations seems to be function dependent. An extreme example is that for Iwata's function (Fujishige & Isotani, 2011), the number of major iterations required is constant (i.e., $O(1)$), regardless of the size of ground set. On the other hand, as we will see, for some type of submodular functions arises in practise, the number of major iterations is at least $O(n^3)$. Second, in contrast to previously reported MN's efficiency compared to combinatorial algorithms Fujishige & Isotani (2011), we show that for some functions, the complexity of MN algorithm could be as bad as the combinatorial algorithms.

So far, there is no generally accepted test bed of instances of SFM. In (Fujishige & Isotani, 2011), MN algorithm was tested on Iwata's function and on graph cut functions with graph generated using `GENRMF` available from DIMACS Challenge. Bach (2011) tested MN algorithm (along with some other continuous optimization based approximate submodular minimization algorithms) on standard graph cut functions as well as the symmetric mutual information function on a relatively small scale of ground sizes (up to 400).

In our experiments, we firstly tested MN algorithm as well as our C++ implementation on the min-cut problem with standard graphs. We then tested MN on a richer family of submodular function, i.e, submodular functions derived from bipartite graphs, showing that MN can be as bad as $O(n^7)$ in running time.

3.2.1 Cut Functions

We tested MN algorithm for graph cut minimization. We used `GENRMF` available from DIMACS Challenge to generate the graphs ("long" type). Each experiments were performed three times and the minimum elapsed real time between invocation and termination of the experiment are reported in Table 3.2.1

Table 3.1: Comparison of my MN code with that used in (Fujishige & Isotani, 2011)

		C (Fujishige)		C++	
num. of vertices	num. of edges	CPU Time (sec)	Memory (Kb)	CPU Time (sec)	Memory (Kb)
392	1687	4.42	12672	1.49	13872
576	2528	24.71	24144	5.85	30624
810	3609	74.79	44896	15.53	41584
1100	4960	228.95	79920	86.32	100272
1452	6611	471.9	136224	208.16	123840
1872	8592	881.88	217616	355.1	199424
2366	10933	2208.47	348128	792.95	402864
2940	13664	3618.2	520000	963	479136

Our implementation of the minimum norm point algorithm in C++ is about four times

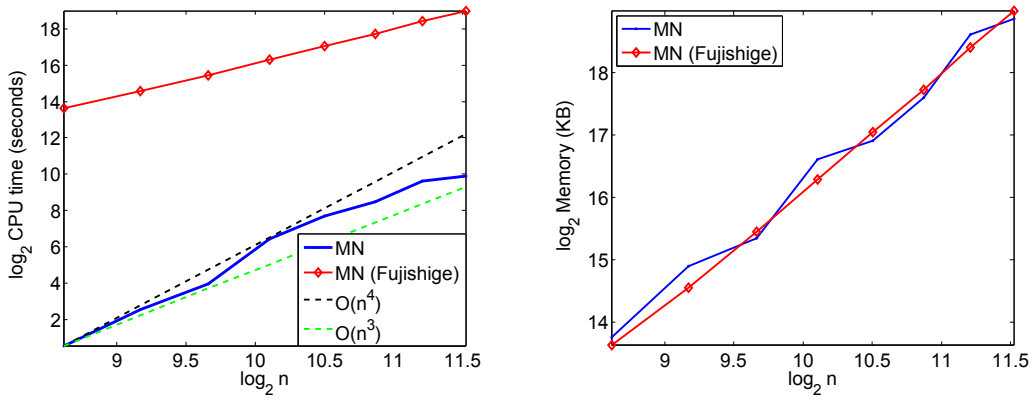


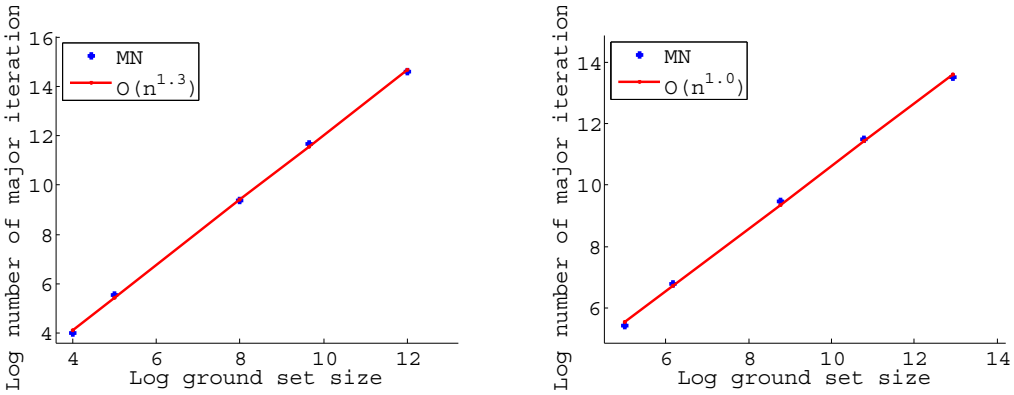
Figure 3.2: Comparison of my min-norm code with that by Fujishige. The data are graphs generated by `GENRMF` with "long" type, and n refers to the number of nodes. The tolerance was set to $\epsilon = 10^{-10}$.

faster than the C code in (Fujishige & Isotani, 2011) [1], ensuring that our results are not due to a slow implementation. Figure 3.2 compares both implementations and shows that our implementation is four times faster while not using more memory. As we can see, the empirically complexity of MN algorithm for graph cut function is between $O(n^3)$ and $O(n^4)$, which is consistent with the empirical complexity of MN reported in (Fujishige & Isotani, 2011; McCormick, 2006).

We also evaluated complexity on the major iterations, as shown in Figure 3.3. As we can see, the number of major iteration required by MN algorithms on graph cut functions is about $O(n)$. The evaluation of a cut function takes $O(n^2)$ time. Therefore we infer that the complexity MN algorithms on graph cuts is about $O(n^3)$, which coincides the actual time complexity as shown in Figure 3.2.

Note that the currently fastest strongly polynomial combinatorial algorithm achieves a running time of $O(n^5T + n^6)$ (Orlin, 2009) (where T is the time for function evaluation). For graph cut functions, time complexity of combinatorial algorithm is then about $O(n^7)$; on the other hand, the time complexity of MN on cuts is $O(n^3)$, which is much faster than

[1] The speed improvement comes from better data structure as well lazy evaluation of function increments.



(a) long (b) wide

Figure 3.3: Number of major iterations of MN algorithm on graph cuts.

currently available combinatorial algorithms for submodular minimization. The superiority of MN algorithm over combinatorial algorithms for minimizing cut functions has previously been demonstrated in (Fujishige & Isotani, 2011), based on which it is commonly accepted that MN algorithm is the most efficient and practical algorithm for general submodular minimization (Nagano *et al.* , 2011; Bach, 2011). However, Fujishige & Isotani (2011) only tested limited types of submodular functions. In the following section, we show MN algorithm's time complexity on a rich family of submodular functions, namely submodular function defined on bipartite graphs, indicating that MN algorithm could be as expensive as combinatorial algorithms on certain types of submodular functions that are particularly useful in many real-world applications.

3.2.2 Submodular Functions on Bipartite Graph

As mentioned in Section 2.1.2, submodular functions defined on bipartite graph could be quite useful for many applications. In the section, we tested MN algorithms for submodular functions defined on bipartite graphs.

Let E and F be the set of left and right nodes in the bipartite graph respectively. Denote $\mathcal{N} : E \to F$ as the neighboring function, and $m : 2^V \to \mathbb{R}$, $w : 2^F \to \mathbb{R}$ are the integral

modular functions. The submodular functions we consider there have the following form:

$$f(S) = -m(S) + \lambda \left(w(\mathcal{N}(S))\right)^{\alpha}$$

where $S \subseteq E$, $\lambda > 0$ and $0 < \alpha \leq 1$.

The bipartite graphs were generated by sampling a large bipartite graph arising from a real world application (Lin & Bilmes, 2010a, 2011b). In the large bipartite graph, left nodes represents speech utterances and right nodes represents distinct words (vocabulary), and a left node (utterance) is connected to a right node (word) if the word appears in the utterance. We have $|E| = 54915$ and $|F| = 6871$.

We used two types of modular functions. In the first type, all singleton values equal to 1, i.e.

$$m_1(e) = 1, \forall e \in E$$
$$w_1(f) = 1, \forall f \in F$$

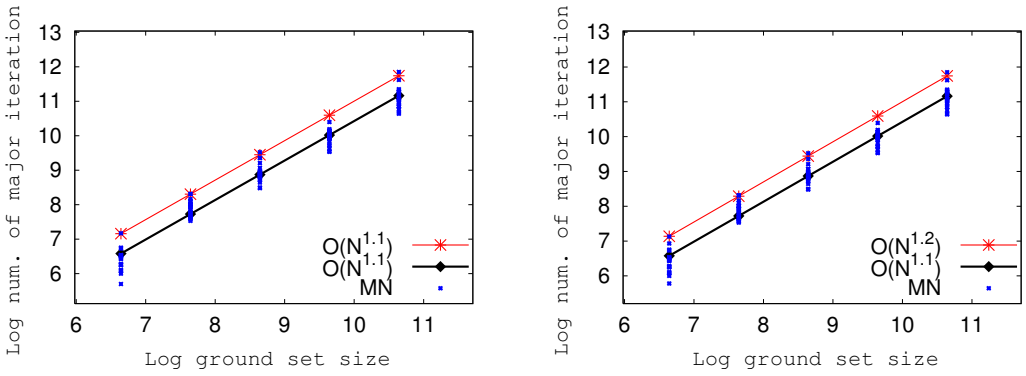
In the second type, $m(e), e \in E$ equals to the number of word tokens in the utterance that node e corresponding to, and $w(f), f \in F$ equals to a hundred divided by the number of syllables in the word that node f corresponding to. I.e.,

$$m_2(e) = \text{WordCount}_e, \forall e \in E$$
$$w_2(f) = \frac{100}{\text{PronunciationCount}_f}, \forall f \in F$$

We measure both average-case complexity and worst-case complexity on the number of major iterations. Formally, for inputs drawn from $\mathcal{G} = \bigcup_n \mathcal{G}_n$, and m instance, $\{G_n^1, \cdots, G_n^m\}$, drawn from $\mathcal{G}_n$ for each n, we have worst-case complexity $T_{\text{worst}}(n)$ and average-case complexity $T_{\text{average}}(n)$ defined and approximately computed as follows:

$$T_{\text{worst}}(n) = \max_{G \in \mathcal{G}_n} t(G) \approx \max_{G \in \{G_n^1, \cdots, G_n^m\}} t(G) \tag{3.12}$$

$$T_{\text{average}}(n) = \mathbb{E}_{G \in \mathcal{G}_n} t(G) \approx \frac{1}{m} \sum_{G \in \{G_n^1, \cdots, G_n^m\}} t(G) \tag{3.13}$$



(a) $\lambda = 10$ (b) $\lambda = 100$

Figure 3.4: The numbers of major iteration for $f(S) = -m_1(S) + \lambda w_1(\mathcal{N}(S))$. The red line is the linear interpolation of the worst case points, and the black line is the linear interpolation of the average case points.

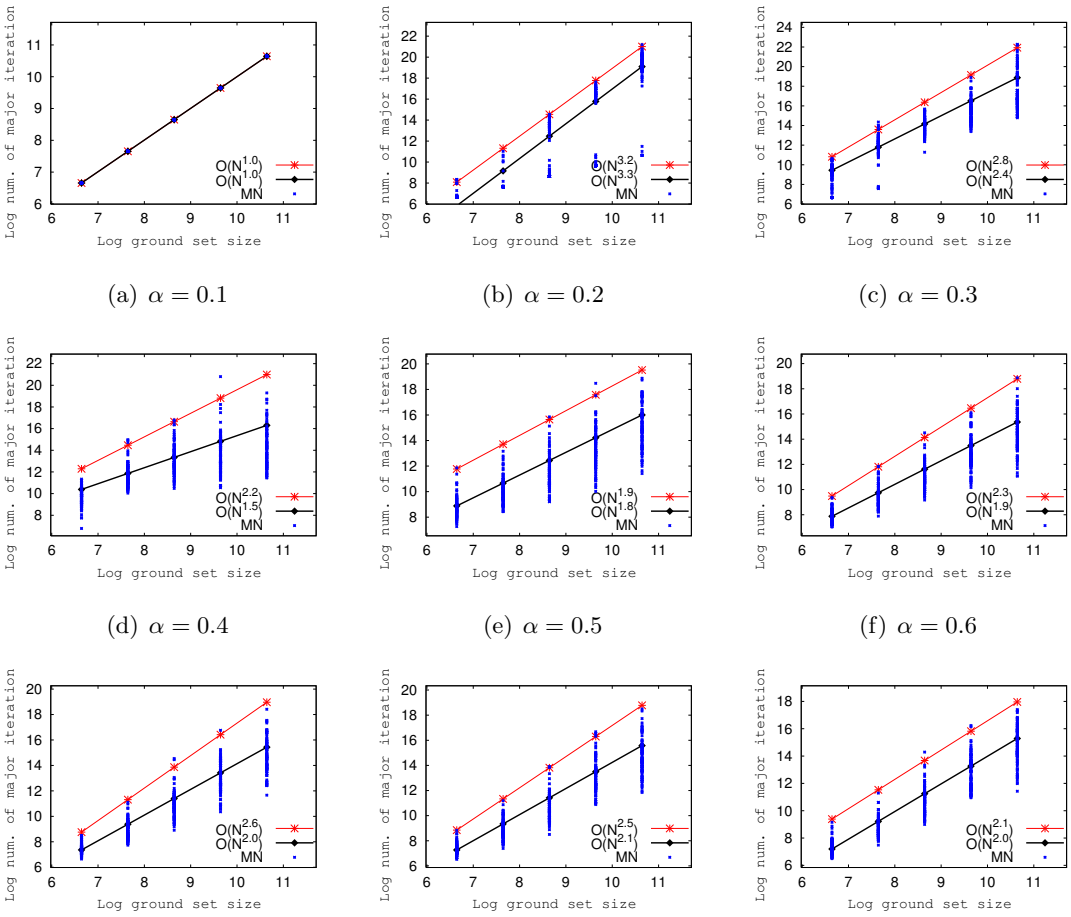
where $t(G)$ is the time of the algorithm spent on instance G, which in this case is the number of major iterations of MN algorithm on input G.

We used bipartite graphs with sizes ranging from 100 and 1600, and for each size, 100 instances were randomly sampled from the large graph. Note that since the sample size is relatively small, the approximate worst-case complexity are not necessary worse than the approximate average-case complexity.

We tested numbers of submodular functions by varying $m(\cdot)$, $w(\cdot)$, λ and α. In particular, we tested

- $f(S) = -m_1(S) + \lambda w_1(\mathcal{N}(S))$, where $\lambda = 10, 100$. The results are shown in Figure 3.4. As we can see, MN algorithm is quite efficient on this type of functions. The number of major iterations grows linearly. This is not surprising as this type of function can be minimized efficiently by converting it to a minimum cut problem.

- $f(S) = -m_1(S) + 100(w_1(\mathcal{N}(S)))^{\alpha}$, where $\alpha = 0.1, 0.2, \cdots, 0.9$. MN algorithm starts to show higher time complexity on this type of functions, as shown in Figure 3.5 The average number of major iterations for functions with different αs is mostly



(a) $\alpha = 0.1$ (b) $\alpha = 0.2$ (c) $\alpha = 0.3$

(d) $\alpha = 0.4$ (e) $\alpha = 0.5$ (f) $\alpha = 0.6$

(g) $\alpha = 0.7$ (h) $\alpha = 0.8$ (i) $\alpha = 0.9$

Figure 3.5: The numbers of major iteration for $f(S) = -m_1(S) + 100 \cdot (w_1(\mathcal{N}(S)))^{\alpha}$. The red lines are the linear interpolations of the worst case points, and the black lines are the linear interpolations of the average case points.

between $O(n^2)$ and $O(n^3)$. In particular, when $\alpha = 0.2$, we see the average-case complexity $O(n^{3.2})$ for the number of major iterations. Note that complexity of each major iteration for this function is $O(n^3)$, giving MN $O(n^{6.2})$ time complexity on this function. This is no better than the time complexity of combinatorial algorithms, both of which are useless for real-problems in NLP (e.g., the task in (Lin & Bilmes, 2010a)). Also note that, when $\alpha = 0.1$, the number of major iterations always equal to

the ground set size; the conjecture is that when we compress the return too much, the function behaves nearly as a modular function (i.e. only the $m(\cdot)$ function matters), making the problem easy to solve.

- $f(S) = -m_2(S)+100(w_2(\mathcal{N}(S)))^{\alpha}$. Complexity results are shown in Figure 3.6. As we can see, when $\alpha = 0.4$, we get average-case complexity about $O(n^{3.1})$ for the number of major iterations, and again gives about $O(n^6)$ time complexity.

3.3 Accelerations for Minimum-norm-point Algorithm

The MN algorithm terminates if

$$x^{\top}x^* > x^{\top}x - \epsilon \max_{j \in S} ||x_j||^2,$$

where $x \in B(f)$ is the current iterate, $x^* \in \operatorname{argmin}_{y \in B(f)} x^{\top}y$, and $x_j, j \in S$ is a set of extreme points of $B(f)$, and $\epsilon \geq 0$ is the "accuracy" or tolerance parameter. In general, we set $\epsilon = 10^{-10}$ to allow some round-off error.

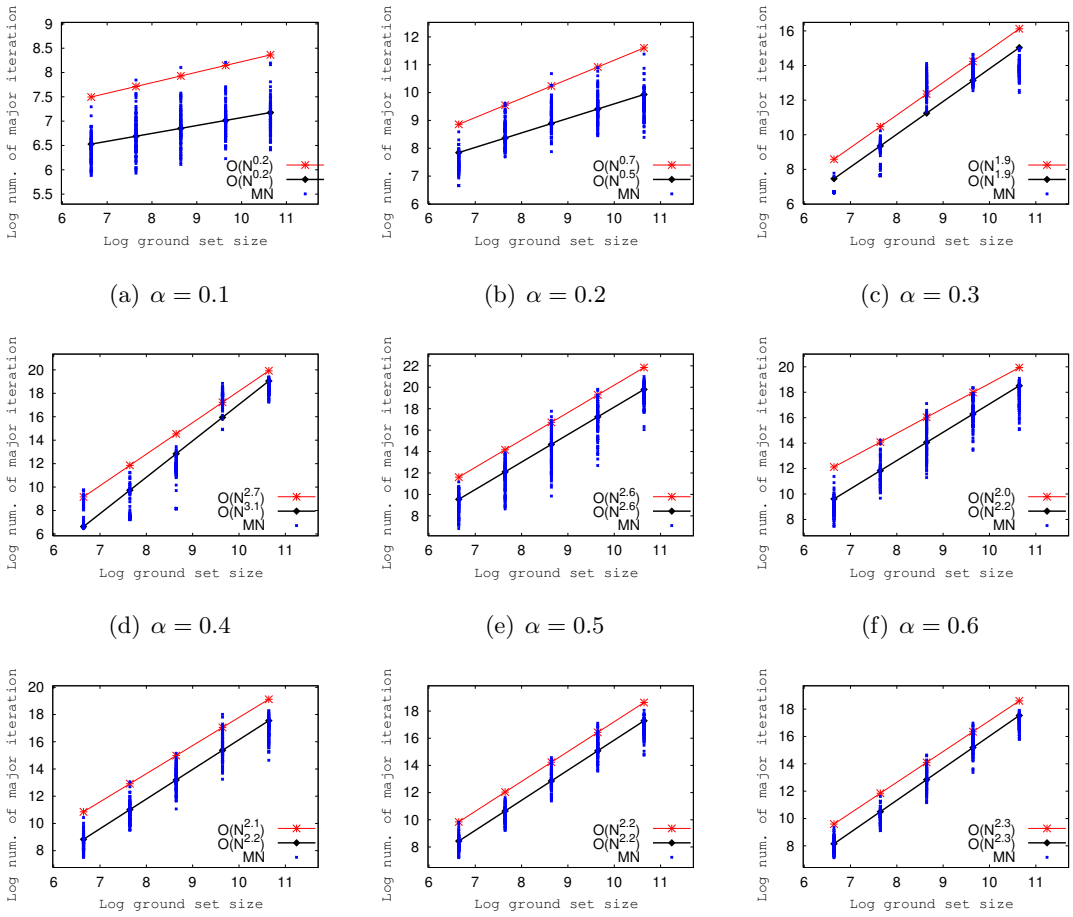
We introduce two stopping criteria that might be triggered before the above termination condition is satisfied.

3.3.1 Early stopping based on subgradient of Lovász extension

Lovász extension is a continuous extension of a set function. In particular, we have $f(S) = \tilde{f}(\chi_S)$ where χ_S is the indicator vector S and $\tilde{f}$ is the Lovász extension of f. Moreover, f is submodular iff $\tilde{f}$ is convex.

Theorem 12. *Let β be a subgradient of $\tilde{f}$ at χ_A, i.e., $\beta \in \partial\tilde{f}(\chi_A)$. Then A is a minimizer of f if*

$$\sum_{i \in E} \max\left\{0, \beta_i(-1)^{1_{i \notin A}}\right\} = 0. \tag{3.14}$$



(a) $\alpha = 0.1$ (b) $\alpha = 0.2$ (c) $\alpha = 0.3$

(d) $\alpha = 0.4$ (e) $\alpha = 0.5$ (f) $\alpha = 0.6$

(g) $\alpha = 0.7$ (h) $\alpha = 0.8$ (i) $\alpha = 0.9$

Figure 3.6: The numbers of major iteration for $f(S) = -m_2(S) + 100 \cdot (w_2(\mathcal{N}(S)))^{\alpha}$. The red lines are the linear interpolations of the worst case points, and the black lines are the linear interpolations of the average case points.

Proof.

$$\begin{aligned} f(A) - f(S^*) &= \tilde{f}(\chi_A) - \tilde{f}(\chi_{S^*}) \\ &\leq \beta^\top (\chi_A - \chi_{S^*}) \\ &\leq \max_{x \in [0,1]^{|E|}} \beta^\top (\chi_A - x) \\ &= \sum_{i \in E} \max\{0, \beta_i (1_{i \in A} - 1_{i \notin A})\} \end{aligned}$$

□

To obtained a subgradient of $\tilde{f}$ at at point x, one could use the maximal chain. I.e., for a permutation $\pi : E \to E$ such that $x_{\pi_1} \geq \cdots \geq x_{\pi_n}$ and define the set $S_k = \{\pi_1, \cdots, \pi_k\}$, we have $\partial \tilde{f}(x) \ni \sum_{k=1}^{n} \delta_{\pi_k}(f(S_k) - f(S_{k-1}))$.

If a function is decomposable, i.e., the function can be represented as sums of truncation potentials (Stobbe & Krause, 2010), gradient of the smoothed Lovász extension can be used as the subgradient. However, evaluating gradient of the smoothed function could become a burden to the main algorithm, especially when the number of truncation potentials is large.

3.3.2 Early stopping based on duality gap

We define the smallest function value difference of a set function as its "resolution" (or reverse resolution):

Definition 8. *The resolution of a set function f is defined as*

$$\gamma_f \triangleq \min_{S,T \subseteq V, f(S) \neq f(T)} |f(S) - f(T)|. \tag{3.15}$$

Denote $x^-(E) = \sum_{e \in E} \min\{x(e), 0\}$, we have the following early stopping theorem.

Theorem 13. *For any $A \subseteq E$ and $x \in B(f)$, A is a minimizer of f (normalized) if*

$$f(A) - x^-(E) < \gamma_f. \tag{3.16}$$

Proof. Based on the min-max theorem of SFM (Fujishige, 2005b):

$$\min_{S \subseteq E} f(S) = \max_{x \in B(f)} x^-(E),$$

therefore, $f(S) \geq x^-(E)$ for any $S \subseteq E, x \in B(f)$. We have

$$f(A) - f(S^*) \leq f(A) - x^-(E). \tag{3.17}$$

If $f(A) - x^-(E) < \gamma_f$, then $f(A) - f(S^*) < \gamma_f$, which implies $f(A) = f(S^*)$. □

In general, it is hard to identify γ_f. For some types of submodular functions, however, a lower bound on γ_f could easily be estimated. For instances, any integral function has

$\gamma_f \geq 1$. Then the early stopping criteria we have is reduced to the early criteria for integral functions introduced in (Fujishige & Isotani, 2011).

For functions of the form

$$f(S) = m(S) + \lambda g(w(\mathcal{N}(S))),$$

where $\lambda \geq 0,\ S \subseteq E,\ \mathcal{N} : 2^E \to 2^F,\ m : 2^V \to \mathbb{Z},\ w : 2^F \to \mathbb{Z}$, and $g : \mathbb{R} \to \mathbb{R}$ is a nondecreasing concave function. We have $\gamma_f \geq |1 - \lambda|g(w(F)) - g(w(F) - 1)||$.

3.3.3 Experiments

Lastly, we tested MN algorithm with the acceleration methods introduced in Section 3.3, where the duality gap based criteria achieves phenomenal speedup for a type of submodular functions.

Accelerations

We evaluated the acceleration methods on submodular functions with bipartite graphs.

For the subgradient based acceleration, we observed several instances where the number of function oracle calls reduced from 21 to 2. In most cases, however, the early stopping do not triggered before the MN algorithm terminates based on its original termination condition.

On the other hand, duality-gap based acceleration reduces the running of MN significantly for all type of functions (f_1, f_2, f_3, f_4 and f_5). Figure 3.7 illustrates the significant reduction. In particular, MN without acceleration finishes in about 4380 seconds while MN with duality-gap acceleration finishes in 0.06 seconds, MN without acceleration calls function oracle 249785250 times and MN with duality-gap acceleration only calls function oracle 77220 times, resulting about $75,000$ times speedup in runtime and $3,200$ times speedup in oracle calls.

3.3.4 Discussion and Conclusion

Wolfe's algorithm is closely related to the simplex algorithm for linear programming. As it has been shown that the lower bound of the time complexity of simplex algorithm has high

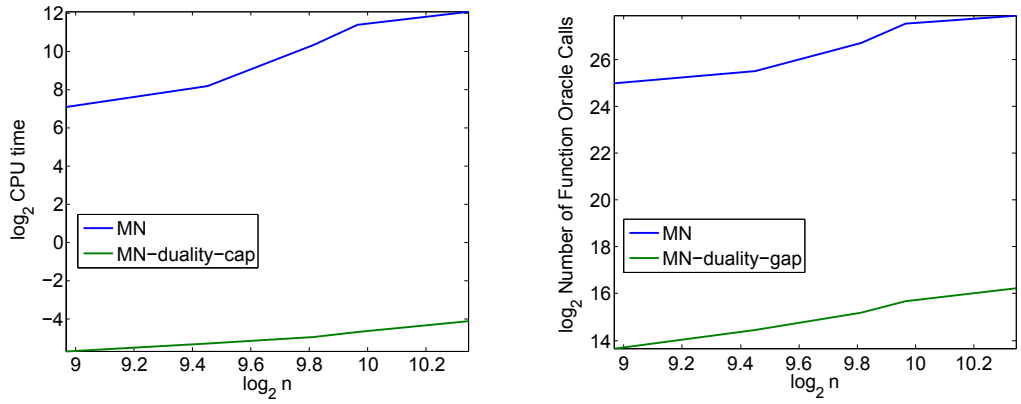


Figure 3.7: Comparison of MN algorithm with and without duality gap acceleration. The random sampled graphs with left node set sizes $500, 700, 900, 1000$ and 1300, and the objective function is $f_2(S) = m(S) + 100\sqrt{w(\mathcal{N}(S))}$.

exponents, our conjecture is that MN might also have a very bad theoretical lower bound, which is actually evidenced by our example showing that MN's performance could be as bad as $O(n^7)$ on running time and $O(n^5)$ on oracle calls.

Most applications of SFM arise in practice have some special structure that can be taken advantage of, resulting in much faster special-purpose algorithm. For instances, when submodular is symmetric, Queyranne's $O(n^3)$ algorithm can be used. When submodular functions are graph-representable (Zivný & Jeavons, 2008), fast min-cut/max-flow can be applied. For certain functions of the form $f(S) = m(S) + g(\mathcal{N}(S))$, exact solutions can be obtained by leveraging parametric flow algorithms (Jegelka *et al.* , 2011).

When one has to solve SFM with general submodular functions, an alternative is to use fast approximation algorithms (e.g., (Jegelka *et al.* , 2011)). To get the exact solution, however, MN algorithm is still the most practical choice to date. When using MN algorithm for large size problem, we suggest designing submodular objective functions with lower resolution (higher γ_f), in which case the early stopping heuristics based on duality gap could potentially drastically improve the running time performance of MN algorithm.

3.4 Budgeted Submodular Maximization

In this section, we consider the problem of budgeted submodular maximization.

Problem 4 (Budgeted Submodular Maximization)**.**

$$\max_{S \subseteq V} f(S), \ \textit{subject to:} \sum_{i \in S} c_i \leq b. \tag{3.18}$$

where $f : 2^V \rightarrow \mathbb{R}$ is monotone submodular, c_i is the cost for element i, and b is the budget.

This problem naturally arises in some NLP tasks such as document summarization. See Chapter 4 about the applications of this problem.

Although algorithm with $1 - \frac{1}{e}$ approximation factor is available for this problem (Sviridenko, 2004), the partial enumeration makes it impractical for NLP scaled problems. Having the scale and scenario of the NLP application in mind, we modified the algorithm in (Sviridenko, 2004) with two modifications for practical considerations:

1. Replacing the partial enumeration with only enumerating singleton solutions, which greatly reduces the computation burden introduced by partial enumeration while making the algorithm's performance still theoretically bounded.

2. A scaling parameter on the cost, addressing the uncalibrated issue of function gain and cost, which has later been shown that affects performance significantly in practise.

The algorithm is illustrated in Algorithm 4, which sequentially finds unit k with the largest ratio of objective function gain to scaled cost, i.e., $(f(G \cup \{k\}) - f(G))/c_k^r$, where $r > 0$ is the scaling factor. If adding k increases the objective function value while not violating the budget constraint, it is then selected and otherwise bypassed. After the sequential selection, set G is compared to the within-budget singleton with the largest objective value, and the

larger of the two becomes the final output.

Algorithm 4: Modified greedy algorithm

1: $G \leftarrow \emptyset$
2: $U \leftarrow V$
3: **while** $U \neq \emptyset$ **do**
4: $\quad k \leftarrow \arg\max_{\ell \in U} \frac{f(G \cup \{\ell\}) - f(G)}{(c_\ell)^r}$
5: $\quad G \leftarrow G \cup \{k\}$ **if** $\sum_{i \in G} c_i + c_k \leq b$ **and** $f(G \cup \{k\}) - f(G) > 0$
6: $\quad U \leftarrow U \setminus \{k\}$
7: **end while**
8: $v^* \leftarrow \arg\max_{v \in V, c_v \leq b} f(\{v\})$
9: return $G_f = \arg\max_{S \in \{\{v^*\}, G\}} f(S)$

The essential aspect of a greedy algorithm is the design of the greedy heuristic. As discussed in (Khuller *et al.* , 1999), a heuristic that greedily selects the k that maximizes $(f(G \cup \{k\}) - f(G))/c_k$ has an unbounded approximation factor. For example, let $V = \{x, y\}$, $f(\{x\}) = 1, f(\{y\}) = p$, $c_x = 1, c_y = p + 1$, and $b = p + 1$. The solution obtained by the greedy heuristic is $\{x\}$ with objective function value 1, while the true optimal objective function value is p. The approximation factor for this example is then p and therefore unbounded.

This issue can be addressed by the modifications we mentioned before. The first modification is the final step (line 8 and 9) in Algorithm 4 where set G and singletons are compared. This step ensures that we could obtain a constant approximation factor for $r = 1$.

The second modification is that we introduce a scaling factor r to adjust the scale of the cost. Suppose, in the above example, we scale the cost as $c_x = 1^r, c_y = (p + 1)^r$, then selecting x or y depends also on the scale r, and we might get the optimal solution using a appropriate r. Indeed, the objective function values and the costs might be uncalibrated since they might measure different units. E.g., in document summarization, it is hard to say if selecting a sentence of 15 words with an objective function gain of 2 is better than selecting sentence of 10 words with gain of 1. Scaling can potentially alleviate this mismatch (i.e., we can adjust r on development set). Interestingly, our theoretical analysis of the performance

guarantee of the algorithm also gives us guidance about how to scale the cost for a particular problem (see Section 3.4.1).

3.4.1 Analysis of performance guarantee

Although Algorithm 4 is essentially a simple greedy strategy, we show that it solves Problem 4 globally and near-optimally, by exploiting the structure of submodularity. As far as we know, this is a new result for submodular optimization, not previously stated or published before.

Theorem 14 (Lin & Bilmes (2010b))**.** *For normalized monotone submodular function $f(\cdot)$, Algorithm 4 with $r = 1$ has a constant approximation factor as follows:*

$$f(G_f) \geq \left(1 - e^{-\frac{1}{2}}\right) f(S^*), \tag{3.19}$$

where S^* is an optimal solution.

We analyze the performance guarantee of Algorithm 4. We use the following notation: S^* is the optimal solution; G_f is the final solution obtained by Algorithm 4; G is the solution obtained by the greedy heuristic (line 1 to 7 in Algorithm 4); v_i is the ith unit added to G, $i = 1, \ldots, |G|$; G_i is the set obtained by greedy algorithm after adding v_i (i.e., $G_i = \cup_{k=1}^{i}\{v_k\}$, for $i = 1, \ldots, |G|$, with $G_0 = \emptyset$ and $G_{|G|} = G$); $f(\cdot) : 2^V \rightarrow \mathbb{R}$ is a monotone submodular function; and $\rho_k(S)$ is the gain of adding k to S, i.e., $f(S \cup \{k\}) - f(S)$.

Lemma 2. *$\forall X, Y \subseteq V$,*

$$f(X) \leq f(Y) + \sum_{k \in X \setminus Y} \rho_k(Y). \tag{3.20}$$

Proof. See (Nemhauser *et al.* , 1978) □

Lemma 3. *For $i = 1, \ldots, |G|$, when $0 \leq r \leq 1$,*

$$f(S^*) - f(G_{i-1}) \leq \left(\frac{b}{c_{v_i}}\right)^r |S^*|^{1-r}(f(G_i) - f(G_{i-1})), \tag{3.21}$$

and when $r \geq 1$,

$$f(S^*) - f(G_{i-1}) \leq \left(\frac{b}{c_{v_i}}\right)^r (f(G_i) - f(G_{i-1})) \tag{3.22}$$

Proof. Based on line 4 of Algorithm 4, we have

$$\forall u \in S^* \setminus G_{i-1}, \frac{\rho_u(G_{i-1})}{c_u^r} \leq \frac{\rho_{v_i}(G_{i-1})}{c_{v_i}^r}.$$

Thus when $0 \leq r \leq 1$,

$$\begin{aligned}
\sum_{u \in S^* \setminus G_{i-1}} \rho_u(G_{i-1}) &\leq \frac{\rho_{v_i}(G_{i-1})}{c_{v_i}^r} \sum_{u \in S^* \setminus G_{i-1}} c_u^r \\
&\leq \frac{\rho_{v_i}(G_{i-1})}{c_{v_i}^r} |S^* \setminus G_{i-1}| \left(\frac{\sum_{u \in S^* \setminus G_{i-1}} c_u}{|S^* \setminus G_{i-1}|} \right)^r \\
&\leq \frac{\rho_{v_i}(G_{i-1})}{c_{v_i}^r} |S^*|^{1-r} \left(\sum_{u \in S^* \setminus G_{i-1}} c_u \right)^r \\
&\leq \frac{\rho_{v_i}(G_{i-1})}{c_{v_i}^r} |S^*|^{1-r} b^r,
\end{aligned}$$

where the second inequality is due to the concavity of $g(x) = x^r, x > 0, 0 \leq r \leq 1$. The last inequality uses the fact that $\sum_{u \in S^*} c_u \leq b$. Similarly, when $r \geq 1$,

$$\begin{aligned}
\sum_{u \in S^* \setminus G_{i-1}} \rho_u(G_{i-1}) &\leq \frac{\rho_{v_i}(G_{i-1})}{c_{v_i}^r} \sum_{u \in S^* \setminus G_{i-1}} c_u^r \\
&\leq \frac{\rho_{v_i}(G_{i-1})}{c_{v_i}^r} \left(\sum_{u \in S^* \setminus G_{i-1}} c_u \right)^r \\
&\leq \frac{\rho_{v_i}(G_{i-1})}{c_{v_i}^r} b^r.
\end{aligned}$$

Applying Lemma 2, i.e., let $X = S^*$ and $Y = G_{i-1}$, the lemma immediately follows. □

Lemma 4 (Lin & Bilmes (2010b))**.** *With normalized monotone submodular $f(\cdot)$, for $i = 1, \ldots, |G|$, let v_i be the ith unit added into G and G_i is the set after adding v_i. When $0 \leq r \leq 1$,*

$$f(G_i) \geq \left(1 - \prod_{k=1}^{i} \left(1 - \frac{c_{v_k}^r}{b^r |S^*|^{1-r}} \right) \right) f(S^*) \tag{3.23}$$

$$\geq \left(1 - \prod_{k=1}^{i} \left(1 - \frac{c_{v_k}^r}{b^r |V|^{1-r}} \right) \right) f(S^*) \tag{3.24}$$

and when $r \geq 1$,

$$f(G_i) \geq \left(1 - \prod_{k=1}^{i} \left(1 - \left(\frac{c_{v_k}}{b} \right)^r \right) \right) f(S^*). \tag{3.25}$$

Proof. Obviously, the lemma is true when $i = 1$ by applying Lemma 3.

Assume that the theorem is true for $i-1, 2 \leq i \leq |G|$, we show that it also holds for i. When $0 \leq r \leq 1$,

$$\begin{aligned}
f(G_i) &= f(G_{i-1}) + (f(G_i) - f(G_{i-1})) \\
&\geq f(G_{i-1}) + \frac{c_{v_i}^r}{b^r |S^*|^{1-r}} (f(S^*) - f(G_{i-1})) \\
&= \left(1 - \frac{c_{v_i}^r}{b^r |S^*|^{1-r}}\right) f(G_{i-1}) + \frac{c_{v_i}^r}{b^r |S^*|^{1-r}} f(S^*) \\
&\geq \left(1 - \frac{c_{v_i}^r}{b^r |S^*|^{1-r}}\right) \left(1 - \prod_{k=1}^{i-1} \left(1 - \frac{c_{v_k}^r}{b^r |S^*|^{1-r}}\right)\right) f(S^*) + \frac{c_{v_i}^r}{b^r |S^*|^{1-r}} f(S^*) \\
&= \left(1 - \prod_{k=1}^{i} \left(1 - \frac{c_{v_k}^r}{b^r |S^*|^{1-r}}\right)\right) f(S^*).
\end{aligned}$$

The case when $r \geq 1$ can be proven similarly. □

Now we are ready to prove Theorem 14.

Proof. Consider the following two cases:

Case 1: $\exists v \in V$ such that $f(\{v\}) > \frac{1}{2} f(S^*)$. Then it is guaranteed that $f(G_f) \geq f(\{v\})) > \frac{1}{2} f(S^*)$ due line 9 of Algorithm 4.

Case 2: $\forall v \in V$, we have $f(\{v\}) \leq \frac{1}{2} f(S^*)$. We consider the following two sub-cases, namely Case 2.1 and Case 2.2:

Case 2.1: If $\sum_{v \in G} c_v \leq \frac{1}{2} b$, then we know that $\forall v \notin G, c_v > \frac{1}{2} b$ since otherwise we can add a $v \notin G$ into G to increase the objective function value without violating the budget constraint. This implies that there is at most one ground set element in $S^* \setminus G$ since otherwise we will have $\sum_{v \in S^*} c_v > b$. By assumption, we have $f(S^* \setminus G) \leq \frac{1}{2} f(S^*)$. Submodularity of $f(\cdot)$ gives us:

$$f(S^* \setminus G) + f(S^* \cap G) \geq f(S^*),$$

which implies $f(S^* \cap G) \geq \frac{1}{2} f(S^*)$. Thus we have

$$f(G_f) \geq f(G) \geq f(S^* \cap G) \geq \frac{1}{2} f(S^*),$$

where the second inequality follows from monotonicity.

Case 2.2: If $\sum_{v \in G} c_v > \frac{1}{2}b$, for $0 \leq r \leq 1$, using Lemma 4, we have

$$\begin{aligned} f(G) &\geq \left(1 - \prod_{k=1}^{|G|} \left(1 - \frac{c_{v_k}^r}{b^r |S^*|^{1-r}}\right)\right) f(S^*) \\ &\geq \left(1 - \prod_{k=1}^{|G|} \left(1 - \frac{c_{v_k}^r |S^*|^{r-1}}{2^r \left(\sum_{k=1}^{|G|} c_{v_k}\right)^r}\right)\right) f(S^*) \\ &\geq \left(1 - \left(1 - \frac{|S^*|^{r-1}}{2^r |G|^r}\right)^{|G|}\right) f(S^*) \\ &\geq \left(1 - e^{-\frac{1}{2}\left(\frac{|S^*|}{2|G|}\right)^{r-1}}\right) f(S^*) \end{aligned}$$

where the third inequality uses the fact (provable using Lagrange multipliers) that for $a_1, \ldots, a_n \in \mathbb{R}^+$ such that $\sum_{i=1}^n a_i = \alpha$, function

$$1 - \prod_{i=1}^{n} \left(1 - \frac{\beta a_i^r}{\alpha^r}\right)$$

achieves its minimum of $1 - (1 - \beta/n^r)^n$ when $a_1 = \cdots = a_n = \alpha/n$ for $\alpha, \beta > 0$. The last inequality follows from $e^{-x} \geq 1 - x$.

In all cases, we have

$$f(G_f) \geq \min\left\{\frac{1}{2}, 1 - e^{-\frac{1}{2}\left(\frac{|S^*|}{2|G|}\right)^{r-1}}\right\} f(S^*)$$

In particular, when $r = 1$, we obtain the constant approximation factor, i.e.

$$f(G_f) \geq \left(1 - e^{-\frac{1}{2}}\right) f(S^*)$$

□

Note that an α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for *all* instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution. So Theorem 14 basically states that the solution found by Algorithm 4 can be at least as good as $(1 - 1/\sqrt{e})f(S^*) \approx 0.39 f(S^*)$ even in the worst case. A constant approximation bound is good since it is true for all instances of the problem, and we always know how good the algorithm is guaranteed to be

without any extra computation. For $r \neq 1$, we resort to instance-dependent bound where the approximation can be easily computed per problem instance.

Lemma 4 gives bounds for a specific instance of the problem. Eqn. (3.23) requires the size $|S^*|$, which is unknown, requiring us to estimate an upper bound of the cardinality of the optimal set S^*. Obviously, $|S^*| \leq |V|$, giving us Eqn. (3.24). A tighter upper bound is obtained, however, by sorting the costs. That is, let $c_{[1]}, c_{[2]}, \ldots, c_{[|V|]}$ be the sorted sequence of costs in nondecreasing order, giving $|S^*| < m$ where $\sum_{k=1}^{m-1} c_{[i]} \leq b$ and $\sum_{k=1}^{m} c_{[i]} > b$. In this case, the computation cost for the bound estimation is $O(|V| \log |V|)$, which is quite feasible.

3.4.2 Related work

Algorithms for maximizing submodular function under budget constraint (Problem. 4) have been studied before. Krause & Guestrin (2005a) generalized the work by Khuller *et al.* (1999) on budgeted maximum cover problem to the submodular framework, and showed a $\frac{1}{2}(1 - 1/e)$-approximation algorithm. The algorithm in (Krause & Guestrin, 2005a) and (Khuller *et al.* , 1999) is actually a special case of Algorithm 4 when $r = 1$, and Theorem 14 gives a better bound (i.e., $(1 - 1/\sqrt{e}) > \frac{1}{2}(1 - 1/e)$) in this case. There is also a greedy algorithm with partial enumerations (Sviridenko, 2004; Krause & Guestrin, 2005a) factor $(1 - 1/e)$. This algorithm, however, is too computationally expensive and thus not practical for real world applications (the computation cost is $O(|V|^5)$ in general). When each unit has identical cost, the budget constraint reduces to cardinality constraint where a greedy algorithm is known to be a $(1 - 1/e)$-approximation algorithm (Nemhauser *et al.* , 1978) which is the best that can be achieved in polynomial time (Feige, 1998b) if P $\neq$ NP.

3.5 On Submodular Knapsack

In this section, we consider *submodular knapsack* problem. In particular, modular maximization subject to submodular knapsack constraint (MMSKC) of the following form is considered:

Problem 5 (Modular Maximization subject to Submodular Knapsack Constraint (MM-

SKC)).

$$\max_{S \subseteq V} m(S), \ \textit{subject to:} \ f(S) \leq b \tag{3.26}$$

where $m \in \mathbb{R}^n_+$ *is a modular function,* $|V| = n$, $b \in \mathbb{R}$, *and* $f : 2^V \to \mathbb{R}$ *is a* monotone submodular *function.*

The motivation for this function come from corpus subset selection (Lin & Bilmes, 2010a, 2011b). I.e., m measures the amount of data that one selects (e.g., the number of speech utterances that are selected, where each utterance has a benefit, measured by m, such as speech length, number of tokens, etc.). We have a vocabulary restriction upper bound given by b and this is measured by monotone nondecreasing submodular function f. I.e., we want the vocabulary size to be no more than b. See Chapter 7 for more about this application.

Problems with similar forms to MMSKC have been studied in the literature (in the below, we use f and g to denote submodular functions, m for modular function). For instance, in the traditional submodular set cover (SSC) problem, we solve

Problem 6 (Submodular Set Cover (SSC))**.**

$$\min_{S \subseteq V} m(S), \ \textit{subject to:} \ f(S) = f(V),$$

For SSC, greedy algorithm is guaranteed to find a solution $\hat{S}$ such that $f(\hat{S}) \leq (1 + \ln n) f(S^*)$ (Wolsey, 1982) where S^* is the optimal solution. Iwata & Nagano (2009) study a problem that generalizes the objective function is traditional SSC problem to a submodular function, but limits the constraint to be set cover rather than general submodular set cover, i.e.,

Problem 7 (Submodular Minimization with Covering constraint)**.**

$$\min_{S \subseteq V} f(S), \ \textit{subject to:} \ \Gamma(S) = \Gamma(V),$$

where $\Gamma : 2^V \to 2^F$ *is the covering function and* F *is the set of elements to be covered.*

Besides Lovász extension, techniques used in Iwata & Nagano (2009) are based on a key fact that the cover constraint can be written as linear constraints over the indicator

variables. E.g., let $x(i) \in \{0,1\}$ be the indicator that whether $i \in S \subseteq V$, and $C(j)$ be the set of elements in V that covers $j \in F$, i.e. $C(j) \triangleq \{i \in V | j \text{ covered by } i\}$. Then $\Gamma(S) = \Gamma(V)$ is equivalent to

$$\sum_{i \in C(j)} x(i) \geq 1, \forall j.$$

Such representation, however, does not seem to be available to an arbitrary submodular function.

Svitkina & Fleischer (2008) study a problem, namely submodular minimization with cardinality lower bound (SML), with the following form:

Problem 8 (Submodular Minimization with Cardinality Lower Bound (SML))**.**

$$\min_{S \subseteq V} f(S), \text{ subject to: } m(S) \geq c.$$

where $f : 2^V \to \mathbb{R}$ *is monotone submodular.*

Interestingly, the MMSKC problem of interest in this section is reducible to $|V|$ SML problems. The reduction from MMSKC to SML is as follows: denote a solution set of min $f(S)$ s.t. $m(S) \geq c$ as X_c^*, and a solution of MMSKC max $m(S)$ s.t. $f(S) \leq b$ as Y^*. Then if $f(X_c^*) \leq b$, we have $m(Y^*) \geq c$. On the other hand, if $f(X_c^*) > b$, we know that for all S such that $m(S) \geq c$ the minimum possible value of $f(S)$ is larger than b, therefore $m(Y^*) < c$. In other words, SML can be used as a certificate problem of MMSKC. By solving SML and examining $f(X_c^*)$, we can bisect the possible range of $m(Y^*)$. Using binary search, a solution of MMSKC can be found. When $m(S)$ is integral, only $|V|$ calls of SML are required.

Now consider the case when only an approximation algorithm is available for SML, say we have an approximation algorithm that finds a solution X_c such that $f(X_c) \leq \sigma f(X_c^*)$ and $m(X_c) \geq \rho c$ where $\sigma \geq 1, \rho \leq 1$. Checking the value of $f(X_c)$, if

- $f(X_c) > \sigma b$: then $f(X_c^*) > b$ which implies that $m(Y^*) < c$;
- $f(X_c) \leq b$: X_c is a feasible solution for MMSKC, therefore $m(Y^*) \geq m(X_c) \geq \rho c$;

- $b < f(X_c) \leq \sigma b$: we have a solution such that $m(X_c) \geq \rho c$ and $f(X_c) \leq \sigma b$.

Using this observation, we can have an approximation algorithm for MMSKC, as shown in Algorithm 5. It is an $(\epsilon\rho, \sigma)$-approximation algorithm, which requires $O(\log_{1-\epsilon} N)$ calls of the (σ, ρ)-approximation algorithm for SML.

Algorithm 5: An algorithm for MMSKC

input : MMSKC problem with groundset V, budget b, $0 < \epsilon < 1$.

begin $c_l \leftarrow 0, c_u \leftarrow m(V)$;

while 1 **do**

$c \leftarrow c_u - \epsilon(c_u - c_l)$;

Obtain (σ, ρ)-approximate solution X_c for SML: min $f(S)$ s.t. $m(S) \geq c$;

if $f(X_c) \leq b$ **then**
$c_l \leftarrow \rho c$

else if $f(X_c) > \sigma b$ **then**
$c_u \leftarrow c$

else Return X_c

Theorem 15. *Algorithm 5 is an $(\epsilon\rho, \sigma)$-approximation algorithm for MMSKC, which requires $O(\log_{1-\epsilon} N)$ calls of the (σ, ρ)-approximation algorithm for SML.*

Proof. Let Y^* be the optimal solution for MMSKC. Algorithm 5 guarantees that $c_l \leq m(Y^*) \leq c_u$. Moreover, we have $c \geq \epsilon c_u$. When the algorithm terminates, we have $f(X_c) \leq \sigma b$ and $m(X_c) \geq \rho c \geq \rho\epsilon c_u \geq \epsilon\rho m(Y^*)$. □

We also have the following hardness result:

Theorem 16. *There is no (ρ, σ) bicriteria approximation algorithm for MMSKC problem, even with monotone submodular function, for any ρ and σ with $\frac{\rho}{\sigma} = o\left(\sqrt{\frac{\ln n}{n}}\right)$.*

In other words, there is no polynomial time algorithm that can find a solution S such that $m(S) \geq \rho m(X^*)$ and $f(S) \leq \sigma b$ such that $\frac{\rho}{\sigma} = o\left(\sqrt{\frac{\ln n}{n}}\right)$, where X^* is the optimal solution to MMSKC.

To proof the theorem, we use the following lemma

Lemma 5 (Lemma 4.1 in Svitkina & Fleischer, 2008)**.** *Let R be a random subset of V of size $\alpha = \frac{x\sqrt{n}}{5}, \beta = \frac{x^2}{5}$ and x be any parameter satisfying $x^2 = \omega(\ln n)$. Consider two submodular functions*

$$f_1(S) = \min(|S|, \alpha)$$
$$f_2(S) = \min(\beta + |S \cap \bar{R}|, |S|, \alpha),$$

then any algorithms that makes polynomial number of oracle queries has probability $n^{-\omega(1)}$ of distinguishing the functions f_1 and f_2.

Proof of the hardness theorem.

Proof. Assume that a bicriteria algorithm exists such that it finds a solution S for MMSKC with $c = \mathbf{1}$ and $\frac{\rho}{\sigma} = o\left(\sqrt{\frac{\ln n}{n}}\right), \rho \leq 1, \sigma \geq 1$, i.e., $|S| \geq \rho \cdot \max_{X \subseteq V, f(X) \leq b} |X|$ and $f(S) \leq \sigma b$. Let $x = \frac{\rho\sqrt{n}}{2\sigma}$. Obviously, $x^2 = \omega(\ln n)$, and we have $\rho\alpha = \rho\frac{x\sqrt{n}}{5} = \frac{2\sigma x}{\sqrt{n}}\frac{x\sqrt{n}}{5} = 2\sigma\beta$.

Consider the output of this algorithm when given f_2 as input and $b = \beta$. The optimal solution in this case is the set R (since $\alpha > \beta$). By the assumption, the algorithm finds an approximation solution S with $|S| \geq \rho|R| = \rho\alpha$ and $f_2(S) \leq \sigma b = \sigma\beta$.

Now let the input function be f_1. Consider two cases $f_1(S) \leq \beta$ and $f_1(S) > \beta$. In the first case, $f_1(S) \leq \beta$ implies that $|S| \leq \beta$, which contradicts that $|S| \geq \rho\alpha = 2\sigma\beta \geq 2\beta$. In the second case, since $\beta < f_1(S) \leq \sigma\beta = \frac{\rho\alpha}{2} \leq \frac{|S|}{2}$, we have $f_1(S) = \min(|S|, \alpha) = \alpha \leq \frac{|S|}{2}$ and therefore $|S| \geq 2\alpha > \alpha$, which contradicts that $|S| \leq \alpha$. In sum, when the input functions f_1, the solution found by the algorithm cannot be S, which implies that it produces a different answer, thus distinguishing f_1 and f_2.

□

Chapter 4

SUBMODULAR SUMMARIZATION: EXTRACTING VALUE FROM CHAOS

We live in an era of "Big Data". As more and more people join the digital tribe — increasingly through internet-enabled mobile devices — the world's digital output is increasing at a rate that even eclipses Moore's Law. Based on the findings of the 2011 IDC Digital Universe Study [1], the amount of information created and replicated has surpassed 1.8 zettabytes (1.8 trillion gigabytes) — growing by a factor of 9 in just five years.

In many fields, it is important to stay abreast of the most recent developments in the world, but this is becoming an ever more arduous tasks because of the large quantity of information. However, there is much redundancy. On the one hand, this makes it difficult to discover the main point or the consensus opinion, as much of it might be irrelevant to a particular information query. On the other hand, this problem presents interesting research challenges in natural language processing, one of which is the problem of document summarization.

In this chapter, we will see how submodularity naturally arises in the task of summarization. We cast the summarization problem as a submodular optimization problem. This has a number of critical benefits. On the one hand, there exists a simple greedy algorithm for monotone submodular function maximization where the summary solution obtained (say $\hat{S}$) is guaranteed to be almost as good as the best possible solution (say S_{opt}) according to an objective f. More precisely, the greedy algorithm is a constant factor approximation to the cardinality constrained version of the problem, so that $f(\hat{S}) \geq (1-1/e)f(S_{\text{opt}}) \approx 0.632 f(S_{\text{opt}})$ (see Chapter 2 for details). This is particularly attractive since the quality of the solution does not depend on the size of the problem, so even very large size problems do well. It is also important to note that this is a worst case bound, and in most cases the quality of the

[1] http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf

solution obtained will be much better than this bound suggests.

Of course, none of this is useful if the objective function f is inappropriate for the summarization task. In this chapter, we will see that monotone nondecreasing submodular functions f are an ideal class of functions to investigate for summarization. We show that the widely used automatic summarization evaluation metric, ROUGE (Lin, 2004), and the score function in Pyramid method that used in recent TAC evaluations [2], are both monotone submodular. We moreover show, in fact, that many well-established methods for summarization (Carbonell & Goldstein, 1998; Filatova & Hatzivassiloglou, 2004; McDonald, 2007; Takamura & Okumura, 2009; Riedhammer *et al.* , 2010; Shen & Li, 2010) correspond to submodular function optimization, a property not explicitly mentioned in these publications (see Section 4.6). We take this fact, however, as testament to the value of submodular functions for summarization: if summarization algorithms are repeatedly developed that, by chance, happen to be an instance of a submodular function optimization, this suggests that submodular functions are a natural fit.

This chapter presents our explorations on three summarization tasks, namely speech summarization, training data summarization, and multi-document summarization. The rest of this chapter is organized as follows. In Section 4.1 we describe our submodular summarization approach in general, followed by results in the task of speech summarization in Section 4.2 where we show that our approach outperforms state-of-art methods including an integer linear programming based approach. In Section 4.3, we show how we can apply our approach to select a subset of un-transcribed data to transcribe such that model trained on these data can perform better than those trained on data by uncertainty sampling. We show some interesting results on query-independent and query-dependent (query-focused) document summarization in Section 4.4 and Section 4.5 respectively. In Section 4.6, we show that many standard methods for summarization are, in fact, already performing submodular function optimization. This motivates us to consider the problem of "how to design a submodular function" for the summarization task, and in Section 4.7, we demonstrate a class of powerful submodular objective for submodular document summarization that extends

[2] http://www.nist.gov/tac/

beyond any previous work. By carefully crafting a class of submodular functions we feel are ideal for extractive summarization tasks, both generic and query-focused, we show better than existing state-of-the-art performance on a number of standard summarization evaluation tasks, namely DUC-04 through to DUC-07. In particular, we show as far as we know better than any previous known ROUGE results for DUC-04 through DUC-06, and the better than previous known precision results for DUC-07, and the best recall DUC-07 results among those that do not use a web search engine for query expansion. Note that results reported in Section 4.7 are further surpassed by the results obtained from the submodular mixture paradigm introduced in Chapter 5, where we achieve best results ever reported on all DUC evaluation tasks (2004 to 2007). The objective functions introduced in Section 4.7 of this chapter, however, server as important components in the submodular mixture, as we will see in Chapter 5.

4.1 Approach

We are given a set of objects $V = \{v_1, \ldots, v_n\}$ and a function $f : 2^V \rightarrow \mathbb{R}$ that returns a real value for any subset $S \subseteq V$. We are interested in finding the subset of bounded size $|S| \leq k$ that maximizes the function, e.g., $\text{argmax}_{S \subseteq V} f(S)$. In general, this operation is hopelessly intractable, an unfortunate fact since the optimization coincides with many important applications. For example, f might correspond to the value or coverage of a set of sensor locations in an environment, and the goal is to find the best locations for a fixed number of sensors (Krause *et al.* , 2008).

If the function f is monotone submodular then the maximization is still NP complete, but it was shown in (Nemhauser *et al.* , 1978) that a greedy algorithm finds an approximate solution guaranteed to be within $\frac{e-1}{e} \sim 0.63$ of the optimal solution, as mentioned in Chapter 2. A version of this algorithm (Minoux, 1978), moreover, scales to very large data sets. In Section 4.2 and Section 4.3, we apply the greedy algorithm to speech summarization and training data summarization task.

In many other applications, a cardinality constraint does not precisely model the budget requirement. For example, in standard document summarization tasks (e.g., DUC evaluations), the summary is usually required to be length-limited. Therefore, constraints on S can

naturally be modeled as *knapsack constraints*: $\sum_{i \in S} c_i \leq b$, where c_i is the non-negative cost of selecting unit i (e.g., the number of words in the sentence) and b is our *budget*. If we use a set function $f : 2^V \rightarrow \mathbb{R}$ to measure the quality of the summary set S, the summarization problem can then be formalized as the following combinatorial optimization problem:

Problem 9. *Find*

$$S^* \in \operatorname*{argmax}_{S \subseteq V} f(S) \ \textit{subject to:} \ \sum_{i \in S} c_i \leq b.$$

Since this is a generalization of the cardinality constraint (where $c_i = 1, \forall i$), this also constitutes a (well-known) NP-hard problem. In this case as well, however, a modified greedy algorithm with partial enumeration can solve Problem 9 near-optimally with $(1 - 1/e)$-approximation factor if f is monotone submodular (Sviridenko, 2004). The partial enumeration, however, is too computationally expensive for real world applications. In particular, it requires enumerating all solutions with sizes no greater than 3, which gives us, for example, an $O(n^5)$ complexity in our summarization application with a graph-based objective function (Eqn 4.5). As shown in Chapter 3 Section 3.4, we generalize the work by Khuller *et al.* (1999) on the budgeted maximum cover problem to the general submodular framework, and show a practical greedy algorithm with a $(1 - 1/\sqrt{e})$-approximation factor, where each greedy step adds the unit with the largest ratio of objective function gain to scaled cost, while not violating the budget constraint (Lin & Bilmes, 2010b). The algorithm is outlined in Algorithm 4.

The algorithm sequentially finds unit k with the largest ratio of objective function gain to scaled cost, i.e., $(f(G \cup \{k\}) - f(G))/c_k^r$, where $r > 0$ is the scaling factor and c_k is the cost of sentence k (e.g. the number of words in sentence k). If adding k increases the objective function value while not violating the budget constraint, it is then selected and otherwise bypassed. After the sequential selection, set G is compared to the within-budget singleton with the largest objective value, and the larger of the two becomes the final output.

Note that we introduce a scaling factor r to adjust the scale of the cost. Suppose, in the above example, we scale the cost as $c_a = 1^r, c_b = (p+1)^r$, then selecting a or b depends also on the scale r, and we might get the optimal solution using a appropriate r. Indeed,

the objective function values and the costs might be uncalibrated since they might measure different units. E.g., it is hard to say if selecting a sentence of 15 words with an objective function gain of 2 is better than selecting sentence of 10 words with gain of 1. Scaling can potentially alleviate this mismatch (i.e., we can adjust r on development set). Interestingly, our theoretical analysis of the performance guarantee of the algorithm also gives us guidance about how to scale the cost for a particular problem.

4.1.1 Theoretical guarantees

Although Algorithm 4 is essentially a simple greedy strategy, we showed a constant approximation performance guarantee in Theorem 14 in Chapter 3.

Note that an α-approximation algorithm for an optimization problem is a polynomial-time algorithm that for *all* instances of the problem produces a solution whose value is within a factor of α of the value of an optimal solution. So Theorem 14 basically states that the solution found by Algorithm 4 can be at least as good as $(1 - 1/\sqrt{e})f(S^*) \approx 0.39f(S^*)$ even in the worst case. A constant approximation bound is good since it is true for all instances of the problem, and we always know how good the algorithm is guaranteed to be without any extra computation. For $r \neq 1$, we resort to instance-dependent bound where the approximation can be easily computed per problem instance, as stated in Lemma 4.

4.1.2 Performance in practise

The theoretical bounds shown in Theorem 14 and Lemma 4 are worst case bounds. In most cases, the quality of the solutions obtained by Algorithm 14 is be much better than the bounds suggest. In practise, Algorithm 14 finds solutions that are very close to the optimal solutions, and sometimes even finds the exact solutions. Figure 4.1 illustrates one such examples.

Note that when $r = 0$, cost of sentences are not involved in the greedy decision heuristic, which usually decreases the performance of the greedy algorithm. As we can see in Figure 4.1 (and Figure 4.7), algorithm with $r = 0$ only finds a solution that is significantly worse than the solutions found by algorithms with $r > 0$. Actually, most of the greedy-like algorithms

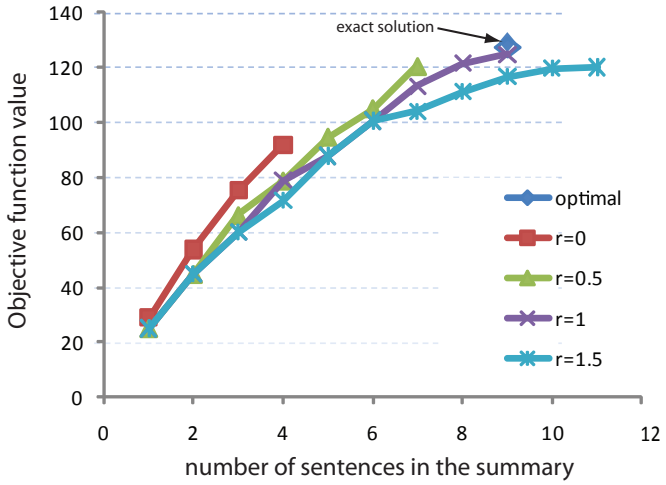


Figure 4.1: Application of Algorithm 4 when summarizing document cluster d30001t in the DUC-04 dataset with summary size limited to 665 bytes. The plots show the achieved objective function as the number of selected sentences grows. The plots stop when in each case adding more sentences violates the budget. Algorithm 4 with $r = 1$ found the optimal solution exactly.

that have been used in document summarization use greedy heuristics that are *purely* based on the objective function gain, without taking into account costs of sentences, which usually lead to solutions that are not near-optimal. See Section 4.4 for further discussion.

4.1.3 Scalability

A desirable property of a practical summarization method is that it is computationally efficient so that it scales to the massive data size that now exist on the Internet. Algorithm 4 possesses such property, not only due to its greedy fashion, but also because the greedy evaluation can be significantly sped up by leveraging the submodularity of the objective function. In particular, when objective function f is submodular, computation at line 4 of Algorithm 4 can be reduced from $O(|V|)$ to $O(\log |V|)$ on average, by using the trick introduced in (Minoux, 1978). On the other hand, integer linear programing (ILP) based algorithms, although capable of finding the exact solutions, do not usually scale. For

instance, when using a graph-based objective function, it could take ILP 17 hours to do the optimization on a document cluster with only about 200 sentences, while Algorithm 14 finishes in milliseconds (Lin & Bilmes, 2010b). See Section 4.4 for further discussion.

4.2 Speech Summarization

Speech summarization is a useful technique to facilitate user browsing a large amount of audio recordings. Automatic speech summarization systems aim to generate a good summary that is concise, informative, and relevant to the input. To summarize and process the speech data, a natural solution is to transcribe the speech recordings to texts, and then summarize the texts. It is essentially a data subset selection problem and therefore the approach introduced in Section 4.1 can be applied here. In particular, we treat speech summarization as a submodular maximization problem with cardinality constraints in this section. In the following, we propose to use several submodular functions to model the quality of a summary.

Common Submodular Objective Functions

Two well-known submodular functions can be used to measure the representativeness of S to the entire set V. The first one is the uncapacitated facility location function (Cornuejols *et al.* , 1977):

$$f_{\text{facility}}(S) = \sum_{i \in V} \max_{j \in S} w_{i,j}. \tag{4.1}$$

This measures the similarity of S to the whole set V. We can also measure the similarity of S to the remainder, i.e., the graph cut function:

$$f_{\text{cut}}(S) = \sum_{i \in V \setminus S} \sum_{j \in S} w_{i,j}. \tag{4.2}$$

Worst-case Objective Function

For $i \in V$, the function

$$g_i(S) = \max_{j \in S} w_{i,j} \tag{4.3}$$

measures the similarity of sentence i to the selected (summary) set S. Actually, $f_{\text{facility}}(S) = \sum_i g_i(S)$ can be viewed as an average of the similarities of *all* the sentences in the document to the summary. In some cases, i.e., when the weights of the graph are noisy, optimizing the average may be inadequate (as we see in our experiments). As an extreme example, consider a document where all the sentences are highly related to each other, except only one of them

is about a somewhat different but important topic. Obviously, the ideal summary for this document should also contain this sentence. Optimizing the average, however, is unlikely to include this sentence in the final summary. This can be resolved if we optimize the worst rather than the average case, motivating our next objective function where we maximize the similarity of the least similar sentence to the summary:

$$f_{\text{worst}}(S) = \min_{i \in V} g_i(S) = \min_{i \in V} \max_{j \in S} w_{i,j}. \tag{4.4}$$

Note that $g_i(S)$ is submodular for all i. However, f_{worst} is not submodular. Fortunately, recent development in robust submodular selection (Krause *et al.* , 2008) enables us to approximately solve the maximization of f_{worst} with strong theoretical approximation guarantees.

Penalty of redundancy

A high quality summary should not only be informative but also compact. Typically, this goal is expressed as a combination of maximizing the information coverage and minimizing the redundancy. Here, we propose the following objective by combining a penalty term $-\sum_{i \in S} \sum_{j \in S, j \neq i} w_{i,j}$ with the graph cut function:

$$f_{\text{penalty}} = \sum_{i \in V \setminus S} \sum_{j \in S} w_{i,j} - \lambda \sum_{i,j \in S: i \neq j} w_{i,j}, \lambda \geq 0. \tag{4.5}$$

This function is still submodular. In MMR (Carbonell & Goldstein, 1998), a similar approach is used, where the algorithm greedily selects sentences that are most similar to the remainder of the document but least similar to the already selected sentences. Our method differs from MMR in that it is graph-based and under a submodular framework. As shown in Sec. 4.2.3, our approach consistently and significantly outperforms MMR.

4.2.1 Algorithms

All the objective functions except $f_{\text{worst}}(\cdot)$ are normalized submodular set functions. In order to benefit from Theorem 9, the objective function should also be nondecreasing. Obviously,

the facility location objective function is nondecreasing since $w_{i,j} \geq 0$. For the graph cut objective, the increment of adding k into S is

$$f_{\text{cut}}(S \cup \{k\}) - f_{\text{cut}}(S) = \sum_{i \in V \setminus S} w_{i,k} - \sum_{j \in S \cup \{k\}} w_{k,j},$$

which is not always nonnegative. Fortunately, the proof of Theorem 9 does not use the monotone property for all possible sets (Nemhauser *et al.* , 1978)(Krause, 2008, page 58). f_{cut} can also respect the conditions for Theorem 9 if $|S| \ll |V|$, which is usually the case in summarization where the extracted summary is usually much smaller than the entire document. Similarly, $f_{penalty}$ can also be non-decreasing in the early stage of selection if λ is not too large. In Section 4.4, we will further theoretically justify this.

Hence for $f_{\text{facility}}, f_{\text{cut}}$ and f_{penalty}, we use greedy algorithms to solve the extractive summarization problem efficiently and near-optimally. The greedy algorithm for f_{facility} is described in Algorithm 6. Note that Algorithm 6 can be seen as an instantiation of Algorithm 4 on objective function f_{facility}, where the excution of Line 4 in Algorithm 4 is sped up for f_{facility} by tracking $\max_{j \in S} w_{i,j}$. In particular, instead of directly evaluting $f_{\text{facility}}(S \cup \{k\}) - f_{\text{facility}}(S)$ (which takes $O(|V||S|)$ time), we keep track of $\max_{j \in S} w_{i,j} \triangleq \rho_i$, and compute $\sum_{i \in V, (i,k) \in E} \max\{\rho_i, w_{i,k}\} - \rho_i$ (which takes $O(|V|)$ time).

The algorithms for f_{cut} and f_{penalty} are similar to Algorithm 6.

f_{worst}, although not submodular, is a minimization over a set of monotone submodular functions and can be optimized using the `SATURATE` algorithm introduced in (Krause *et al.* , 2008). `SATURATE` is an efficient algorithm for the robust submodular observation selection problem which guarantees solutions to be at least as informative as the optimal solution, at only a slightly higher cost. Basically, the algorithm maintains an upper bound for the problem as well as a lower bound for a relaxed version of the original problem. It successively improves the upper and lower bounds using a binary search procedure. The `SATURATE` algorithm for f_{worst} is given in Algorithm 7, and its theoretical gaurantee in Theorem 17.

Theorem 17 (Krause *et al.* , 2008)**.** *For any integer K, Algorithm 7 finds a solution S such that*

$$f_{worst}(S) \geq f_{worst}(S^*), \quad \textit{and} \quad |S| \leq \alpha K,$$

Algorithm 6: Greedy algorithm for f_{facility}

Input: $G = (V, E)$ with weights $w_{i,j}$ on edge (i, j); K: the number of sentences to be selected

Initialization:

$S = \emptyset$,

$\rho_i = 0, i = 1, ..., N$ where $N = |V|$.

while $|S| \leq K$ **do**

  $k^* = \arg\max_{k \in V \setminus S} \sum_{i \in V, (i,k) \in E} (\max\{\rho_i, w_{i,k}\} - \rho_i)$

  $S = S \cup \{k^*\}$

  for all $i \in V$ **do**

    $\rho_i = \max\{\rho_i, w_{i,k^*}\}$

  end for

end while

where

$$\alpha = 1 + \log(\max_{j \in V} \sum_{i \in V} w_{i,j}),$$

and

$$S^* \in \operatorname*{argmax}_{|S| \leq K} f_{worst}(S).$$

4.2.2 Related work

Our approach here is essentially a graph-based approach where we build a semantic graph for the speech utterances to be summarized, and use submodular function optimization over the graph to select summaries. Several graph-based methods have been proposed for extractive summarization previously. Erkan and Radev (Erkan & Radev, 2004) introduced a stochastic graph-based method, *LexRank*, for computing the relative importance of textual units for multi-document summarization. In LexRank the importance of sentences is computed based on the concept of eigenvector centrality in the graph representation of sentences. Mihalcea and Tarau also proposed an eigenvector centrality algorithm on weighted

Algorithm 7: SATURATE algorithm for f_{worst}

Input: $G = (V, E)$ with weights $w_{i,j}$ on edge (i, j); K: the number of sentences to be selected

Initialization:$c_{min} = 0$, $c_{max} = \min_{i \in V} \max_{i \in V} w_{i,j}$, $S^* = \emptyset$, $N = |V|$,$\alpha = 1.1$

while $c_{max} - c_{min} > \frac{1}{N}$ **do**

  $c = (c_{max} - c_{min})/2$; $S = \emptyset$

  Define $\bar{f}_c(S) = \frac{1}{N} \sum_{i \in V} \min\{\max_{j \in S} w_{i,j}, c\}$

  while $\bar{f}_c(S) < c$ **do**

    $S = S \cup \{\arg\max_{k \in V \setminus S} \bar{f}_c(S \cup \{k\}) - \bar{f}_c(S)\}$

  end while

  if $|S| > \alpha K$ **then**

    $c_{max} = c$

  else

    $c_{min} = c, S^* = S$

  end if

end while

graphs for document summarization (Mihalcea & Tarau, 2004). Mihalcea et al. later applied Google's *PageRank* (Brin & Page, 1998) to natural language processing tasks ranging from automatic keyphrase extraction and word sense disambiguation, to extractive summarization (Mihalcea *et al.* , 2004; Mihalcea, 2004). Graph-based ranking algorithms, such as PageRank, can be applied to natural language processing applications by building lexical or semantic graphs extracted from the documents to be processed. In (Mihalcea, 2004), PageRank was adopted to incorporate edge weights, and the power $P(i)$ (importance) of a sentence i was iteratively computed as

$$P(i) = (1 - d) + d \times \sum_{j \in Parent(i)} w_{j,i} \frac{P(j)}{\sum_{k \in Child(j)} w_{k,j}}, \tag{4.6}$$

where d is a parameter usually set between 0 and 1. A summary is then extracted by taking the top K sentences ranked based on $P(i)$. We implemented and compared the PageRank-based algorithm to our approach in the experiments.

4.2.3 Experiments

Experimental setup

We evaluated our approach on the ICSI meeting corpus (Janin *et al.* , 2003). There are 75 meeting recordings in this corpus. Each meeting is about one hour long and has multiple speakers. Since we focus on unsupervised meeting summarization, only the development set and the test set were used in the evaluation, both of which consist of 6 meetings as used in (Murray *et al.* , 2005) and (Xie *et al.* , 2009).

Both human transcripts and automatic speech recognition (ASR) outputs are available for this corpus, where the ASR output is obtained from the state-of-the-art SRI conversational telephone speech system (Zhu *et al.* , 2005), having an overall word error rate of about 38.2%. Three reference summaries from different annotators for each meeting were used for the test set, while for the development set, only one reference summary was used. The lengths of the reference summaries are not fixed and vary across annotators and meetings. The average word compression ratio of the reference summaries is 14.3% with a mean deviation 2.9%. In our experiments, all methods were evaluated when extracting summaries with word compression ratios varying from 13% to 17%.

ROUGE (Lin, 2004), which is widely used in the study of speech summarization (Murray *et al.* , 2005; Zhang *et al.* , 2007; Zhu & Penn, 2006), was used to evaluate summarization performance in our experiments. To be consistent with previous work, we provide ROUGE-1 (unigram) F-measure results for all experiments.

Semantic graphs

We built semantic graphs for each meeting recording in the development and test sets, on both human transcripts and ASR outputs. Two methods were used.

The first method is based on cosine similarity, where the cosine similarity between two sentences D_i and D_j is:

$$w_{i,j} = \text{sim}(D_i, D_j) = \frac{\sum_k t_{ik} t_{jk}}{\sqrt{\sum_k t_{ik}^2} \times \sqrt{\sum_k t_{jk}^2}}, \tag{4.7}$$

where t_{ik} is the TF-IDF (term frequency, inverse document frequency) weight:

$$t_{ik} = n_{ik} \log \frac{N}{N_k}, \tag{4.8}$$

where n_{ik} is the number of occurrences of word W_k in sentence D_i, N_k is the number of documents that contain word W_k and N is the total number of doucments.

To have a better estimate of IDF such that it reflects topic-related importance of a word, we split each of the 75 meetings, for both human transcripts and ASR outputs, into multiple topics based on manual topic segment annotations, and then used these new "documents" to calculate the IDF values. The weighted graph was built by connecting vertices (corresponding to sentences) with non-zero weight. Any unconnected vertex was removed from the graph, which is equivalent to pre-excluding certain sentences from the summary. On average, about 99% of the sentences are preserved in the graph for all meeting recordings on both human transcripts and ASR outputs.

The second method we used is based on the ROUGE score itself. Words with low IDF weights (stop words) were initially removed from the sentences. Since meeting conversations usually contain sentences with only words with low IDF weights (e.g., sentences like "yes","yeah" and etc), the removal of stop words resulted in many "empty" sentences. The similarity measures were then computed as the ROUGE-1 F-measure scores between each pair of stop-word-removed sentences. In particular,

$$w_{i,j} = \text{ROUGE}(D_i, D_j) = \frac{\sum_{e \in D_j} \min(r_{e,i}, r_{e,j})}{\sum_{e \in D_j} r_{e,j}},$$

where $r_{e,i}$ and $r_{e,j}$ are the numbers of times that unigram e occurs in D_i and D_j respectively. The empty sentences always had zero connections in the graph, and thus were pre-excluded from the final summary, yielding sparse final graphs. On human transcripts, the graph preserves only 45% of the sentences while 40% are preserved in the graph for ASR outputs.

Comparison to other approaches

We compared our approach to MMR (Carbonell & Goldstein, 1998) and a global optimization framework using integer linear programming (referred as "ILP" in the rest of this paper)

recently proposed in (Gillick *et al.* , 2009). The setups of both MMR and ILP approaches were the same as the baseline systems introduced in (Xie *et al.* , 2009).

In addition to non-graph-based approaches, we also compared our method to the recursive graph-based ranking algorithm using PageRank. The importance of sentences was estimated in an iterative way using Equation 4.6, where the value for d was set at 0.85 as in (Brin & Page, 1998)(Mihalcea *et al.* , 2004). Iteration stopped when the relative difference from the successive iteration fell below 0.01%. As introduced in (Mihalcea *et al.* , 2004), the graph in this algorithm can be represented as: (a) an undirected graph where a vertex's parents and children are both those vertices connected to it (PageRank-U); (b) a directed weighted graph with the orientation of edges set from sentence to sentences that follow in the text (PageRank-F); or (c) a directed weighted graph with edges oriented from a sentence to previous sentences in the text (PageRank-B).

Results and discussion

Results using the cosine similarity graph are shown in Table 4.1 for human transcripts and in Table 4.2 for ASR outputs. Results with graphs built on ROUGE scores are illustrated in Table 4.3 and Table 4.4 for human transcripts and ASR outputs, respectively. All the results are presented as ROUGE-1 F-measure scores under different word compression ratios ranging from 13% to 17%. In each table, results on both the development (dev.) set and test set of all methods are shown. There are 4 categories of methods: MMR, ILP, PageRank and submodular selection. A number is bold if it beats the results of *all* the methods in the other three categories under the same word compression rate. The best result under the same word compression rate is marked with a "*".

As we can see, for all tables, all the bold and starred numbers appear in the rows for submodular selection, indicating that our graph-based submodular selection outperforms MMR, ILP and PageRank consistently in all cases.

Note that optimizing f_{cut} performs poorly on the graph built on human transcripts using cosine similarity (Table 4.1). One reason is that this graph is quite noisy. Meeting summarization is particularly challenging due to the presence of disfluencies. In the human

Table 4.1: ROUGE-1 F-measure results (%) for different word compression ratios for human transcripts (REF) on both dev. set and test set with graphs based on cosine similarity.

REF.G-cosine.DEV.	ROUGE-1 F-Measure (%)				
Word comp. ratio	13%	14%	15%	16%	17%
MMR	66.28	66.81	67.06	66.90	66.64
ILP	66.46	67.20	67.98	68.30	67.82
PageRank-U	49.54	49.84	50.10	50.20	50.17
PageRank-F	61.41	61.93	62.12	62.15	61.80
PageRank-B	63.01	63.71	64.36	64.21	64.40
Submodular-f_{facility}	**66.71**	67.11	**68.03**	67.92	67.74
Submodular-f_{cut}	60.36	61.45	61.89	62.39	62.57
Saturate-f_{worst}	**69.02***	**69.29***	**69.42***	**69.24***	**68.50***
Submodular-f_{penalty}	**66.70**	**67.26**	67.00	66.73	66.34
REF.G-cosine.TEST	ROUGE-1 F-Measure (%)				
Word comp. ratio	13%	14%	15%	16%	17%
MMR	64.67	65.69	66.23	66.69	66.70
ILP	66.11	67.08	67.84	68.35	68.82
PageRank-U	51.90	52.90	53.41	53.49	53.56
PageRank-F	60.86	61.50	62.19	62.41	62.37
PageRank-B	63.15	63.89	64.50	64.93	64.83
Submodular-f_{facility}	66.06	66.99	67.31	67.62	67.46
Submodular-f_{cut}	60.95	62.17	63.11	63.62	63.91
Saturate-f_{worst}	**67.89***	**68.57***	**69.14***	**69.23***	**69.01***
Submodular-f_{penalty}	**67.45**	**67.89**	**68.30**	**68.35**	67.97

transcripts, disfluencies are precisely transcribed where filler words (such as “so”, “yeah”, “uh”) and partial words are frequently used. This has an impact on the quality of the semantic graph, e.g., two totally semantically irrelevant sentences will still have a (small) nonzero similarity score if they both begin with the filler word “so”. Consequently, one vertex may be weakly connected to many other semantically unrelated vertices, and the f_{cut} function leads to this noise being accumulated to the point of becoming significant. This

Table 4.2: ROUGE-1 F-measure results (%) for different word compression ratios for ASR outputs (ASR) on both dev. set and test set with graphs based on cosine similarity.

ASR.G-cosine.DEV.	ROUGE-1 F-Measure (%)				
Word comp. ratio	13%	14%	15%	16%	17%
MMR	62.59	63.60	64.32	64.80	65.03
ILP	62.59	63.99	65.04	65.45	65.44
PageRank-U	54.56	54.60	54.50	54.41	54.41
PageRank-F	62.21	62.21	62.22	61.97	61.57
PageRank-B	63.83	64.19	64.23	63.94	63.45
Submodular-f_{facility}	**65.54**	**65.82**	**66.15**	**66.21**	**65.72**
Submodular-f_{cut}	63.33	**64.00**	64.15	64.29	63.89
Saturate-f_{worst}	**65.78**	**65.91**	**66.10**	**65.99**	65.40
Submodular-f_{penalty}	**66.71***	**66.81***	**66.71***	**66.60***	**65.90***
ASR.G-cosine.TEST	ROUGE-1 F-Measure (%)				
Word comp. ratio	13%	14%	15%	16%	17%
MMR	61.29	62.35	63.36	63.91	64.22
ILP	62.18	63.30	64.51	65.31	65.27
PageRank-U	56.01	56.17	56.35	56.38	56.33
PageRank-F	61.23	61.78	62.03	62.01	61.70
PageRank-B	61.96	62.50	62.94	62.93	62.88
Submodular-f_{facility}	**64.74**	**65.35**	**65.65**	**65.86**	**65.43**
Submodular-f_{cut}	**63.04**	**63.72**	64.25	64.29	64.05
Saturate-f_{worst}	**64.25**	**64.93**	**65.06**	64.81	64.48
Submodular-f_{penalty}	**66.17***	**66.60***	**66.76***	**66.61***	**66.08***

explains why in Table 4.1 we see f_{cut} (summation of summation) performing poor, f_{facility} (summation of maximums) performing better, and f_{worst} (no summation) performing the best. On the other hand, with ASR outputs (Table 4.2), filler and partial words can be "removed" or miss-recognized by the ASR engine's imperfect output, which tends not to produce a consistent low-level similarity between many sentences. Therefore, Table 4.2's results are less affected by such noise.

Table 4.3: ROUGE-1 F-measure results (%) for different word compression ratio for human transcripts (REF) on both dev. set and test set with ROUGE score based graphs.

REF.G-ROUGE.DEV.	ROUGE-1 F-Measure (%)				
Word comp. ratio	13%	14%	15%	16%	17%
MMR	66.28	66.81	67.06	66.90	66.64
ILP	66.46	67.20	67.98	68.30	67.82
PageRank-U	68.69	69.24	69.24	69.12	68.92
PageRank-F	65.43	66.37	66.69	66.67	66.58
PageRank-B	69.00	69.27	69.63	69.30	68.85
Submodular-f_{facility}	68.53	69.16	69.29	**69.32**	68.83
Submodular-f_{cut}	**69.21***	**69.82***	**69.82***	**70.16***	**69.89***
Saturate-f_{worst}	68.62	69.07	69.32	**69.58**	**68.97**
Submodular-f_{penalty}	68.75	69.17	69.01	69.02	**68.97**
REF.G-ROUGE.TEST	ROUGE-1 F-Measure (%)				
Word comp. ratio	13%	14%	15%	16%	17%
MMR	64.67	65.69	66.23	66.69	66.70
ILP	66.11	67.08	67.84	68.35	68.82
PageRank-U	67.98	69.15	69.69	69.73	69.59
PageRank-F	66.37	67.29	67.81	67.99	68.21
PageRank-B	67.30	68.02	68.40	68.73	68.50
Submodular-f_{facility}	67.53	68.65	69.03	69.31	68.87
Submodular-f_{cut}	**68.00**	**69.04**	**69.94**	**70.27**	**70.16**
Saturate-f_{worst}	67.75	68.82	69.20	69.60	**69.61**
Submodular-f_{penalty}	**69.08***	**69.65***	**70.07***	**70.50***	**70.48***

With the ROUGE-1 graph construction method, words with low TF-IDF scores were removed prior to the actual computation of the similarity scores. The resulting graph is sparse and less noisy. As we can see in Table 4.3 and Table 4.4, all graph-based methods including PageRank perform better on both human transcripts and ASR outputs. Nevertheless, given the same semantic graph, submodular selection outperforms the recursive graph-based ranking algorithm, demonstrating the power of submodularity.

Table 4.4: ROUGE-1 F-measure results (%) for different word compression ratio for ASR outputs (ASR) on both dev. set and test set with ROUGE score based graphs.

ASR.G-ROUGE.DEV.	ROUGE-1 F-Measure (%)				
Word comp. ratio	13%	14%	15%	16%	17%
MMR	62.59	63.60	64.32	64.80	65.03
ILP	62.59	63.99	65.04	65.45	65.44
PageRank-U	64.51	65.16	65.20	65.36	64.98
PageRank-F	63.36	64.13	64.42	64.33	64.15
PageRank-B	65.03	65.43	65.76	65.78	65.43
Submodular-f_{facility}	**64.84**	**65.51**	65.72	65.52	65.07
Submodular-f_{cut}	**65.85**	**66.13***	**66.04***	**66.02***	**65.64***
Saturate-f_{worst}	64.42	65.08	65.47	65.39	65.07
Submodular-f_{penalty}	**65.94***	65.96	**65.94**	**65.82**	**65.48**
ASR.G-ROUGE.TEST	ROUGE-1 F-Measure (%)				
Word comp. ratio	13%	14%	15%	16%	17%
MMR	61.29	62.35	63.36	63.91	64.22
ILP	62.18	63.30	64.51	65.31	65.27
PageRank-U	64.11	64.95	65.49	65.55	65.45
PageRank-F	63.08	63.82	64.54	64.68	64.61
PageRank-B	64.77	65.49	65.62	65.96	65.56
Submodular-f_{facility}	64.35	65.46	**65.98**	65.90	**65.73**
Submodular-f_{cut}	**64.97**	**65.69**	**66.38**	**66.59**	**66.52**
Saturate-f_{worst}	64.15	65.23	**65.88**	**66.02**	**65.80**
Submodular-f_{penalty}	**65.53***	**66.51***	**66.96***	**67.05***	**67.19***

4.3 Training Data Summarization

Given a large un-transcribed corpus of speech utterances, we address the problem of how to select a good subset for word-level transcription under a given fixed transcription budget. We call this task as *training data summarization*. In this section, we employ submodular active selection on a Fisher-kernel based graph over un-transcribed utterances. Our approach is able to bootstrap without requiring *any* initial transcribed data, whereas traditional

approaches rely heavily on the quality of an initial model trained on some labeled data. Experiments on phone recognition show that our approach outperforms both average-case random selection and uncertainty sampling (Lewis & Gale, 1994) significantly.

4.3.1 Introduction

In automatic speech recognition and many other language applications, unlabeled data are abundant but labels (e.g., transcriptions) are expensive and time-consuming to acquire. For example, large amounts of speech data can easily be obtained via telephone calls, and via modern voice-based applications such as Microsoft's Tellme and Google's voice search. Ideally, it would be possible to label all of this data for use as a training set in a speech recognition system, as aptly conveyed by the well known phrase "there is no data like more data." Unfortunately, this would not be practical given today's continual torrent of unlabeled data. Accurate phonetic transcription of speech utterances requires phonetic training and even then it may take a month to annotate 1 hour of speech (Lamel *et al.* , 1989), not to mention the difficulty of transcribing at the articulatory level. Partly due to this, such low-level transcription efforts have been sidelined by the community in favor of word-level transcriptions. But even word level transcriptions are time consuming (about 10 times real time), especially for conversational spontaneous speech. This problem is particularly acute for underrepresented languages or dialects with few speakers, where linguistic experts are even harder to find.

In this section, we address the following question: given limited resources (time and/or budget), how can we optimally select a training data subset for transcription such that the resulting system has optimal performance. In fact, this is a well-known problem and goes by the name of *batch active learning* (Settles, 2009), where a subset of data that is most informative and representative of the whole is selected for labeling. Often, examples are queried in a greedy fashion according to an informativeness measure used to evaluate all examples in the pool. Two popular strategies for measuring informativeness include *uncertainty sampling* and the *query-by-committee* approach. Uncertainty sampling (Lewis & Gale, 1994) is the simplest and most commonly used strategy. In this framework, an

initial system is trained typically using a small set of labeled examples. Then, the system examines the rest of the unlabeled examples, and queries examples that it is most uncertain about. The measurement of uncertainty can either be entropy (Settles & Craven, 2008; Varadarajan *et al.* , 2009; Wu *et al.* , 2007) or a confidence score (Hakkani-tür & Gorin, 2002; Culotta & McCallum, 2005; Settles & Craven, 2008). Query-by-committee (Cohn *et al.* , 1994; Dagan & Engelson, 1995; Tür *et al.* , 2003) also starts with labeled data. A set of distinct models are trained as committee members. Each committee member is then allowed to vote on the labellings of the unlabeled examples. The most informative example is taken as the one the committee most disagrees about.

It has been shown that both uncertainty sampling and query-by-committee may fail when they tend to query outliers, which is the main motivating factor for other strategies like estimated error reduction (Roy & McCallum, 2001). The problem is that outliers might have high uncertainty (or a committee might find them controversial) but they are not good surrogates for "typical" samples. Indeed, an ideal selection strategy should choose a subset of samples that, when considered together, constitute in some form a good representation of the entire training data set. Methods such as (Fujii *et al.* , 1998; Nguyen & Smeulders, 2004; Hoi *et al.* , 2006; Settles & Craven, 2008) address this problem, all of which have been shown to be superior to methods that do not consider representativeness measures. Our approach herein also belongs to this category. In particular, we use Fisher kernel to build a graph over the unlabeled sample sequences, and optimize submodular functions over the graph to find the most representative subset. Note that our Fisher kernel is over an unsupervised generative model, which enables us to bootstrap our active learning approach without needing *any* initial labeled data, yet we achieve good performance perhaps because of the approximate optimality of our submodular procedures. This approach portends well to underrepresented languages for which an initial labeled set might be unavailable.

Despite pre-existing extensive studies of active learning, there is relatively little work on active learning for sequence labeling. Several methods have been proposed, most of which are based either on uncertainty sampling or query-by-committee. In (Tür *et al.* , 2003, 2005; Hakkani-tür & Gorin, 2002), confidence scores from a speech recognizer are used to indicate the informativeness of speech utterances. Active learning methods in (Scheffer

et al. , 2001) select the most uncertain examples based on EM-style algorithm for learning HMMs from partially labeled data. In (Anderson & Moore, 2005), several objective functions and algorithms are introduced for active learning in HMMs. Several new query strategies for probabilistic sequence models are introduced in (Settles & Craven, 2008) and an empirical analysis is conducted on a variety of benchmark datasets. Our approach can be distinguished from these methods in that we select the most representative subset in a *submodular* framework, where submodularity theoretically guarantees that the selection problem can be solved efficiently and near-optimally. Submodularity has already been successfully used in active learning tasks. Robust submodular observation selection is explored in (Krause, 2008). In (Hoi *et al.* , 2006), the authors relate Fisher information matrices to submodular functions so that the optimization can be done efficiently and effectively. Guillory & Bilmes (2010) recently generalized submodular set cover (Wolsey, 1982) and active learning to interactive submodular set cover. To the best of our knowledge, our approach is the first work that incorporates submodularity for active learning in sequence labeling tasks such as speech recognition.

Similar to speech summarization (Section 4.2), we want to select a good subset S of training data V that maximizes some objective function, such that the size of S is no larger than K (our budget). We use $f_{\text{facility}}, f_{\text{cut}}$ as our objectives. Again, we use the greedy algorithm to solve the data selection problem efficiently and near-optimally. The greedy algorithm for submodular data selection with the facility location objective is described in Algorithm 4, where $\rho_i = \max_{j \in S} w_{i,j}$ is updated to optimize the running of the algorithm. The graph-cut objective algorithm is similar and omitted here.

4.3.2 Fisher Kernel

We express the pairwise "similarity" between the utterances i and j in terms of kernel function $\kappa(i, j)$ so that $w_{i,j} = \kappa(i, j)$. Since the examples are sequences with possibly different lengths, we use the Fisher kernel (Jaakkola & Haussler, 1999), which is applicable to variable length sequences. Consider a generative model (e.g., a hidden Markov models, or more generally, a dynamic Bayesian network (DBN)) with parameters θ that models the

generation process of the sequence. Denote $X_i = (x_{i,1}, \ldots, x_{i,T_i})$ as the i^{th} feature sequence with length T_i. Then a fixed length vector, known as the Fisher score, can be extracted as:

$$U_i = \frac{\partial}{\partial \theta} \log p(X_i|\theta) \tag{4.9}$$

Each component of U_i is a derivative of the log-likelihood score for the sequence X_i with respect to a particular parameter — the Fisher score is thus a vector having the same length as the number of parameters θ. The computation of gradients in Eq. 4.9 in the context of DBNs is described in detail in (Bilmes, 2008).

Given Fisher scores, different sequences with different lengths may be represented by fixed-length vectors, so we can easily define several Fisher kernel functions to measure pairwise similarity, e.g., cosine similarity, radial-basis function (RBF) kernel similarity, or as shown below, the negative ℓ_1 similarity:

$$\text{Negative } \ell_1 \text{ norm: } \kappa(i,j) = -||U_i - U_j||_1 \tag{4.10}$$

The generative model that is used to generate the Fisher score may contain several types of parameters (i.e., discrete conditional probability tables and continuous Gaussian parameters), and the values associated with different types of parameters may have quite different numeric dynamic ranges. In order to reduce the heterogeneity within the Fisher score vector, all our experiments apply the following global variance normalization to produce the final Fisher score vectors U_i':

$$U_i' = (\mathbf{diag}(\Sigma))^{-\frac{1}{2}} \cdot (U_i - \bar{U}) \tag{4.11}$$

where $\bar{U} = \frac{1}{N}\sum_{i=1}^{N} U_i$ and $\Sigma = \frac{1}{N}\sum_{i=1}^{N}(U_i - \bar{U})^T(U_i - \bar{U})$

4.3.3 Experiments

We evaluated our methods on a phone recognition task using the TIMIT corpus. Random selection was used as a baseline. Specifically, we randomly take p% of the TIMIT training set, where $p = 2.5, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90$. For each subset, a 3-state context-independent (CI) hidden Markov model (HMM) (implemented as a DBN; see (Bilmes & Bartels, 2005) for details) was trained for each of the 48 phones. The number of Gaussian

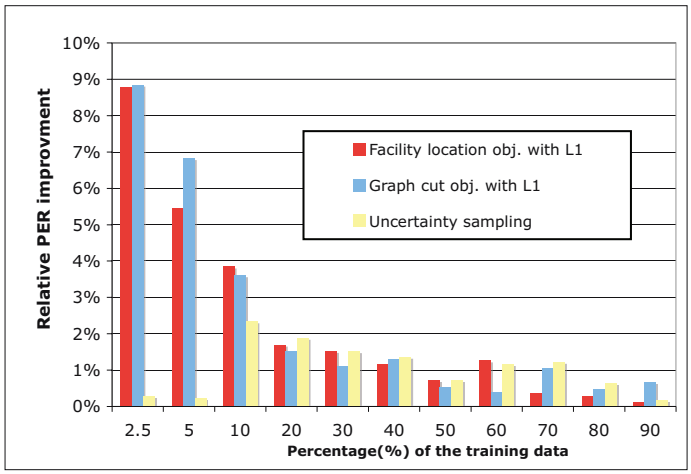


Figure 4.2: Relative improvements over the average phone error rate of random selection. No initial model scenario.

components in the Gaussian mixture model (GMM) was optimized according to the amount of training data available. In particular, GMMs with components sizes of $2, 4, 8, 16, 32$ and 64 were trained using available data, and the one with best development set performance was used. The 48 phones were then mapped down to 39 phones for scoring purposes following standard practice (Lee & Hon, 1989). Recognition was performed using standard Viterbi search without a phonetic language model (a language model was not used here to emphasize the acoustic modeling performance, and since this speeds up experimental turnaround time by avoiding tedious language model scaling and penalty parameter tuning when large random selection experiments are performed). 100 trials of random selection experiments were performed for each of the percentage numbers above. The average phone error rates (PER) were calculated and used as baseline. The standard deviation was around 0.01 for small p and about 0.005 for larger p. Apart from the data selection strategy, experiments on uncertainty sampling and submodular selection followed exactly the same setups as random selection.

Uncertainty sampling and submodular selection were evaluated under two scenarios. The first scenario we considered is when there is no initial model available. In this scenario,

uncertainty sampling would typically randomly select a small portion of the unlabeled data to label, and then train an initial model using these randomly selected data. We did the following: a) randomly select $\alpha\%$ of the training data, acquire the labels and train an initial model; b) use the learned model to predict the unlabeled data, select the M most uncertain samples for labelling; c) retrain the model using all labeled data. If the number of labeled samples reaches the target amount, stop, else go to step b). We used $\alpha = 1$ and $M = 100$ in the experiments, and the average per-frame log-likelihood was used as the uncertainty measurement.

For our submodular selection method, HMMs with 16-component GMMs were obtained by unsupervised training using all the unlabeled data. This model was used as the generative model for the Fisher score using `gmtkKernel`, a GMTK (Bilmes & Bartels, 2005) DBN implementation of Fisher kernels. The negative ℓ_1 norm was used to construct the graph (we also tested other measures which had similar results). The relative PER improvements over the average of the 100 random experiments are shown in Figure 4.2. As we can see, uncertainty sampling achieves improvements over random sampling in general, but when the target percentage number is small (i.e., 2.5% and 5%), which is usually the case in real-world applications, it performs similarly to random selection since the model used for the uncertainty measurement is of low quality. On the other hand, submodular data selection outperforms both random selection and uncertainty sampling when the percentage is small. This implies that even a model trained without any labeling information works quite well for our approach. In other words, the submodular data selection approach proposed here is quite robust to the scenario where no initial "boot" model is available.

Note that when the percentage is larger (e.g., on 70% and 80%), submodular data selection has similar performance to uncertainty sampling, and uncertainty sampling even outperforms submodular approach slightly. First of all, the differences here are small ($< 1\%$). Secondly, note that for uncertainty sampling approach, we gradually increase the amount of labelled data and update the model accordingly, while for the submodular selection approach, the graph is based on a model trained without any label information, and never get updated when labeled data become available. One could imagine that better performance might be achieved by updating the graph using Fisher kernel generated with better models

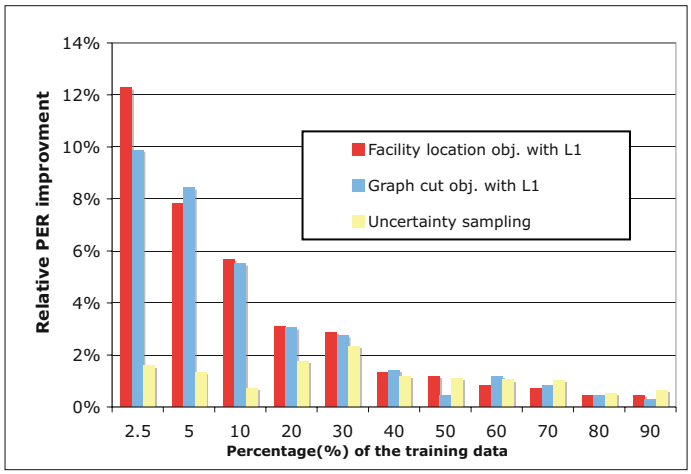


Figure 4.3: Relative improvements over the average phone error rate of random selection. With initial model scenario.

(trained on some labeled data).

Our second scenario is when an initial model is available to help the data selection. Such a model should have reasonable quality. In our experiments, we assume a very high quality initial model to strongly contrast with our first scenario – an initial model with 16-component GMM-HMMs was trained on *all* the labeled TIMIT data, which was then used in the uncertainty sampling approach, and also in the submodular selection method as the generative model. The results are shown in Figure 4.3 — uncertainty sampling performs better with a better quality initial model, when selecting small percentages of the data but not necessarily with more data (presumably due to its selection of unrepresentative outliers). Submodular data selection also performs better in general with a better quality initial model. In particular, more than 12% relative improvement over random selection is achieved when selecting 2.5% of the data. And again, submodular selection outperforms both random sampling and uncertainty sampling. Also, notice that there are only relatively minor performance drops in our approach when shifting from a supervised trained initial model (Figure 4.3) to an unsupervised trained initial model (Figure 4.2), illustrating yet again that submodular selection seems robust to the quality of the initial model.

4.4 Generic Document Summarization

The goal of automatic document summarization is to help individuals deal with large quantities of text data. Document summarization must address problems in both language semantics and data compression in order to extract meaning from a large body of text in a concise and precise way.

There are several variants of document summarization. For example, based on the constituent sentences of a summary, a summarization can be conducted using either *extraction* (select only sentences from the original document set), or *abstraction*, (involving natural language generation). Summarization can also be generic (summarize a given collection of documents) or query-focused (the summary generated should be relevant only to a given query, something more akin to web search). In this section, we address the generic summarization problem, and query-focused summarization in next section.

In extractive text summarization, textual units (e.g., sentences) from a document set are extracted to form a summary, where grammaticality is assured at the local level. Finding the optimal summary can be viewed as a combinatorial optimization problem which is NP-hard to solve (McDonald, 2007). One of the standard methods for this problem is called *Maximum Marginal Relevance* (MMR) (Dang, 2005; Carbonell & Goldstein, 1998), where a greedy algorithm selects the most relevant sentences, and at the same time avoids redundancy by removing sentences that are too similar to already selected ones. One major problem of MMR is that it is non-optimal because the decision is made based on the scores at the current iteration. McDonald (2007) proposed to replace the greedy search of MMR with a globally optimal formulation, where the basic MMR framework can be expressed as a knapsack packing problem, and an integer linear program (ILP) solver can be used to maximize the resulting objective function. ILP Algorithms, however, can sometimes either be expensive for large scale problems or themselves might only be heuristic without associated theoretical approximation guarantees.

In the previous two sections, we modeled problems of interest with cardinality constraint and already showed some supreme results. From this section on, we consider using knapsack constraints, which essentially generalize cardinality constraints. A budget constraint is

natural in summarization tasks as the length of the summary is often restricted. The length (byte budget) limitation represents the real world scenario where, for example, summaries are displayed using only limited computer screen real estate at a given fixed font size.

We use the proposed modified greedy algorithm (Algorithm 4) with f_{penalty} (Eqn. 4.5) as the objective function, i.e.

$$f_{\text{penalty}}(S) = \sum_{i \in V \setminus S} \sum_{j \in S} w_{i,j} - \lambda \sum_{i,j \in S: i \neq j} w_{i,j}, \lambda \geq 0.$$

As mentioned before, this function is submodular but non-monotone. Note that both Theorem 14 and Lemma 4 are for monotone submodular functions while our practical objective function (f_{penalty}) is not guaranteed everywhere monotone. However, our theoretical results still holds for f_{penalty} with high probability (Theorem 18).

Intuitively, in summarization tasks, the summary is usually small compared to the ground set size ($|S| \ll |V|$). When $|S|$ is small, f_{penalty} is often monotone and our theoretical results still hold. Precisely, assume that all edge weights are bounded: $w_{i,j} \in [0, 1]$ (which is the case for cosine similarity between non-negative vectors). Also assume that edges weights are independently identically distributed with mean μ, i.e. $\mathbb{E}(w_{i,j}) = \mu$. Given a budget $\mathcal{B}$, assume the maximum possible size of a solution is K. Let $\alpha = 2\lambda + 1$, and $\beta = 2K - 1$. Notice that $\beta \ll |V|$ for our summarization task. We have the following theorem:

Theorem 18. *Algorithm 4 solves the summarization problem near-optimally (i.e. Theorem 14 and Lemma 4 hold) with high probability of at least*

$$1 - \exp\left\{-\frac{2(|V| - (\alpha + 1)\beta)^2\mu^2}{|V| + (\alpha^2 - 1)\beta} + \ln K\right\}$$

Proof. See Appendix A. □

The probability given in Theorem 18, for example, is about $1 - \exp(-6.5) = 0.9985$ given a typical document with 500 sentences and a summary with at most 30 sentences under the budget, i.e. $|V| = 500, K = 30$ and $\lambda = 2$.

4.4.1 Experiments

We evaluated our approach on the data set of DUC'04 with the setting of task 2, which is a multi-document summarization task on English news articles. In this task, 50 document clusters are given, each of which consists of 10 documents. For each document cluster, a short multi-document summary is to be generated. The summary should not be longer than 665 bytes including spaces and punctuation, as required in the DUC'04 evaluation. We used DUC'03 as our development set. All documents were segmented into sentences using a script distributed by DUC. ROUGE version 1.5.5 (Lin, 2004), which is widely used in the study of summarization, was used to evaluate summarization performance in our experiments [3]. We focus on ROUGE-1 (unigram) F-measure scores since it has demonstrated strong correlation with human annotation (Lin, 2004).

The basic textual/linguistic units we consider in our experiments are sentences. For each document cluster, sentences in all the documents of this cluster forms the ground set V. We built semantic graphs for each document cluster based on cosine similarity, where cosine similarity is computed based on the TF-IDF (term frequency, inverse document frequency) vectors for the words in the sentences. The cosine similarity (Eqn 4.7) measures the similarity between sentences, i.e., $w_{i,j}$.

Here the IDF values were calculated using all the document clusters. The weighted graph was built by connecting vertices (corresponding to sentences) with weight $w_{i,j} > 0$. Any unconnected vertex was removed from the graph, which is equivalent to pre-excluding certain sentences from the summary.

Comparison with exact solution

In this experiment, we empirically show that Algorithm 4 works near-optimally in practice. To determine how much accuracy is lost due to approximations, we compared our approximation algorithms with an exact solution. The exact solutions were obtained by Integer Linear Programming (ILP). Solving arbitrary ILP is an NP-hard problem. If the size of the problem is not too large, we can sometimes find the exact solution within a manageable

[3] Options used: -a -c 95 -b 665 -m -n 4 -w 1.2

time using a branch-and-bound method. In our experiments, MOSEK (MOSEK, 2010) was used as our ILP solver.

We formalize budgeted submodular maximization with f_{penalty} objective as an ILP by introducing indicator (binary) variables $x_{i,j}, y_{i,j}, i \neq j$ and z_i for $i, j \in V$. In particular, $z_i = 1$ indicates that unit i is selected, i.e., $i \in S$, $x_{i,j} = 1$ indicates that $i \in S$ but $j \notin S$, and $y_{i,j} = 1$ indicates both i and j are selected. Adding constraints to ensure a valid solution, we have the following ILP formulation for budgeted submodular maximization with objective function $f_{\text{penalty}}(S)$:

$$\begin{aligned}
&\text{maximize} \sum_{i \neq j, i,j \in V} w_{i,j} x_{i,j} - \lambda \sum_{i \neq j, i,j \in V} w_{i,j} y_{i,j} \\
&\text{subject to:} \sum_{i \in V} c_i z_i \leq \mathcal{B}, \\
&x_{i,j} - z_i \leq 0, x_{i,j} + z_j \leq 1, z_i - z_j - x_{i,j} \leq 0, \\
&y_{i,j} - z_i \leq 0, y_{i,j} - z_j \leq 0, z_i + z_j - y_{i,j} \leq 1, \\
&x_{i,j}, y_{i,j}, z_i \in \{0, 1\}, \forall i \neq j, i, j \in V
\end{aligned}$$

Note that the number of variables in the ILP formulation is $O(|V|^2)$. For a document cluster with hundreds of candidate textual units, the scale of the problem easily grows involving tens of thousands of variables, making the problem very expensive to solve. For instance, solving the ILP exactly on a document cluster with 182 sentences (as used in Figure 4.1) took about 17 hours while our Algorithm 4 finished in less than 0.01 seconds.

We tested both approximate and exact algorithms on DUC'03 data where 60 document clusters were used (30 TDT document clusters and 30 TREC document clusters), each of which contains 10 documents on average. The true approximation factor was computed by dividing the objective function value found by Algorithm 4 over the optimal objective function value (found by ILP). The average approximation factors over the 58 document clusters (ILP on 2 of the 60 document clusters failed to finish) are shown in Table 4.5, along with other statistics. On average Algorithm 4 finds a solution that is over 90% as good as the optimal solution for many different r values, which backs up our claim that the modified greedy algorithm solves the problem near-optimally, even occasionally optimally (Figure 4.1

shows one such example).

The higher objective function value does not always indicate higher ROUGE-1 score. Indeed, rather than directly optimizing ROUGE, we optimize a surrogate submodular function that indicates the quality of a summary. Optimality in the submodular function does not necessary indicate optimality in ROUGE score. Nevertheless, we will show that our approach outperforms several other approaches in terms of ROUGE. We note that ROUGE is itself a surrogate for true human-judged summary quality, it might possibly be that f_{MMR} is a still better surrogate — we do not consider this possibility further in this work, however.

Table 4.5: Comparison of Algorithm 4 to exact algorithms on DUC'03 dataset. All the numbers shown in the table are the average statistics (mean/std). The "true" approximation factor is the ratio of objective function value found by Algorithm 4 over the ILP-derived true-optimal objective value, and the approximation bounds were estimated using Theorem 4.

	Approx. factor		ROUGE-1
	true	bound	(%)
exact	1.00	-	33.60/5.05
$r = 0.0$	0.65/0.15	$\geq$0.19/0.08	33.50/5.94
$r = 0.1$	0.71/0.15	$\geq$0.24/0.08	33.68/6.03
$r = 0.3$	0.88/0.11	$\geq$0.37/0.06	34.77/5.49
$r = 0.5$	0.96/0.04	$\geq$0.48/0.05	34.33/5.94
$r = 0.7$	0.98/0.02	$\geq$0.56/0.05	34.08/5.41
$r = 1.0$	0.98/0.02	$\geq$0.65/0.04	33.32/5.14
$r = 1.2$	0.97/0.02	$\geq$0.48/0.05	32.54/4.69

4.4.2 Summarization Results

We used DUC'03 (as above) for our development set to investigate how r and λ relate to the ROUGE-1 score. From Figure 4.4, the best performance is achieved with $r = 0.3, \lambda = 4$. Using these settings, we applied our approach to the DUC'04 task. The results, along with

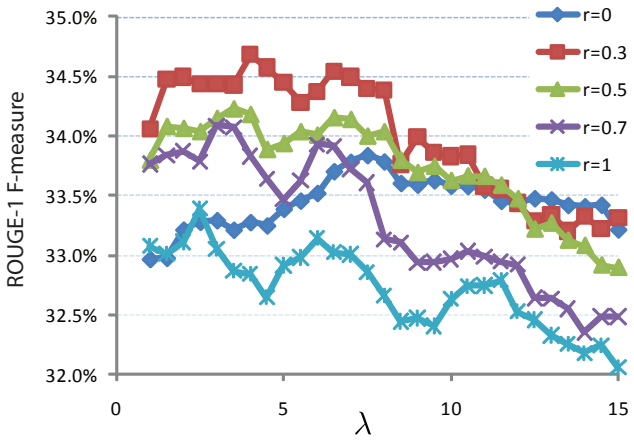


Figure 4.4: Different combinations of r and λ for f_{MMR} related to ROUGE-1 score on DUC'03 task 1.

the results of other approaches, are shown in Table 4.6. All the results in Table 4.6 are presented as ROUGE-1 F-measure scores. [4]

We compared our approach to two other well-known graph-based approaches, LexRank and PageRank. LexRank was one of the participating systems in DUC'04, with peer code 104 (a peer code was assigned for each participant). For PageRank, we implemented the recursive graph-based ranking algorithm ourselves. The importance of sentences was estimated in an iterative way as in (Brin & Page, 1998)(Mihalcea *et al.* , 2004). Sentences were then greedily selected based on their importance rankings until the budget constraint was violated. The graphs used for PageRank were *exactly* the graphs in our submodular approaches (i.e., an undirected graph). In both cases, submodular summarization achieves better ROUGE-1 scores. The improvement is statistically significant by the Wilcoxon signed rank test at level $p < 0.05$. Our approach also outperforms the best system (Conroy *et al.* , 2004), peer code 65 in the DUC'04 evaluation although not as significant ($p < 0.08$). The reason might be that DUC'03 is a poor representation of DUC'04 — indeed, by varying r

[4]When the evaluation was done in 2004, ROUGE was still in revision 1.2.1, so we re-evaluated the DUC'04 submissions using ROUGE v1.5.5 and the numbers are slightly different from the those reported officially.

and λ over the ranges $0 \leq r \leq 0.2$ and $5 \leq \lambda \leq 9$ respectively, the DUC'04 ROUGE-1 scores we got using our submodular approach were all $> 38.8\%$ with the best DUC'04 score being 39.3%.

Table 4.6: ROUGE-1 F-measure results (%)

Method	ROUGE-1 score
peer65 (best system in DUC04)	37.94
peer104 (LexRank)	37.12
PageRank	35.37
Submodular ($r = 0.3, \lambda = 4$)	**38.39**

4.5 Query-focused Document Summarization

In query-focused (i.e., question-based, or topic-focused) summarization, a user query (e.g., a specific topic description) is given, and the task is to generate a summary from a document (or a set of documents) which either explains the query or answers the need for information expressed in the query. The challenges for query-focused summarization are as follows. First, as is common with document summarization, the summary should be compact while still covering all relevant information scatted about in the set of documents. The second challenge, which is unique to query-focused summarization, is that information contained in the summary should indeed be query-relevant. Therefore, effective summarization methods that take into account the query during the summarization process is desired.

Many methods have been proposed for query-focused summarization. The most commonly used approaches usually consist of two major steps. In the first step, key sentences (or other linguistic units) are identified based on their relevance to the query. The key sentences could be found either by training a binary classifier (Kupiec *et al.* , 1995), or by directly assigning weights to sentences based on a variety of features (Toutanova *et al.* , 2007), or by ranking sentence models that are statistically related to both document and query models in a Bayesian fashion (Daumé III & Marcu, 2006). In the second step, relevant sentences are selected and chosen to be included into the summary while simultaneously avoiding redundancy. One of the standard methods for this step is called *maximum marginal relevance* (MMR) (Carbonell & Goldstein, 1998), where a greedy algorithm selects the most relevant sentences, and at the same time avoids redundancy by removing sentences that are too similar to already selected ones.

In this section, we propose three approaches to explore submodularity in query-focused summarization. We show that we are able to achieve extremely high quality results with only a fairly simple and efficient algorithm based on the submodularity of the functions to be optimized. Our approaches are also graph-based, i.e., documents are represented as a weighted graph $G = (V, E; W)$, where V is the ground set of the candidate sentences, E is the edge set, and W is the weights associated with the edges. When a query Q (e.g., a description of a topic) is given, we adapt the graph to be query-dependent such that the

information carried in Q can be incorporated into the graph seamlessly. The query-focused summarization problem is then formulated as constrained submodular function optimization over a "query-focused" graph, say G_Q, where we can leverage the power of submodularity to solve the problem efficiently and effectively. We introduce several ways of producing the query-depend graphs as well as the submodular functions that are defined over these graphs.

In the following, we assume that for each $i \in V$, the relevance of sentence i to the query Q is measured by q_i.

There could be several ways to estimate q_i. For instance, we could use the cosine similarity as inspiration and form:

$$q_i = \frac{\sum_{w \in Q} \mathrm{tf}_{w,Q} \times \mathrm{tf}_{w,i} \times \mathrm{idf}_w^2}{\sqrt{\sum_{w \in Q} \mathrm{tf}_{w,Q}^2 \mathrm{idf}_w^2} \sqrt{\sum_{w \in s_i} \mathrm{tf}_{w,i}^2 \mathrm{idf}_w^2}} \tag{4.12}$$

where idf_w is the inverse document frequency (IDF) of term w, and $\mathrm{tf}_{w,Q}$ and $\mathrm{tf}_{w,i}$ are the term frequency (TF) numbers of times that w appears in Q and sentence s_i, respectively.

We could also use the relevance measure that has proven to be successful in query-based sentence retrieval (Allan *et al.* , 2003):

$$q_i = \sum_{w \in Q} \log(\mathrm{tf}_{w,Q} + 1) \times \log(\mathrm{tf}_{w,i} + 1) \times \mathrm{idf}_w. \tag{4.13}$$

We next introduce three novel ways of incorporating query information into the submodular summarization framework.

4.5.1 Vertex-induced Subgraph

In this approach, sentences that are query-unrelated are firstly excluded from the candidate set (ground set), summarization is then performed on the reduced set, where generic summarization techniques, such as the submodular function optimization approach (Lin *et al.* , 2009; Lin & Bilmes, 2010b), can be applied. The logic behind this approach is that when dealing with query-focused summarization, we only need to summarize the content that is query-related.

In particular, we firstly exclude the vertices that are not similar to the query Q, and then a subgraph is induced based on this reduced set of vertices, say V_Q where $V_Q \subseteq V$,

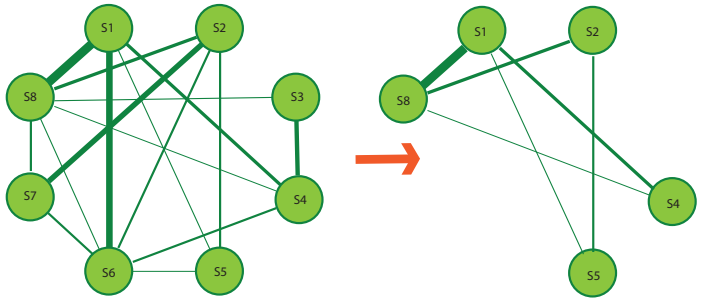


Figure 4.5: An example of a vertex induced subgraph $G_Q = (V_Q, E_Q)$, which is on the right, based on $G = (V, E)$, which is on the left.

i.e. we build a vertex-induced subgraph $G_Q = (V_Q, E_Q)$, where a subset of the vertices are chosen and only edges $E_Q = (V_Q \times V_Q) \cap E$ whose endpoints are both in this subset are preserved. Figure 4.5 shows one such example.

Excluding less relevant sentences can be done by setting a threshold θ on the relevance measurement q_i. Precisely, we define the subset of vertices for a given threshold θ as

$$V_Q = \{i \in V | q_i > \theta\}.$$

On the induced subgraph (V_Q, E_Q), we then solve the following problem:

Problem 10. *Given the vertex-induced weighted graph (V_Q, E_Q) where each node $i \in V_Q$ is associated with a non-negative cost c_i and each edge $e_{i,j} \in E_Q$ associates with a non-negative weight $w_{i,j}$, find Find*

$$S^* = \arg\max_{S \subseteq V_Q} f(S) \text{ subject to: } \sum_{i \in S} c_i \leq b$$

This is exactly the problem for generic summarization (i.e. Problem 9) except that the set function f is now defined on the reduced set V_Q.

We use the f_{penalty} as the objective function defined on subsets of V_Q:

$$f_{\text{penalty}}(S) = \mathcal{X}(S; V_Q \setminus S) - \gamma \mathcal{R}(S) \tag{4.14}$$

where $\mathcal{X}(S; V_Q \setminus S)$ is the graph cut function, i.e.

$$\mathcal{X}(S; V_Q \setminus S) = \sum_{i \in S} \sum_{j \in V_Q \setminus S} w_{i,j}, \tag{4.15}$$

$\mathcal{R}(S)$ measures the redundancy in the summary set S:

$$\mathcal{R}(S) = \sum_{i \in S} \sum_{j \in S, j \neq i} w_{i,j}, \tag{4.16}$$

and $\gamma \geq 0$ is a trade-off coefficient. f_{MMR} is submodular since $\mathcal{X}$ is submodular, $\mathcal{R}$ is supermodular and non-negative scaling preserves submodularity (supermodularity).

Note that the computational complexity of producing the induced subgraph approach is only $O(|V|)$ which is quite affordable.

4.5.2 Query-focused Objective function

In the previous approach, although we can solve the optimization problem on the vertex-induced subgraph both near-optimally and efficiently, the subgraph inducing procedure excludes certain candidates before optimization occurs. While the optimization on the subgraph is still optimal, the entire summarization process can be far from optimal. The main reason for this due to the lack of transitivity in this approach: intuitively, it is not only sentences that get high relevance scores (i.e. q_i) that we wish to include in our summary, but also sentences that are related to q_i should sometimes be included to produce a better summary, even if they are not directly (and quantitatively based on our measure) related to the query. Excluding such sentences can hurt the coverage of the final summary. Another way of viewing this is via graph-based semi-supervised transductive learning: if an unlabeled point lies on the geodesicaly shortest path between two identically labeled points, that point is a likely candidate to receive the same label. In summarization, if a sentence is geodesically close to two query-relevant sentences, perhaps that sentence should be included in the summary if it is not too redundant with those two query-relevant sentences.

In our second approach introduced in this section, we address this issue. We utilize the inter-sentence interaction and formularize the query-focused summarization problem as a single optimization problem. In particular, we propose to optimize the following objective

function:

$$f_Q(S) = \alpha\mathcal{X}(S; V \setminus S) + \beta\mathcal{Q}(S) - \gamma\mathcal{R}(S) \tag{4.17}$$

where $\mathcal{X}, \mathcal{R}$ are graph cut functions and redundancy functions as introduced before, $\alpha, \beta, \gamma \geq 0$ are trade-off co-efficients, and $\mathcal{Q}(S)$ measures the relevance of the summary set S to the query Q.

As both $\mathcal{X}$ and $-\mathcal{R}$ are submodular functions, in order to make the objective function f_Q submodular, we need to find a $\mathcal{Q}$ that models the relevance as well as being submodular. One handy option is:

$$\mathcal{Q}(S) = \sum_{i \in S} q_i, \tag{4.18}$$

which is in fact modular and thus submodular, therefore rendering $f_Q(S)$ submodular.

Query-focused summary can then be found by solving Problem 11:

Problem 11. *Given a weighted graph (V, E) where each node $i \in V$ is associated with a non-negative cost c_i and a non-negative score q_i, and each edge $e_{i,j} \in E$ associates with a non-negative weight $w_{i,j}$, find*

$$S^* = \arg\max_{S \subseteq V} f_Q(S) \text{ subject to: } \sum_{i \in S} c_i \leq b.$$

Note that there are two special cases of the objective function f_Q:

- $\beta = 0, \alpha > 0, \gamma > 0$. This reduces to the generic summarization, i.e. everything is independent from the query Q.

- $\alpha = 0, \beta > 0, \gamma > 0$. This is similar to common query-focused summarization approach in the sense it finds the most relevant sentences while being compact. It is an essentially different approach, however, as both relevance and redundancy are jointly optimized here, via submodular function optimization.

4.5.3 Query-focused Augmented Graph

In previous two approaches, we never re-compute the graph weights to make them query-dependent. This is mainly because such re-weighting is expensive (it would cost $O(|V|^2)$ to recompute weights for each query) and this cannot be pre-computed before seeing the query. On the other hand, it is quite reasonable at least conceptually to take the query Q into account when considering the similarity between two sentences in the context of attempting to describe Q. For instance, when the query is a named entity, repeatedly mentioning such a named entity in the summary should not be seen as exorbitant redundant. That is, if two sentences each mention the named entity several times, they could have a high similarity score in the query-free case, but they should not necessarily have a high similarity in the query-dependent scenario as they might describe very different aspects of the name entity included in the query.

In our third approach, therefore, we propose to partially re-compute graph weights with respect to a query. In the first phase of this approach, a relatively large query-independent (generic) summary is generated for the document. Here a large summary refers to a summary that has cost over our budget b. Denoting this summary as set U with $U \subseteq V$, we have

$$|U| \ll |V|$$

and

$$b \leq \sum_{i \in U} c_i \leq \lambda b$$

where $\lambda > 1$.

On this set U, we form a new graph (U, F) where edges weights are re-computed to be query-focused. Unlike in the complete graph, we are able to recompute the query-dependent edge weights in an affordable complexity of $O(|U|^2)$, since U is still much smaller than V.

As U is a summary of V, we use it as a surrogate of V. In particular, for each $i \in V$, define $\delta(i) \in U$ to be i's surrogate, which is the summary sentence that i is most close to. Figure 4.6 shows one example of graph (V, E), graph (U, F) and their connections.

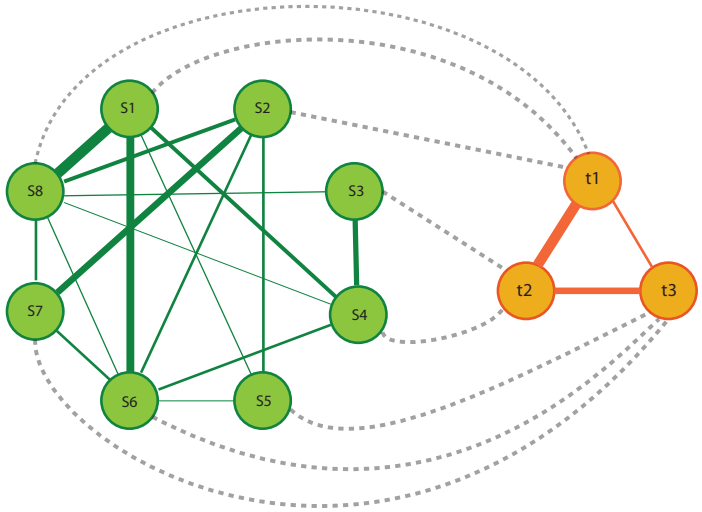


Figure 4.6: An example showing the graph augmentation. On the left is the original document graph (V, E). On the right is the surrogate graph (U, F) with edge weights query-dependent. The dash lines indicate the connections between vertices in V and its surrogate in U. For example, $\delta(s_1) = t_1$, $\delta(\{s_1, s_8\}) = \{t_1\}$, and $\delta(\{s_3, s_4, s_5\}) = \{t_2, t_3\}$.

We also generalize δ to be defined on sets, and let

$$\delta(S) \triangleq \bigcup_{i \in S} \{\delta(i)\}, S \subseteq V.$$

Then, given a set function f on U, we define a new function

$$\mathcal{G}(S) \triangleq f(\delta(S)) \tag{4.19}$$

and we use $\mathcal{G}$ to measure the quality of summary S. Intuitively, the interaction between elements in set S is now measured by the interaction between their surrogates in set U where the graph weights are query-dependent.

We then solve the following problem in this approach:

Problem 12. *Given a weighted graph (V, E) where each node $i \in V$ is associated with a non-negative cost c_i and each edge $e_{i,j} \in E$ associates with a non-negative query-independent weight, and another weighted graph (U, F) where $U \subseteq V$ and each edge $e_{i,j} \in F$ is associated*

with a non-negative query-dependent weight, find

$$S^* = \arg\max_{S \subseteq V} \mathcal{G}(S) \text{ subject to: } \sum_{i \in S} c_i \leq b.$$

Interestingly, in Eqn. 4.19, the submodularity in f can sometimes be transfered to $\mathcal{G}$, as indicated by the following theorem.

Theorem 19. *$\mathcal{G}$ is submodular if f is submodular and non-decreasing.*

Proof. See Appendix A. □

Note that even though $\mathcal{G}$ is submodular when f is monotone submodular, it is not necessary monotone. Also, when f is not monotone, $\mathcal{G}$ is not necessary submodular.

4.5.4 Experiments

Task and evaluation method

The document understanding conference (DUC) was the main evaluation forum providing benchmarks for researchers working on document summarization[5]. The tasks in DUC evolve from single-document summarization to multi-document summarization and from generic summarization (2001–2004) to query-focused summarization (2005–2007). We evaluated our approaches on DUC data. In particular, we use DUC 2006 as our development set and DUC 2007 as our evaluation set.

In DUC 2006, participants were given 50 document clusters, where each cluster contains 25 news articles related to the same topic. Participants were asked to generate summaries of at most 250 words for each cluster. For each cluster, a title and a narrative (query) describing a user's information need are provided. The narrative is usually composed of a set of questions or a multi-sentence task description. Below is an example of a narrative in DUC-05:

Identify and describe types of organized crime that crosses borders or involves more than one country. Name the countries involved. Also identify the perpetrators involved with each type of crime, including both individuals and organizations if possible.

[5] http://duc.nist.org.

The main summarization task in DUC 2007 is the same as in DUC 2006, where participants were required to generate 250-word summaries of 45 clusters of 25 newswire documents each in response to short questions (query) about their content.

For summary evaluation, it is difficult to come up with a universally accepted method to measure the quality of machine-generated summaries [6]. As ROUGE (Lin, 2004) has been officially adopted for DUC automatic evaluations since 2004, we take it as our main evaluation criteria in this paper. In both DUC 2006 and DUC 2007, ROUGE-2 (bigram overlap) was used as the major criteria for evaluation, and therefore we report ROUGE-2 in this paper [7].

Pre-processing

In all experiments, documents and queries were pre-processed by segmenting sentences. and splitting words. Stop-words [8] were then removed and the remaining words were stemmed with the Porter Stemmer. We built our query-independent graphs using cosine similarity, where the cosine similarity between two sentences s_i and s_j is computed as:

$$w_{i,j} = \frac{\sum_{w \in s_i} \text{tf}_{w,i} \times \text{tf}_{w,j} \times \text{idf}_w^2}{\sqrt{\sum_{w \in s_i} \text{tf}_{w,s_i}^2 \text{idf}_w^2} \sqrt{\sum_{w \in s_j} \text{tf}_{w,j}^2 \text{idf}_w^2}}. \tag{4.20}$$

Here idf_w is the IDF of term w (up to bigram), and $\text{tf}_{w,i}$ and $\text{tf}_{w,j}$ are the numbers of times that w appears in s_i and sentence s_j, respectively. The IDF values were calculated using all articles in all document clusters (Eqn 4.8). The weighted graph was then built by connecting vertices (corresponding to sentences) with weight $w_{i,j} > 0$. Any unconnected vertex was removed from the graph, which is equivalent to pre-excluding certain sentences from the summary. On average, about 99% of the sentences are preserved in the graph. Similarly, we used cosine similarity to represent the similarity between query and sentences, i.e. q_i were computed using Eqn. 4.12, where the same IDF scores used in graph construction were applied.

---

[6] Workshop on Evaluation Metrics and System Comparison for Automatic Summarization (http://www.nist.gov/tac/2012/WEAS/), 2011.

[7] ROUGE version 1.5.5 was used with option -n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -d -l 250

[8] http://www.ee.washington.edu/people/hlin/stopword.large.list

Results

We adjusted all parameters in our approaches using grid search on DUC 2006, and the best configurations were then applied to DUC 2007. Results are shown in Table 4.7.

For the first approach, objective function in Eqn. 4.14 was used with a grid search on γ and scaling factor r. The best configuration we found for this approach is $\theta = 0, \gamma = 4$. Notice that we used Eqn. 4.12 to measure q_i. Therefore $\theta = 0$ basically means that the reduced vertex set V_Q consists of sentences that are at least have one term overlap with the query.

In the second approach, the best configuration found was $\alpha = 1, \beta = 5, \gamma = 7$, indicating that jointly optimizing the similarity (weighted by α) , redundancy (weighted by β) and the relevance (weighted by γ) is beneficial. On DUC 2007, our second approach achieves competitive results to the best result where the 95% confidence interval overlaps a large portion of that of the best system.

Table 4.7: ROUGE-2 F-measure results (%) on DUC 2007 (test set)

Method	ROUGE-2 (95% conf. interval)
System 15 (Pingali *et al.* , 2007)	12.29 (11.80 – 12.77)
System 4 (Toutanova *et al.* , 2007)	11.89 (11.46 – 12.35)
Approach I (Induced graph)	11.75 (11.34 – 12.17)
Approach II (New subm. obj.)	12.06 (11.59 –12.55)
Approach III (Augmented graph)	11.07 (10.54 – 11.53)

As for the third approach, we used Eqn. 4.14 as f and the query-dependent edge weights on (U, F) were simply estimated by reducing the IDF scores of terms that appear in the query. We tried different sizes of U. Results reported in Table 4.7 are the results with $|U| = 0.5 \times |V|$. When $|U| = |V|$, this approach reduces to the (expensive) case where all the graph weights were recomputed to be query-dependent. Further improvements on this approach is possible by investigating a better way of producing query-dependent edge weights.

4.6 Submodularity in Summarization

In this section, we show that many well-established methods for summarization are in fact correspond to submodular optimization. Even the widely used automatic summarization evaluation metric, ROUGE-N, is monotone submodular, giving evidence that submodular functions are a natural fit for document summarization.

4.6.1 Maximum Margin Relevance (MMR) and related approaches

As mentioned in Section 4.1, summarization problem can be seen as a maximization problem with knapsack (budget) constraint, and greedy-like algorithms, in fact, have been widely used to solve this problem. One of the more popular approaches is *maximum marginal relevance* (MMR) (Carbonell & Goldstein, 1998), where a greedy algorithm selects the most relevant sentences, and at the same time avoids redundancy by removing sentences that are too similar to ones already selected. Interestingly, the gain function defined in the original MMR paper (Carbonell & Goldstein, 1998) satisfies diminishing returns, a fact apparently unnoticed until now. In particular, Carbonell & Goldstein (1998) define an objective function gain of adding element k to set S ($k \notin S$) as:

$$\lambda \texttt{Sim}_1(s_k, q) - (1-\lambda) \max_{i \in S} \texttt{Sim}_2(s_i, s_k), \tag{4.21}$$

where $\texttt{Sim}_1(s_k, q)$ measures the similarity between unit s_k to a query q, $\texttt{Sim}_2(s_i, s_k)$ measures the similarity between unit s_i and unit s_k, and $0 \leq \lambda \leq 1$ is a trade-off coefficient. We have:

Theorem 20. *Given an expression for f_{MMR} such that $f_{MMR}(S \cup \{k\}) - f_{MMR}(S)$ is equal to Eq. 4.21, f_{MMR} is non-monotone submodular.*

Obviously, diminishing-returns hold since

$$\max_{i \in S} \texttt{Sim}_2(s_i, s_k) \leq \max_{i \in R} \texttt{Sim}_2(s_i, s_k)$$

for all $S \subseteq R$, and therefore f_{MMR} is submodular. On the other hand, f_{MMR}, would not be monotone. Note that submodularity and monotonicity are two necessary ingredients to guarantee that the greedy algorithm with scaled cost gives near-optimal solutions. As f_{MMR}

is not monotone, the greedy algorithm's constant-factor approximation guarantee does not apply in this case. Moreover, the greedy algorithm of MMR does not take cost into account, and therefore could lead to solutions that are significantly worse than the solutions found by Algorithm 14 with scaled cost (see Figure 4.1 and 4.7 for examples).

Actually, MMR-like objective functions, in which a relevance (coverage) term is combined with a redundancy penalization term, are commonly used in document summarization. For instance, in (McDonald, 2007), the following objective function is studied:

$$f_{\text{MMR-I}}(S) = \sum_{i \in S} \texttt{Rel}(i) - \sum_{i,j \in S, i<j} \texttt{Red}(i,j) \tag{4.22}$$

where $\texttt{Rel}(i)$ represents the relevance of textual unit (sentence) i, and $\texttt{Red}(i,j)$ represents the redundancy between unit i and j. This function is submodular but non-monotone (see Appendix A for the proof). In this function, average redundancy of a sentence to other sentences in the summary is penalized, whereas in the original MMR, maximum is used instead of summation (averaging).

The f_{penalty} function (Eqn. 4.5) used in Section 4.2, 4.4 and 4.5 where a graph cut function, measuring the similarity of the summary to the rest of document, is combined with a subtracted redundancy penalty function. The objective function is submodular but again, non-monotone. We theoretically justify that the performance guarantee of the greedy algorithm holds for this objective function with high probability (Theorem 18). Our justification, however, is shown to be applicable only to certain particular non-monotone submodular functions, under certain reasonable assumptions about the probability distribution over weights of the graph.

Because of the redundancy penalization terms, all these MMR and MMR-like objective functions do not have monotone property. As we will see in Section 4.7, we propose to positively reward diversity, instead of negatively penalizing redundancy, leading to a rich family of monotone submodular functions that not only naturally fit into the paradigm of summarization modeling, but also equipped with fast and scalable algorithm with performance guarantee.

4.6.2 Concept-based approaches

When scoring a summary at the sub-sentence level, submodularity naturally arises. Concept-based summarization (Filatova & Hatzivassiloglou, 2004; Takamura & Okumura, 2009; Riedhammer *et al.* , 2010; Qazvinian *et al.* , 2010) usually maximizes the weighted credit of concepts covered by the summary. Although the authors may not have noticed, their objective functions are also submodular, adding more evidence suggesting that submodularity is natural for summarization tasks. Indeed, let S be a subset of sentences in the document and denote $\Gamma(S)$ as the set of concepts contained in S. The total credit of the concepts covered by S is then

$$f_{\text{concept}}(S) \triangleq \sum_{i \in \Gamma(S)} c_i, \tag{4.23}$$

where c_i is the credit of concept i. This function is known to be monotone submodular (Narayanan, 1997a). Optimization methods used in most of these approaches, however, are either greedy algorithms without scaled cost (e.g., (Filatova & Hatzivassiloglou, 2004; Qazvinian *et al.* , 2010)), or over-complex algorithms (e.g., ILP) that are not usually scalable (e.g., (Takamura & Okumura, 2009; Riedhammer *et al.* , 2010)).

4.6.3 Summarization with a covering constraint

Another perspective is to treat the summarization problem as finding a low-cost subset of the document under the constraint that a summary should cover all (or a sufficient amount of) the information in the document. Formally, this can be expressed as

Problem 13. *Find*

$$S^* \in \operatorname*{argmin}_{S \subseteq V} \sum_{i \in S} c_i \ \textit{subject to:}\ f(S) \geq \alpha,$$

where c_i are the element costs, and set function $f(S)$ measure the information covered by S. When f is submodular, the constraint $f(S) \geq \alpha$ is called a *submodular cover* constraint. When f is monotone submodular, a greedy algorithm that iteratively selects k with minimum $c_k/(f(S \cup \{k\}) - f(S))$ has approximation guarantees (Wolsey, 1982). In particular, when f is integer valued, the greedy solution is within $H(\max_{k \in V} f(\{k\}))$ of the optimal

solution where $H(i)$ is the ith harmonic number. Recent work (Shen & Li, 2010) proposes to model document summarization as finding a minimum dominating set and a greedy algorithm is used to solve the problem. The dominating set constraint is also a submodular cover constraint. Define $\delta(S)$ be the set of elements that is either in S or is adjacent to some element in S. Then S is a dominating set if $|\delta(S)| = |V|$. Note that

$$f_{\text{dom}}(S) \triangleq |\delta(S)|$$

is monotone submodular. The dominating set constraint is then also a submodular cover constraint, and therefore the approaches in (Shen & Li, 2010) are special cases of Problem 13. The solutions found in this framework, however, do not necessarily satisfy a summary's budget constraint. Consequently, a subset of the solution found by solving Problem 13 has to be constructed as the final summary, and the near-optimality is no longer guaranteed. Therefore, solving Problem 9 for document summarization appears to be a better framework regarding global optimality. In the present chapter, our framework is that of Problem 9.

4.6.4 Summarization evaluation

Pyramid Method

A natural way to evaluate the quality of machined generated summaries is to use human evaluations. The Pyramid method (Nenkova & Passonneau, 2004) is one of those manual evaluation metrics, which has been used in DUC and recent TAC summarization track [9]. Basically, the Pyramid method is a content based metric for which human annotators mark *Summary Content Units* (SCUs) in the human-generated summaries, and a weight is computed for each SCU based on how many human-generated summaries include this SCU. Given a machine generated summary, human annotators select SCUs that has been expressed in the summary. Scores are then computed as a ratio of the sum of the weighted SCUs found in the machine summary over a constant consisting of the sum of the weights of the SCUs in an ideal summary. Similar to the concept based approaches (Section 4.6.2), sum of weights of the SCUs found in a summary is monotone submodular, and therefore scores

[9] http://www.nist.gov/tac/2011/Summarization/

used in Pyramid method are monotone submodular. In particular, we have the following theorem.

Theorem 21. *The modified score in Pyramid method is monotone submodular.*

Proof. Denote $\Gamma(S)$ as the set of SCUs contained in summary S, and w_i as the weight for SCU i. Following the modified score defined in (Passonneau *et al.* , 2005), we have

$$f_{\text{pyramid}}(S) = \frac{\sum_{i\in\Gamma(S)} w_i}{\sum_{i\in\Omega_m} w_i} \tag{4.24}$$

where

$$\Omega_m \triangleq \operatorname*{argmax}_{A\subseteq V,|A|=m} \sum_{i\in A} w_i,$$

V is the set of all SCUs, and m is the average number of SCUs found in human summaries. Since the denominator is a constant given the pyramid and human summaries, and the numerator is monotone submodular (similar to the f_{concept} in Eqn 4.23), f_{pyramid} is therefore monotone submodular. □

ROUGE

Automatic evaluation of summary quality is important for the research of document summarization as it avoids the labor-intensive and potentially inconsistent human evaluation. ROUGE (Lin, 2004) is widely used for summarization evaluation and it has been shown that ROUGE-N scores are highly correlated with human evaluation (Lin, 2004). Interestingly, ROUGE-N is monotone submodular, adding further evidence that monotone submodular functions are natural for document summarization.

Theorem 22. *ROUGE-N is monotone submodular.*

Proof. By definition (Lin, 2004), ROUGE-N is the n-gram recall between a candidate summary and a set of reference summaries. Precisely, let S be the candidate summary (a set of sentences extracted from the ground set V), $c_e : 2^V \to \mathbb{Z}_+$ be the number of times n-gram e occurs in summary S, and R_i be the set of n-grams contained in the reference summary

i (suppose we have K reference summaries, i.e., $i = 1, \cdots, K$). Then ROUGE-N can be written as the following set function:

$$f_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K} \sum_{e \in R_i} \min(c_e(S), r_{e,i})}{\sum_{i=1}^{K} \sum_{e \in R_i} r_{e,i}},$$

where $r_{e,i}$ is the number of times n-gram e occurs in reference summary i. Since $c_e(S)$ is monotone modular and $\min(x, a)$ is a concave non-decreasing function of x, $\min(c_e(S), r_{e,i})$ is monotone submodular by Theorem 8. Since summation preserves submodularity, and the denominator is constant, we see that $f_{\text{ROUGE-N}}$ is monotone submodular.

□

Now that we know ROUGE-N is monotone submodular, we can then use Algorithm 14 to directly optimize $f_{\text{ROUGE-N}}$ to get near-optimal summaries if the reference summaries are given. Figure 4.7 shows such oracle experiment results on DUC-05. As we can see, the greedy algorithm even outperforms human in terms of ROUGE-2 recall scores.

However, since the reference summaries are unknown, it is of course impossible to optimize $f_{\text{ROUGE-N}}$ directly. Therefore, some approaches (Filatova & Hatzivassiloglou, 2004; Takamura & Okumura, 2009; Riedhammer *et al.* , 2010) instead define "concepts". Alternatively, we herein propose a class of monotone submodular functions that naturally models the quality of a summary while not depending on an explicit notion of concepts, as we will see in the following section.

4.7 A Class of Submodular Functions for Summarization

Two properties of a good summary are *relevance* and *non-redundancy*. Objective functions for extractive summarization usually measure these two separately and then mix them together trading off encouraging relevance and penalizing redundancy. The redundancy penalty usually violates the monotonicity of the objective functions (Carbonell & Goldstein, 1998; Lin & Bilmes, 2010b). We therefore propose to positively *reward diversity* instead of negatively penalizing redundancy. In particular, we model the summary quality as

$$f(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S), \tag{4.25}$$

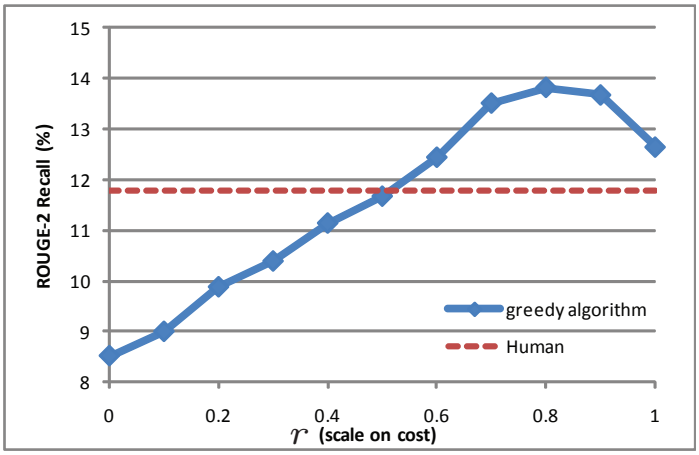


Figure 4.7: Oracle experiments on DUC-05. The red dash line indicates the best ROUGE-2 recall score of human summaries (summary with ID C).

where $\mathcal{L}(S)$ measures the coverage, or "fidelity", of summary set S to the document, $\mathcal{R}(S)$ rewards diversity in S, and $\lambda \geq 0$ is a trade-off coefficient. Note that the above is analogous to the objectives widely used in machine learning, where a loss function that measures the training set error (we measure the coverage of summary to a document), is combined with a regularization term encouraging certain desirable (e.g., sparsity) properties (in our case, we "regularize" the solution to be more diverse). In the following, we discuss how both $\mathcal{L}(S)$ and $\mathcal{R}(S)$ are naturally monotone submodular.

4.7.1 Coverage function

$\mathcal{L}(S)$ can be interpreted either as a set function that measures the similarity of summary set S to the document to be summarized, or as a function representing some form of "coverage" of V by S. Most naturally, $\mathcal{L}(S)$ should be monotone, as coverage improves with a larger summary. $\mathcal{L}(S)$ should also be submodular: consider adding a new sentence into two summary sets, one a subset of the other. Intuitively, the increment when adding a new sentence to the small summary set should be larger than the increment when adding it to the larger set, as the information carried by the new sentence might have already been covered by those sentences that are in the larger summary but not in the smaller summary. This is

exactly the property of diminishing returns. Indeed, Shannon entropy, as the measurement of information, is another well-known monotone submodular function.

There are several ways to define $\mathcal{L}(S)$ in our context. For instance, we could use $\mathcal{L}(S) = \sum_{i \in V, j \in S} w_{i,j}$ where $w_{i,j}$ represents the similarity between i and j. $\mathcal{L}(S)$ could also be facility location objective, i.e., $\mathcal{L}(S) = \sum_{i \in V} \max_{j \in S} w_{i,j}$, as used in (Lin *et al.* , 2009). We could also use $\mathcal{L}(S) = \sum_{i \in \Gamma(S)} c_i$ as used in concept-based summarization, where the definition of "concept" and the mechanism to extract these concepts become important. All of these are monotone submodular.

Alternatively, in this paper we propose the following objective that does not rely on concepts. Let

$$\mathcal{L}(S) = \sum_{i \in V} \min \left\{ \mathcal{C}_i(S), \alpha \, \mathcal{C}_i(V) \right\}, \tag{4.26}$$

where $\mathcal{C}_i : 2^V \rightarrow \mathbb{R}$ is a monotone submodular function and $0 \leq \alpha \leq 1$ is a threshold co-efficient. Firstly, $\mathcal{L}(S)$ as defined in Eqn. 4.26 is a monotone submodular function. The monotonicity is immediate. To see that $\mathcal{L}(S)$ is submodular, consider the fact that $f(x) = \min(x, a)$ where $a \geq 0$ is a concave non-decreasing function, and by Theorem 8, each summand in Eqn. 4.26 is a submodular function, and as summation preserves submodularity, $\mathcal{L}(S)$ is submodular.

Next, we explain the intuition behind Eqn. 4.26. Basically, $\mathcal{C}_i(S)$ measures how similar S is to element i, or how much of i is "covered" by S. Then $\mathcal{C}_i(V)$ is just the largest value that $\mathcal{C}_i(S)$ can achieve. We call i "saturated" by S when $\min\{\mathcal{C}_i(S), \alpha\mathcal{C}_i(V)\} = \alpha\mathcal{C}_i(V)$. When i is already saturated in this way, any new sentence j can not further improve the coverage of i even if it is very similar to i (i.e., $\mathcal{C}_i(S \cup \{j\}) - \mathcal{C}_i(S)$ is large). This will give other sentences that are not yet saturated a higher chance of being better covered, and therefore the resulting summary tends to better cover the entire document.

One simple way to define $\mathcal{C}_i(S)$ is just to use

$$\mathcal{C}_i(S) = \sum_{j \in S} w_{i,j} \tag{4.27}$$

where $w_{i,j} \geq 0$ measures the similarity between i and j. In this case, when $\alpha = 1$, Eqn. 4.26 reduces to the case where $\mathcal{L}(S) = \sum_{i \in V, j \in S} w_{i,j}$. As we will see in Section 4.7.3, having

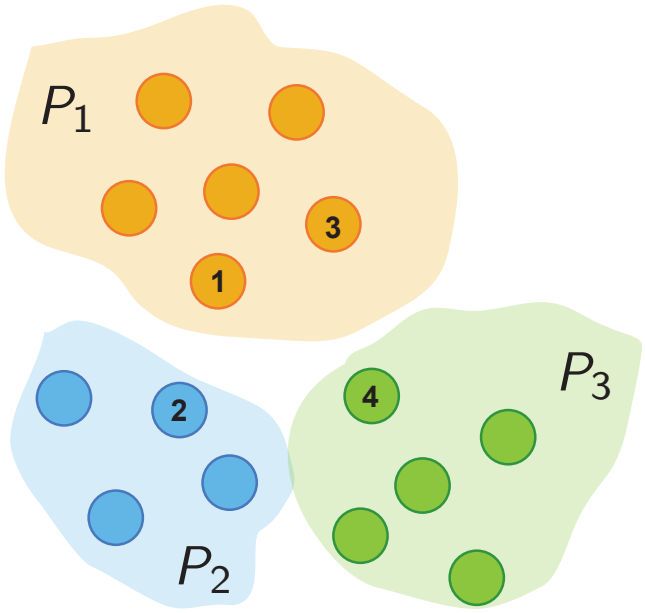


Figure 4.8: Example of diversity rewards.

an α that is less than 1 significantly improves the performance compared to the case when $\alpha = 1$, which coincides with our intuition that using a truncation threshold improves the final summary's coverage.

4.7.2 Diversity reward function

Instead of penalizing redundancy by subtracting from the objective, we propose to reward diversity by adding the following to the objective:

$$\mathcal{R}(S) = \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i \cap S} r_j}. \tag{4.28}$$

where $P_i, i = 1, \cdots K$ is a partition of the ground set V (i.e., $\bigcup_i P_i = V$ and the P_is are disjoint) into separate clusters, and $r_i \geq 0$ indicates the *singleton reward* of i (i.e., the reward of adding i into the empty set). The value r_i estimates the importance of i to the summary. The function $\mathcal{R}(S)$ rewards diversity in that there is usually more benefit to selecting a sentence from a cluster not yet having one of its elements already chosen. As soon as

an element is selected from a cluster, other elements from the same cluster start having diminishing gain, thanks to the square root function. For instance, consider the case where $k_1, k_3 \in P_1$, $k_2 \in P_2$, and $r_{k_1} = 4$, $r_{k_3} = 9$, and $r_{k_2} = 4$ (See Figure 4.7.2 for illustration). Assume k_1 is already in the summary set S. Greedily selecting the next element will choose k_2 rather than k_3 since $\sqrt{13} < 2 + 2$. In other words, adding k_2 achieves a greater reward as it increases the diversity of the summary (by choosing from a different cluster). Note, $\mathcal{R}(S)$ is distinct from $\mathcal{L}(S)$ in that $\mathcal{R}(S)$ might wish to include certain outlier material that $\mathcal{L}(S)$ could ignore.

It is easy to show that $\mathcal{R}(S)$ is submodular by using the composition rule from Theorem 8. The square root is non-decreasing concave function. Inside each square root lies a modular function with non-negative weights (and thus is monotone). Applying the square root to such a monotone submodular function yields a submodular function, and summing them all together retains submodularity. The monotonicity of $\mathcal{R}(S)$ is straightforward. Note, the form of Eqn. 4.28 is similar to structured group norms (e.g., (Zhao *et al.* , 2009)), recently shown to be related to submodularity (Bach, 2010; Jegelka & Bilmes, 2011).

Several extensions to Eqn. 4.28 are discussed next: First, instead of using a ground set partition, intersecting clusters can be used. Second, the square root function in Eqn. 4.28 can be replaced with any other non-decreasing concave functions (e.g., $f(x) = log(1 + x)$) while preserving the desired property of $\mathcal{R}(S)$, and the curvature of the concave function then determines the rate that the reward diminishes. Last, multi-resolution clustering (or partitions) with different sizes (K) can be used, i.e., we can use a mixture of components, each of which has the structure of Eqn. 4.28. A mixture can better represent the core structure of the ground set (e.g., the hierarchical structure in the documents (Celikyilmaz & Hakkani-tür, 2010)). All such extensions preserve both monotonicity and submodularity.

4.7.3 Experiments

The document understanding conference (DUC) (`http://duc.nist.org`) was the main forum providing benchmarks for researchers working on document summarization. The tasks in DUC evolved from single-document summarization to multi-document summarization,

and from generic summarization (2001–2004) to query-focused summarization (2005–2007). As ROUGE (Lin, 2004) has been officially adopted for DUC evaluations since 2004, we also take it as our main evaluation criterion. We evaluated our approaches on DUC data 2003-2007, and demonstrate results on both generic and query-focused summarization. In all experiments, Algorithm 4 was used.

4.7.4 Pre-processing

Documents given in DUC 2003-2007 are all English news articles. For each news article, texts between `<TEXT>` and `</TEXT>` tags were extracted, and news press headers (e.g., AP, Xinhua) at the beginning of each article were removed. Texts between parentheses were also removed. Sentence segmentation was then applied using the maximum entropy sentence boundary detector (Reynar & Ratnaparkhi, 1997). The resulting segmented sentences consist our ground set. In other words, all our summaries were generated by extracting sentences from this ground set.

After segmentation, quotation marks (i.e. “ and ”) were removed, and words were stemmed using the Porter Stemmer. Each sentence was represented using a bag-of-terms vector, where we used context terms up to bi-grams. Similarity between sentence i and sentence j, i.e., $w_{i,j}$, was computed using cosine similarity:

$$w_{i,j} = \frac{\sum_{w \in s_i} \mathrm{tf}_{w,i} \times \mathrm{tf}_{w,j} \times \mathrm{idf}_w^2}{\sqrt{\sum_{w \in s_i} \mathrm{tf}_{w,s_i}^2 \mathrm{idf}_w^2} \sqrt{\sum_{w \in s_j} \mathrm{tf}_{w,j}^2 \mathrm{idf}_w^2}},$$

where $\mathrm{tf}_{w,i}$ and $\mathrm{tf}_{w,j}$ are the numbers of times that w appears in s_i and sentence s_j respectively, and idf_w is the inverse document frequency (IDF) of term w (up to bigram), which was calculated as the logarithm of the ratio of the number of articles that w appears over the total number of all articles in all document clusters (Eqn 4.8). Since cosine similarity is symmetric, we have $w_{i,j} = w_{j,i}, \forall i, j \in V$.

4.7.5 Generic summarization

Summarization tasks in DUC-03 and DUC-04 are multi-document summarization on English news articles. In each task, 50 document clusters are given, each of which consists of 10

documents. For each document cluster, the system generated summary may not be longer than 665 bytes including spaces and punctuation. We used DUC-03 as our development set, and tested on DUC-04 data. We show ROUGE-1 scores[10] as it was the main evaluation criterion for DUC-03, 04 evaluations.

Table 4.8: ROUGE-1 recall (R) and F-measure (F) results (%) on DUC-04. DUC-03 was used as development set.

DUC-04	R	F
$\sum_{i \in V} \sum_{j \in S} w_{i,j}$	33.59	32.44
$\mathcal{L}_1(S)$	39.03	38.65
$\mathcal{R}_1(S)$	38.23	37.81
$\mathcal{L}_1(S) + \lambda \mathcal{R}_1(S)$	**39.35**	**38.90**
Takamura & Okumura (2009)	38.50	-
Wang *et al.* (2009)	39.07	-
Lin & Bilmes (2010b)	-	38.39
Best system in DUC-04 (peer 65)	38.28	37.94

We first tested our coverage and diversity reward objectives separately. For coverage, we use a modular $\mathcal{C}_i(S) = \sum_{j \in S} w_{i,j}$ for each sentence i, i.e.,

$$\mathcal{L}_1(S) = \sum_{i \in V} \min \left\{ \sum_{j \in S} w_{i,j}, \alpha \sum_{k \in V} w_{i,k} \right\}. \tag{4.29}$$

When $\alpha = 1$, $\mathcal{L}_1(S)$ reduces to $\sum_{i \in V, j \in S} w_{i,j}$, which measures the overall similarity of summary set S to ground set V. As mentioned in Section 4.7.1, using such similarity measurement could possibly over-concentrate on a small portion of the document and result in a poor coverage of the whole document. As shown in Table 4.8, optimizing this objective function gives a ROUGE-1 F-measure score 32.44%. On the other hand, when using $\mathcal{L}_1(S)$ with an $\alpha < 1$ (the value of α was determined on DUC-03 using a grid search), a ROUGE-1 F-measure score 38.65% is achieved, which is already better than the best performing system in DUC-04.

[10]ROUGE version 1.5.5 with options: -a -c 95 -b 665 -m -n 4 -w 1.2

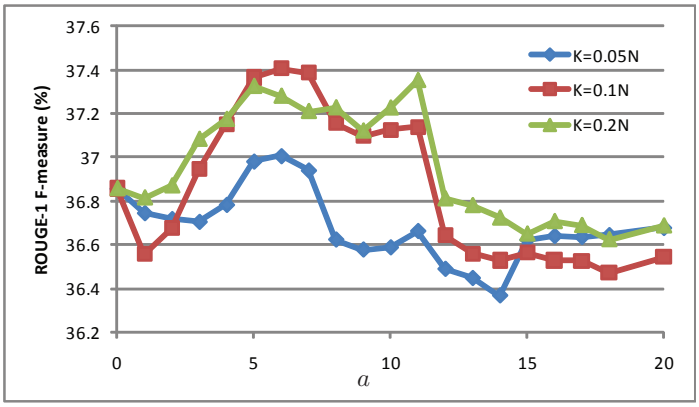


Figure 4.9: ROUGE-1 F-measure scores on DUC-03 when α and K vary in objective function $\mathcal{L}_1(S) + \lambda\mathcal{R}_1(S)$, where $\lambda = 6$ and $\alpha = \frac{a}{N}$.

As for the diversity reward objective, we define the singleton reward as $r_i = \frac{1}{N}\sum_j w_{i,j}$, which is the average similarity of sentence i to the rest of the document. It basically states that the more similar to the whole document a sentence is, the more reward there will be by adding this sentence to an empty summary set. By using this singleton reward, we have the following diversity reward function:

$$\mathcal{R}_1(S) = \sum_{k=1}^{K} \sqrt{\sum_{j \in S \cap P_k} \frac{1}{N} \sum_{i \in V} w_{i,j}}. \tag{4.30}$$

In order to generate $P_k, k = 1, \cdots K$, we used CLUTO[11] to cluster the sentences, where the IDF-weighted term vector was used as feature vector, and a direct K-mean clustering algorithm was used. In this experiment, we set $K = 0.2N$. In other words, there are 5 sentences in each cluster on average. And as we can see in Table 4.8, optimizing the diversity reward function alone achieves comparable performance to the DUC-04 best system.

Combining $\mathcal{L}_1(S)$ and $\mathcal{R}_1(S)$, our system outperforms the best system in DUC-04 significantly, and it also outperforms several recent systems, including a concept-based summarization approach (Takamura & Okumura, 2009), a sentence topic model based system (Wang *et al.* , 2009), and our MMR-styled submodular system (Lin & Bilmes, 2010b). Figure 4.9 illustrates how ROUGE-1 scores change when α and K vary on the development set

[11] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

Table 4.9: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-05, where DUC-05 was used as training set.

DUC-05	R	F
$\mathcal{L}_1(S) + \lambda\mathcal{R}_Q(S)$	**8.38**	**8.31**
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	8.48	8.42
Daumé III & Marcu (2006)	7.62	-
Extr, Daumé *et al.* (2009)	7.67	-
Vine, Daumé *et al.* (2009)	8.24	-

Table 4.10: ROUGE-2 recall (R) and F-measure (F) results on DUC-05 (%). We used DUC-06 as training set. DUC-06 and DUC-07 were used as training sets for objective $\mathcal{L}_1(S) + \sum_\kappa \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$.

DUC-05	R	F
$\mathcal{L}_1(S) + \lambda\mathcal{R}_Q(S)$	7.82	7.72
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	**8.19**	**8.13**
Daumé III & Marcu (2006)	6.98	-
Best system in DUC-05 (peer 15)	7.44	7.43

(DUC-03). As we can see, when $K = 0.1N$ and $\alpha = \frac{5}{N}$, we have the submodular function by maximizing which gives us the best ROUGE-1 F-measure score on DUC-03.

4.7.6 Query-focused summarization

We evaluated our approach on the task of query-focused summarization using DUC 05-07 data. In DUC-05 and DUC-06, participants were given 50 document clusters, where each cluster contains 25 news articles related to the same topic. Participants were asked to generate summaries of at most 250 words for each cluster. For each cluster, a title and a narrative describing a user's information need are provided. The narrative is usually composed of a set of questions or a multi-sentence task description. The main task in DUC-07 is the same as in DUC-06.

In DUC 05-07, ROUGE-2 was the primary criterion for evaluation, and thus we also

Table 4.11: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-06, where DUC-05 was used as training set. DUC-05 and DUC-07 were used as training sets for objective $\mathcal{L}_1(S) + \sum_\kappa \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$.

DUC-06	R	F
$\mathcal{L}_1(S) + \lambda \mathcal{R}_Q(S)$	9.75	9.77
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	**9.81**	**9.82**
Celikyilmaz & Hakkani-tür (2010)	9.10	-
Shen & Li (2010)	9.30	-
Best system in DUC-06 (peer 24)	9.51	9.51

Table 4.12: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-07. DUC-05 was used as training set for objective $\mathcal{L}_1(S) + \lambda \mathcal{R}_Q(S)$. DUC-05 and DUC-06 were used as training sets for objective $\mathcal{L}_1(S) + \sum_\kappa \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$.

DUC-07	R	F
$\mathcal{L}_1(S) + \lambda \mathcal{R}_Q(S)$	12.18	12.13
$\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$	12.38	**12.33**
Toutanova *et al.* (2007)	11.89	11.89
Haghighi & Vanderwende (2009)	11.80	-
Celikyilmaz & Hakkani-tür (2010)	11.40	-
Best system in DUC-07 (peer 15)	12.45	12.29

report ROUGE-2[12] (both recall R, and precision F). Documents were processed as in Section 4.7.5. We used both the title and the narrative as query, where stop words, including some function words (e.g., "describe") that appear frequently in the query, were removed. All queries were then stemmed using the Porter Stemmer.

Note that there are several ways to incorporate query-focused information into both the coverage and diversity reward objectives. For instance, $\mathcal{C}_i(S)$ could be query-dependent in how it measures how much query-dependent information in i is covered by S. Also, the coefficient α could be query and sentence dependent, where it takes larger value when a

[12]ROUGE version 1.5.5 was used with option -n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -d -l 250

sentence is more relevant to query (i.e., a larger value of α means later truncation, and therefore more possible coverage). Similarly, sentence clustering and singleton rewards in the diversity function can also be query-dependent. In this experiment, we explore an objective with a query-independent coverage function ($\mathcal{R}_1(S)$), indicating prior importance, combined with a query-dependent diversity reward function, where the latter is defined as:

$$\mathcal{R}_Q(S) = \sum_{k=1}^{K} \sqrt{\sum_{j \in S \cap P_k} \left(\frac{\beta}{N} \sum_{i \in V} w_{i,j} + (1-\beta) r_{j,Q} \right)},$$

where $0 \leq \beta \leq 1$, and $r_{j,Q}$ represents the relevance between sentence j to query Q. This query-dependent reward function is derived by using a singleton reward that is expressed as a convex combination of the query-independent score ($\frac{1}{N}\sum_{i \in V} w_{i,j}$) and the query-dependent score ($r_{j,Q}$) of a sentence. We simply used the number of terms (up to a bi-gram) that sentence j overlaps the query Q as $r_{j,Q}$, where the IDF weighting is not used (i.e., every term in the query, after stop word removal, was treated as equally important). Both query-independent and query-dependent scores were then normalized by their largest value respectively such that they had roughly the same dynamic range.

To better estimate of the relevance between query and sentences, we further expanded sentences with synonyms and hypernyms of its constituent words. In particular, part-of-speech tags were obtained for each sentence using the maximum entropy part-of-speech tagger (Ratnaparkhi, 1996), and all nouns were then expanded with their synonyms and hypernyms using WordNet (Fellbaum, 1998). Note that these expanded documents were only used in the estimation $r_{j,Q}$, and we plan to further explore whether there is benefit to use the expanded documents either in sentence similarity estimation or in sentence clustering in our future work. We also tried to expand the query with synonyms and observed a performance decrease, presumably due to noisy information in a query expression.

While it is possible to use an approach that is similar to (Toutanova *et al.* , 2007) to learn the coefficients in our objective function, we trained all coefficients to maximize ROUGE-2 F-measure score using the Nelder-Mead (derivative-free) method. Using $\mathcal{L}_1(S) + \lambda \mathcal{R}_Q(S)$ as the objective and with the same sentence clustering algorithm as in the generic summarization experiment ($K = 0.2N$), our system, when both trained and tested on DUC-

05 (results in Table 4.9), outperforms the Bayesian query-focused summarization approach (Daumé III & Marcu, 2006) and the search-based structured prediction approach (Daumé *et al.* , 2009), which were also trained and tested on DUC-05. Note that the system in (Daumé *et al.* , 2009) that achieves its best performance (8.24% in ROUGE-2 recall) is a so called "vine-growth" system, which can be seen as an abstractive approach, whereas our system is *purely* an extractive system. Comparing to the extractive system in (Daumé *et al.* , 2009), our system performs much better (8.38% v.s. 7.67%). More importantly, when trained only on DUC-06 and tested on DUC-05 (results in Table 4.10), our approach outperforms the best system in DUC-05 significantly.

We further tested the system trained on DUC-05 on both DUC-06 and DUC-07. The results on DUC-06 are shown in Table. 4.11. Our system outperforms the best system in DUC-06, as well as two recent approaches (Shen & Li, 2010; Celikyilmaz & Hakkani-tür, 2010). On DUC-07, in terms of ROUGE-2 score, our system outperforms PYTHY (Toutanova *et al.* , 2007), a state-of-the-art supervised summarization system, as well as two recent systems including a generative summarization system based on topic models (Haghighi & Vanderwende, 2009), and a hybrid hierarchical summarization system (Celikyilmaz & Hakkani-tür, 2010). It also achieves comparable performance to the best DUC-07 system. Note that in the best DUC-07 system (Pingali *et al.* , 2007; Jagarlamudi *et al.* , 2006), an external web search engine (Yahoo!) was used to estimate a language model for query relevance. In our system, no such web search expansion was used.

To further improve the performance of our system, we used both DUC-05 and DUC-06 as a training set, and introduced three diversity reward terms into the objective where three different sentence clusterings with different resolutions were produced (with sizes $0.3N, 0.15N$ and $0.05N$). Denoting a diversity reward corresponding to clustering κ as $\mathcal{R}_{Q,\kappa}(S)$, we model the summary quality as $\mathcal{L}_1(S) + \sum_{\kappa=1}^{3} \lambda_\kappa \mathcal{R}_{Q,\kappa}(S)$. As shown in Table 4.12, using this objective function with multi-resolution diversity rewards improves our results further, and outperforms the best system in DUC-07 in terms of ROUGE-2 F-measure score. We also tested this multi-resolution objective (again, with sizes $0.3N, 0.15N$ and$0.05N$) on DUC-05 and DUC-06 (see Table 4.9,4.10,4.11), and observed improvements on DUC-05.

4.8 Conclusion and Discussion

In this chapter, we see submodularity naturally arises in summarization tasks. Not only do many existing automatic summarization methods correspond to submodular function optimization, but also the widely used ROUGE evaluation is closely related to submodular functions. As the corresponding submodular optimization problem can be solved efficiently and effectively, the remaining question is then how to design a submodular objective that best models the task. To address this problem, we introduce a powerful class of monotone submodular functions that are well suited to document summarization by modeling two important properties of a summary, fidelity and diversity. While more advanced NLP techniques could be easily incorporated into our functions (e.g., language models could define a better $\mathcal{C}_i(S)$, more advanced relevance estimations for the singleton rewards r_i, and better and/or overlapping clustering algorithms for our diversity reward), we already show top results on standard benchmark evaluations using fairly basic NLP methods (e.g., term weighting and WordNet expansion), all, we believe, thanks to the power and generality of submodular functions. In Chapter 5, we further generalize this class of submodular functions to mixtures of submodular functions with learning algorithm proposed accordingly, which further improves the performance we have seen in this chapter.

As information retrieval and web search are closely related to query-focused summarization, our approach might be beneficial in those areas as well. For instance, since submodularity has been known (Agrawal *et al.* , 2009; Chapelle *et al.* , 2011) to enhance diversity in ranking, our class of submodular functions might apply to the search results diversification as well.

Chapter 5

LEARNING SUBMODULAR MIXTURES

So far, we have seen how submodular functions are natural as score functions for some NLP problems and how we can leverage properties of submodular function optimization to not only increase the expressive power of score functions, but also avoid imposing prohibitive computation burden on inference. Often, there might be an underlying submodular function that fully characterizes the problem. The function, however, is not usually accessible and cannot be directly evaluated. For example, as we have seen in Chapter 4, ROUGE score for summarization is submodular, but since the human references are not supposed to be available for the task of automatic summarization, we could not directly evaluate this score function when generating machine summaries. Alternatively, we propose a class of submodular functions and evaluate them as *surrogates* to the ROUGE score function. One issue we have not addressed, however, is how good the surrogate submodular function approximates the ROUGE score that we actually want to optimize.

In fact, approximating a submodular function everywhere is not an easy problem. Goemans *et al.* (2009) studied the algorithmic problem of efficiently finding a function which approximates a submodular function at every point of its domain. In particular, the problem studied in (Goemans *et al.* , 2009) is "can one make only polynomial number of queries to an unknown submodular function f and constructs a $\hat{f}$ such that $\hat{f}(S) \leq f(S) \leq g(n)\hat{f}(S)$ where $g : \mathbb{N} \rightarrow \mathbb{R}$?". For some submodular function, the above problem can be solved exactly (i.e. with $g(n) = 1$) and one example of this sort is the graph cut function, where one can completely reconstruct the graph in $O(n^2)$ queries. For more general monotone submodular functions, any algorithm performing a polynomial number of queries must have $g(n) = \Omega(\sqrt{n}/\log n)$; in other words, no algorithm can achieve a factor better than $\Omega(\sqrt{n}/\log n)$ (Goemans *et al.* , 2009). Balcan & Harvey (2011) studied the submodular function learning problem from a learning theory perspective, where given a polynomial

number of samples from a distribution over subsets of some ground set, along with the values of a submodular function at those sample points, the task is to approximate the submodular function within a multiplicative factor at subsequent sample points drawn from the same distribution, with sufficiently high probability. It turns out that this task is extremely hard for general sample distributions: no algorithm can approximate the function to within a factor of $O(n^{\frac{1}{3}})$, even if the algorithm knows the underlying distribution and it can adaptively query the target function at points of its choice (Balcan & Harvey, 2011).

In this chapter, instead of addressing the problem of learning a submodular function with unknown forms or structures, we focus on learning submodular functions with known forms but unknown parameters. In particular, we represent an underlying unknown submodular function as a conical combination of a finite number of submodular functions with known forms. We refer such conical combination of submodular functions as a *submodular mixture*, and each submodular function as a *component* of the mixture.

By using a conic combination, submodularity is preserved, and therefore when a submodular mixture is used as score function, the decoding can still be done efficiently and near-optimally. Moreover, by using a variety of components, the expressive capacity of the score function is further increased, and it might even be possible to well approximate the unknown submodular function of interest (e.g., ROUGE score).

Since learning a submodular function is in general hard, we assume components in the submodular mixtures are given. Indeed, as we will see (Section 5.3), a fairly rich class of monotone submodular functions can be represented as a submodular mixture of components with simple form (e.g., weight matroid rank function as a component, or truncation function as a component, etc.).

The task is then learning the component weights given finite number of samples. In such learning, we inevitably need to use approximate learning since by using a more expressive class of score functions, the learning problem is intractable. On the other hand, we of course want to leverage the rigorous approximation guarantees of submodular optimization such that the approximate learning of submodular mixtures can be bounded in performance in some way. As shown in (Kulesza *et al.* , 2007), if the learning and inference are not compatible, learning could fail even with approximate inference with rigorous guarantees. One

example given in (Kulesza *et al.* , 2007) is perceptron learning with loopy belief propagation (Pearl, 1988), where the assumption of perceptron learning is increasing weights for features in correct labeling will lead to better predation, and the approximate inference, however, can disturb and even invert such assumption, leading to the failure of learning. In general, it is invalid to assume that an arbitrary choice of approximate inference will lead to useful results when the learning method expects exact feedback. We therefore carefully choose the learning algorithm such that it is compatible to our approximate inference (Section 5.4). We moreover theoretically analysis the performance of submodular mixtures with approximate learning (Section 5.5), and show how the best results ever reported on DUC-2004, DUC-2005, DUC-2006 and DUC-2007 can be obtained by learning submodular mixtures (Section 5.6).

5.1 Learning for Structured Prediction

We are given a set of training instances $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ drawn independently from a distribution D over pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is an input space and $\mathcal{Y}$ is the associated output or label space. The problem of interest is to find a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ in a hypothesis space $\mathcal{H}$ such that the risk $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{y}}(h(\boldsymbol{x}))]$ is minimized, where $\ell : \mathcal{Y} \to \mathbb{R}$ is the loss function and $\ell_{\boldsymbol{y}}(\hat{\boldsymbol{y}})$ measures the loss of predicting $\hat{\boldsymbol{y}}$ when the true label is $\boldsymbol{y}$. Usually, simpler model is preferred. Using a regularizer $q : \mathcal{H} \to \mathbb{R}$ that measures the model complexity, the learning problem can then be written as

$$\min_{h\in\mathcal{H}} \left(\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{y}}(h(\boldsymbol{x}))] + q(h)\right).$$

The model complexity term severs to encourage a model with better generalization, and the risk term is typically approximated using *empirical risk*:

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{y}}(h(\boldsymbol{x}))] \approx \frac{1}{T}\sum_{t=1}^{T} \ell_{\boldsymbol{y}}(h(\boldsymbol{x})).$$

Directly optimizing the empirical risk is usually difficult since the loss function is not convex in general. Alternatively, convex loss functions that upperbound the original (zero-one) loss are used as surrogates, and learning is performed using these surrogate convex loss functions. Different choices of the hypothesis space (model) and/or the (surrogate) loss functions lead

to different learning algorithms. For instance, a generative model assigns probability $p(\boldsymbol{x}, \boldsymbol{y})$ for outcome $(\boldsymbol{x}, \boldsymbol{y})$ of $\mathcal{X} \times \mathcal{Y}$, and a typical loss used in learning is the logistic loss, i.e.

$$\ell_{\boldsymbol{y}}(h(\boldsymbol{x})) = -\log(p(\boldsymbol{x}, \boldsymbol{y})).$$

Minimizing the empirical logistic loss is equivalent to maximizing the log likelihood, and the learning algorithm, also known as maximum likelihood training, is widely used for generative models. When priors are assumed for the model parameters (e.g., Dirichlet distribution is often used as the parametric prior for a multinomial distribution) and used as regularizer, the learning is then known as maximum a posterior training.

As mentioned in Chapter 1, NLP problems usually have a complex and structured output space $\mathcal{Y}$, and the decoding problem is often referred as structured prediction problem. For structured prediction, one commonly used model is the conditional random field (Lafferty *et al.* , 2001). In a conditional random field with log-linear models, the conditional probability is modeled as

$$p_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp\left[\boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})\right]}{\sum_{\boldsymbol{y}' \in \mathcal{Y}_{\boldsymbol{x}}} \exp\left[\boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}')\right]}, \tag{5.1}$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is the weight (parameter) vector, and $\boldsymbol{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ is the feature function. And the loss used in learning is again the logistic loss, i.e.,

$$\ell^{\text{CRF}}_{\boldsymbol{y}}(h(\boldsymbol{x})) = -\log(p_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x})).$$

In order to make the learning practical, one must be able to efficiently compute the log normalization constant in Eqn. 5.1, and often structure of $\mathcal{Y}_{\boldsymbol{x}}$ is assumed (e.g., linear chain structure) such that dynamic programming techniques can be used (Lafferty *et al.* , 2001).

Another class of models for structured predictions simply ignores the probabilistic nature. Instead, a score is directly assigned to each $(\boldsymbol{x}, \boldsymbol{y})$ pair without normalization. While the scores are unnormalized, and thus cannot be treated as probabilities, the decoding is done by choosing the highest scoring output, i.e. the hypothesis function has the following form:

$$h(\boldsymbol{x}; \boldsymbol{w}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}), \tag{5.2}$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is the weight vector, and $\boldsymbol{f} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ is the feature function. Two commonly used models of this kind are known as structured perceptrons (Collins, 2002) and structured large margin models (Taskar *et al.* , 2004; Tsochantaridis *et al.* , 2006). In this chapter, we focus on this type of models as we treat the score function as a submodular mixture without normalization in the probability sense.

In the rest of this chapter, we denote

$$\mathcal{Y}_t \triangleq \mathcal{Y}_{\boldsymbol{x}^{(t)}}$$
$$\boldsymbol{f}_t(\boldsymbol{y}) \triangleq \boldsymbol{f}(\boldsymbol{x}^{(t)}, \boldsymbol{y})$$
$$\ell_t(\boldsymbol{y}) \triangleq \ell_{\boldsymbol{y}^{(t)}}(\boldsymbol{y})$$

for simplicity. Note that the output space (ground set) is instant-dependent, and see Section 1.2 for more detailed discussions.

5.1.1 Structured perceptron

The structured perceptron (Collins, 2002) is an extension of the standard perceptron to structured prediction problems. In perceptron learning, while looping through a set of training instances $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$, whenever the predicted

$$\boldsymbol{y}_t^* = \operatorname*{argmax}_{\boldsymbol{y} \in Y_t} \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) \tag{5.3}$$

differs from $\boldsymbol{y}^{(t)}$, we update the weights according to

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) - \boldsymbol{f}_t(\boldsymbol{y}_t^*). \tag{5.4}$$

Intuitively, this weight update serves to bring the weight vector closer to the correct output and further from the incorrect output. To avoid overfitting, weight averaging is often used to produce final weights. A typical perceptron training algorithm is outlined in Algorithm 8.

The actual loss function used in the perceptron algorithm is:

$$\ell_{\boldsymbol{y}}^{\text{perceptron}}(h(\boldsymbol{x})) = \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, h(\boldsymbol{x})) - \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}). \tag{5.5}$$

Learning of Perceptrons is easy to implement given the decoder. Moreover, some theoretical guarantees are available ensuring that the number of errors made can be bounded

Algorithm 8: Averaged structured perceptron learning (Collins, 2002)

Input : $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ and the number of iterations I.

$w = \mathbf{0}, w_a = \mathbf{0}$;

for $i = 1, \cdots, I$ **do**

  for $t = 1, \cdots, T$ **do**

    Inference: $\boldsymbol{y}_t^* = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(t)}}} \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y})$;

    if $\boldsymbol{y}_t^* \neq \boldsymbol{y}^{(t)}$ **then**

      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) - \boldsymbol{f}_t(\boldsymbol{y}_t^*)$;

      $\boldsymbol{w}_a \leftarrow \boldsymbol{w}_a + \boldsymbol{w}$;

Return: the averaged parameters $\frac{1}{TI}\boldsymbol{w}_a$.

(Collins, 2002). One limitation of Perceptrons, however, is that it is only applicable to problems where zero-one loss applies (Eqn 5.5 is a surrogate for zero-one loss). In other words, when judging whether $\boldsymbol{y}_t^*$ differs from $\boldsymbol{y}^{(t)}$, the implicit assumption is that an output is either "correct" or "incorrect", while many problems in NLP, zero-one loss is far from satisfactory in terms of modeling the problem properly. E.g., in extractive document summarization, an output is never "correct": even two humans, when asked to summarize a document, would be unlikely to give identical summaries. In such problems, more sophisticated loss functions are required in order to have the problem well-modeled, and perceptron is not directly applicable.

5.1.2 Maximum margin Markov networks

Following the formalism for the support vector machine for binary classification, Maximum margin Markov network (M^3N) considers learning for structured prediction as a quadratic problem (Taskar *et al.* , 2005b). In particular, M^3N requires that the differences in scores between the true output $\boldsymbol{y}^{(t)}$ and a hypothesis output $\boldsymbol{y}_t^*$ should be at least as much as the value of the loss $\ell_t(\boldsymbol{y}_t^*)$. In other words, M^3N scales the margin to be proportional to the

loss, which gives a learning problem of the following form:

$$\min_{\boldsymbol{w},\xi_t} \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

subject to:

$$\boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) \geq \ell_t(\boldsymbol{y}) - \xi_t, \forall t, \forall \boldsymbol{y} \in Y_t$$

$$\xi_t \geq 0, \forall t.$$

where $\xi_t, t = 1, \cdots, T$ are slack variables and $\lambda \geq 0$ is a parameter that trades off constraint violations and margin maximization. Note that the number of constraints in the above problem could be exponentially large due to the complexity of $\mathcal{Y}$ in structured prediction. Converting the exponential constraint set for each training example into a single nonlinear constraint, we have:

$$\min_{\boldsymbol{w},\xi_t} \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

subject to:

$$\boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) \geq \max_{\boldsymbol{y}\in Y_t} \Big(\boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y})\Big) - \xi_t, \forall t, \forall \boldsymbol{y}$$

$$\xi_t \geq 0, \forall t.$$

Equivalently, we can have the following unconstrained objective function

$$\min_{\boldsymbol{w}} \frac{1}{T}\sum_{t=1}^{T} \left(\left(\max_{\boldsymbol{y}\in Y_t} \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y})\right) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)})\right) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2, \tag{5.6}$$

from which we see that the following loss

$$\ell_{\boldsymbol{y}}^{\text{hinge}}(h(\boldsymbol{x})) = \left(\max_{\boldsymbol{y}'\in\mathcal{Y}_{\boldsymbol{x}}} \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}') + \ell_{\boldsymbol{y}}(\boldsymbol{y}')\right) - \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) \tag{5.7}$$

is actually used. Optimization techniques have been proposed to solve Eqn. 5.6, including those based on exponentiated gradient method (Collins & McAllester, 2004), dual extragadient method (Taskar *et al.* , 2006) and the subgradient descent method (Ratliff *et al.* , 2006a,b).

Note that a *subgradient* of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at $\boldsymbol{w} \in \mathbb{R}^n$ is defined as a vector $\boldsymbol{g}_{\boldsymbol{w}}$ such that

$$f(\boldsymbol{w}') - f(\boldsymbol{w}) \geq \boldsymbol{g}_{\boldsymbol{w}}^\top (\boldsymbol{w}' - \boldsymbol{w}), \forall \boldsymbol{w}' \in \mathbb{R}^n$$

Algorithm 9: Subgradient descent for learning M^3N

Input : $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ and a learning rate sequence $\{\eta_t\}_{t=1}^{T}$.

$w_0 = 0$;

for $t = 1, \cdots, T$ **do**

Loss augmented inference: $\boldsymbol{y}_t^* = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}_t} \boldsymbol{w}_{t-1}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y})$;

Compute the subgradient: $\boldsymbol{g}_t = \lambda \boldsymbol{w}_{t-1} + \boldsymbol{f}_t(\boldsymbol{y}^*) - \boldsymbol{f}_t(\boldsymbol{y}^{(t)})$;

Update the weights: $\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta_t \boldsymbol{g}_t$;

Return: the averaged parameters $\frac{1}{T}\sum_t \boldsymbol{w}_t$.

Subgradients need not be unique, but for a differentiable convex function, its gradient is the unique subgradient. The following property can be used to compute the subgradient of the objective in Eqn 5.6:

Lemma 6 (Shor *et al.* , 1985)**.** *Let f_i be convex functions, $i = 1, \cdots, m$. Then $\phi(x) = \max_{i=1,\cdots,m} f_i(x)$ is convex. A subgradient of f_{i^*} at point x_0 is also a subgradient for ϕ at x_0, where $i^* \in \{i : \phi(x_0) = f_i(x_0)\}$.*

Therefore, subgradient computation for Eqn 5.6 can seen as finding subgradients for

$$\frac{1}{T}\sum_t \left(\boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}_t^*) + \ell_t(\boldsymbol{y}_t^*) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) \right) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

at point $\boldsymbol{w}_t$ where

$$\boldsymbol{y}_t^* \in \underset{\boldsymbol{y} \in Y_t}{\text{argmax}}\, \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y}).$$

With the subgradient computed, subgradient descent methods can then be used to optimize the objective function and Algorithm 9 illustrates one instance of the (online) subgradient method for M^3N learning.

As we can see in Algorithm 9, whether the learning can be done efficiently depends whether the so called *loss augmented inference* (LAI) problem,

$$\max_{\boldsymbol{y}' \in \mathcal{Y}_{\boldsymbol{x}}} \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}') + \ell_{\boldsymbol{y}}(\boldsymbol{y}'), \tag{5.8}$$

can be solved efficiently.

The LAI problem has a term that precisely matches the prediction program whose parameters we are trying to learn but with an additional term corresponding to the loss function. Tractability of LAI is therefore not only depends on the tractability of the prediction problem, but also on the form of loss functions. M^{3}N assumes that the loss function can be decomposed over parts of the output space, with a natural choice of Hamming distance that counts the number of parts in a candidate $\boldsymbol{y}$ differ from the reference $\boldsymbol{y}^{(t)}$. Therefore, although M^{3}N could apply to loss functions other than zero-one loss in theory, in practise it could only apply to loss functions that are decomposable over structures (e.g., Hamming loss).

5.1.3 Structured SVMs

Algorithm 10: Cutting plane algorithm for structured SVM learning (Tsochantaridis *et al.* , 2006)

Input : $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ and the tolerance ϵ.

$W_t \leftarrow \emptyset$ for all $t = 1, \cdots, T$;

repeat

  for $t = 1, \cdots, T$ **do**

    $H(\boldsymbol{y}) \triangleq \left(1 - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y})\right) \ell_t(\boldsymbol{y})$;

    Find the cutting plane: $\boldsymbol{y}_t^* = \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(t)}}} H(\boldsymbol{y})$;

    Determine the value of current slack variable: $\xi_t = \max\{0, \max_{\boldsymbol{y} \in S} H(\boldsymbol{y})\}$;

    if $H(\boldsymbol{y}_t^*) > \xi_t + \epsilon$ **then**

      $W_t \leftarrow W_t \cup \{\boldsymbol{y}_t^*\}$;

      Update $\boldsymbol{w}$ by solving a QP with constraints based on $\bigcup_t W_t$.

until *no* W_t *has changed during iteration*;

Return: $\boldsymbol{w}$.

The structured support vector machines formalism (Tsochantaridis *et al.* , 2006) is very

similar to the M^{3}N formalism. The quadratic problem for learning is:

$$\min_{\boldsymbol{w},\xi_t} \frac{1}{T}\sum_t \xi_t + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

subject to:

$$\boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) \geq 1 - \frac{\xi_t}{\ell_t(\boldsymbol{y})} \forall t, \forall \boldsymbol{y} \in Y_t$$

$$\xi_t \geq 0, \forall t.$$

The difference between M^{3}N and structured SVM formalisms lies in the fact that M^{3}N uses the loss to do "margin rescaling" while structured SVM uses the loss to do "slack rescaling". To address the issue of exponentially many constraints, Tsochantaridis *et al.* (2006) uses cutting plane algorithm to add constraints only when needed. In particular, it starts with an empty set of constraint and adds the most violated constraint into the set in each iteration. The algorithm is outlined in Algorithm 10, and it can be shown that after polynomial number of iterations, the algorithm will converge to a solution within ϵ of the optimal.

The primary disadvantage of structured SVMs for structured prediction is that it is often difficult to optimize due to the fact that the search of cutting plane:

$$\max_{\boldsymbol{y}\in\mathcal{Y}_t} \left(1 - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y})\right) \ell_t(\boldsymbol{y}), \tag{5.9}$$

where the loss appears as a multiplicative term in the objective, is never tractable for many problems of interest. A cutting plane algorithm, on the other hand, can also be applied to M^{3}N formalism with margin rescaling.

5.2 Approximate Learning

Structured prediction problems are those for which the output space $\mathcal{Y}$ is complex and consists of combinatorial structures. To make prediction tractable, assumptions have to be made either on the score functions, or on independences of the parts of $\mathcal{Y}$, or both (see Section 1.1 for details). Similarly, to make discriminative learning for structured prediction tractable, assumptions or approximations have to be made. This is primarily because in discriminative learning, one has to handle the finite but exponentially large output space

as well. For instance, in CRF learning, one has to do summation over the entire output space. In large margin learning, inference (decoding, prediction) itself plays an important role. As we have seen, inference or its variant with loss functions involved is one of the critical ingredients in the learning algorithms for structured perceptrons (Eqn. 5.3), M^3N (Eqn. 5.8), and structured SVM (Eqn. 5.9).

Since some form of inference is a dominant subroutine in many learning algorithms for structured models, it is natural to use good approximate inference techniques to make the learning problem tractable as well.

We refer to learning with approximate inference as *approximate learning*.

Using approximate inference as a drop-in replacement of exact inference in learning, however, could mislead the learning algorithm and result in arbitrary poorly learnt models. This is theoretically analyzed in (Kulesza *et al.* , 2007), where it is pointed out that approximate learning could fail even with an approximate inference method with rigorous approximation guarantees. Two counterexamples are given in (Kulesza *et al.* , 2007). The first counterexample is on perceptron learning with loopy belief propagation (Pearl, 1988). The assumption of perceptron learning is increasing weights for features in correct labeling will lead to better predation. However, when using the approximate inference (loopy belief propagation), such assumption can be disturbed and even inverted, leading to a failure of learning. In the second counterexample, an approximate inference (linear programming relaxation) with performance guarantee is used in the learning of a Markov random field model, which is shown to make simple concept impossible to learn. In general, it is invalid to assume that an arbitrary choice of approximate inference will lead to useful results when the learning method expects exact feedback. Choosing compatible inference and learning procedures is therefore crucial.

When learning submodular mixtures, we inevitably need to use approximate learning since by using more expressive class of score functions that do not necessary required to be decomposable over structures, the exact learning problem is intractable. On the other hand, we of course want to leverage the rigorous approximation guarantees of submodular optimization such that the performance of approximate learning of submodular mixtures can be bounded in some way. One possible way of bounding is to investigate the degree

to which we can approximate the parameters $\boldsymbol{w}$ that would be obtained by exact learning, since the parameters themselves offer little consequence if good prediction cannot be made from them. Alternatively, we concern ourselves from a practical point of view, and focus on the quality of prediction obtained from an approximately learned model. In particular, we seek to bound the risk gap. That is, the difference between the expected loss of predictions from an approximate (but efficient) scheme and from exact (but intractable) methods. I.e., we want to show

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{y}}(h(\boldsymbol{x};\hat{\boldsymbol{w}}))] - \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{y}}(h(\boldsymbol{x};\boldsymbol{w}^*))] \leq B$$

for some constant B where $\hat{\boldsymbol{w}}$ are the parameters learnt with approximate inference and $\boldsymbol{w}^*$ are the parameters learnt with exact inference. Or alternatively, we want to show the risk of the approximately learnt model is bounded by the empirical risk of a model learnt with exact inference.

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{y}}(h(\boldsymbol{x};\hat{\boldsymbol{w}}))] - \sum_{t=1}^{T} \ell_t(h(\boldsymbol{x}^{(t)};\boldsymbol{w}^*)) \leq B(T)$$

for some $B : \mathbb{Z}_+ \to \mathbb{R}_+$ which decreases as the number of training instances, T, grows.

The rest of this chapter is organized as follows. We formally introduce submodular mixture and its learning algorithm in Section 5.3 and Section 5.4 respectively. In Section 5.5, we bound the difference between the expected loss of predictions from the approximately learned submodular mixtures and from the exactly learned submodular mixtures. Application of submodular mixture to document summarization and the corresponding experimental results are given in Section 5.6.

5.3 Submodular Mixtures

We consider hypothesis functions with the following form

$$\begin{aligned} h(\boldsymbol{x};\boldsymbol{w}) &= \operatorname*{argmax}_{\boldsymbol{y}\in\mathcal{Y}_{\boldsymbol{x}}} s(\boldsymbol{x},\boldsymbol{y}) \\ &= \operatorname*{argmax}_{\boldsymbol{y}\in\mathcal{Y}_{\boldsymbol{x}}} \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) \\ &= \operatorname*{argmax}_{\boldsymbol{y}\in\mathcal{Y}_{\boldsymbol{x}}} \sum_i w_i f_i(\boldsymbol{x},\boldsymbol{y}). \end{aligned}$$

where $\boldsymbol{w} \geq \mathbf{0}$ and $f_i : \mathcal{X} \times \mathcal{Y}_{\boldsymbol{x}} \to \mathbb{R}$ is *submodular* on $\mathcal{Y}_{\boldsymbol{x}}$.

We call the score function

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_i w_i f_i(\boldsymbol{x}, \boldsymbol{y}) \tag{5.10}$$

submodular mixture, and each submodular function f_i a *component* of the mixture.

Obviously, $s(\boldsymbol{x}, \boldsymbol{y})$ is also submodular on $\mathcal{Y}_{\boldsymbol{x}}$ since conic combination of submodular functions preserves submodularity.

A fairly rich subclass of submodular functions can actually be presented as submodular mixtures with components being simpler submodular functions.

Mixtures of weighted matroid rank functions

Recall that the rank function for a matroid $\mathcal{M} = (V, \mathcal{I})$ is defined as

$$r_{\mathcal{M}}(S) = \max \{|I| : I \subseteq S, I \in \mathcal{I}\}.$$

The weighted matroid rank function (Calinescu *et al.* , 2007) generalizes the above rank function by assuming there is a non-negative cost c_i for each $i \in V$, and the weighted rank function $g : 2^V \to \mathbb{R}^+$:

$$g(S) = \max \left\{ \sum_{i \in I} c_i : I \subseteq S, I \in \mathcal{I} \right\}. \tag{5.11}$$

A submodular mixture of weighted matroid rank functions can then be defined on different matroids, say $\mathcal{M}_1, \cdots, \mathcal{M}_d$:

$$f(S) = \sum_{i=1}^{d} w_i r_{\mathcal{M}_i} = \sum_{i=1}^{d} w_i \max \left\{ \sum_{j \in I} c_{i,j} : I \subseteq S, I \in \mathcal{I}_j \right\}, \tag{5.12}$$

where $c_{i,j}$ is the cost for i associated with matroid $\mathcal{M}_j = (V, \mathcal{I}_j)$.

A fairly rich subclass of submodular functions can be represented by $f(S)$ above by changing $\mathcal{M}_j$, w_i, and $c_{i,j}$. For example, coverage type functions, canonical examples of submodular functions, can be written as submodular mixtures in the form of Eqn. 5.12. In particular, let a collection of sets be $\{A_i\}_{1,\cdots,|V|}$ on a ground set E, and the set cover

function is known as

$$f(S) = \left| \bigcup_{i \in S} A_i \right|.$$

We can define costs $c_{i,j} = 1$ if A_i contains $j \in E$ and $c_{i,j} = 0$ otherwise. For each $j \in E$, we define a simple uniform matroid, i.e. $\mathcal{M} = (V, \mathcal{I})$ where $\mathcal{I} = \{I | I \subseteq V, |I| \leq 1\}$. Then the weighted rank of matroid $\mathcal{M}_j$ is

$$\max \left\{ \sum_{j \in I} c_{i,j} : I \subseteq S, I \in \mathcal{I} \right\} = \max \left\{ c_{i,j} : j \in S \right\},$$

which simply indicates whether $\bigcup_{i \in S} A_i$ covers element j. Using uniform weights, we have the submodular mixture representing the covering function. I.e.,

$$f(S) = \left| \bigcup_{i \in S} A_i \right| = \sum_{j=1}^{|E|} \max\{c_{i,j} : j \in S\}.$$

Using non-uniform weights, we could similarly represent weighted coverage function (i.e. $f(S) = m\left(\bigcup_{i \in S} A_i\right)$ where $m : 2^E \to \mathbb{R}$ is a modular function) by submodular mixtures.

Mixtures of truncation functions

Define a truncation function $f : 2^V \to \mathbb{R}$ as

$$f(S) = \min \left\{ c(S), \alpha \right\}, \tag{5.13}$$

where $c : 2^V \to \mathbb{R}$ is a modular function and $\alpha \in \mathbb{R}$ is the truncation threshold. It is easy to show that this function is submodular. A modular function can also be written as a truncation function with $\alpha = \infty$.

The canonical set cover functions can also be represented as mixtures of truncation functions. Using the same definition of $c_{i,j}$ as in the mixture of weighted matroid rank function example, we have

$$f(S) = \left| \bigcup_{i \in S} A_i \right| = \sum_{j=1}^{|E|} \min \left\{ \sum_{i \in S} c_{i,j}, 1 \right\}.$$

Moreover, as pointed out in (Stobbe & Krause, 2010), any concave over cardinality function can be decomposed into mixtures of truncation functions, any sum of concave over cardinality functions can be expressed as a sum of a modular function and nonnegative linear combination of truncation functions (i.e., mixtures of truncation functions as a modular function could be the special case of a truncation function).

Actually, the submodular score functions introduced in Section 4.7 can indeed been seen as mixture of submodular functions, with components being either the coverage or the diversity objectives. We will further explore submodular mixture as score function for summarization in Section 5.6.

In sum, by using a rich enough families of submodular components, a submodular mixture could be very expressive, representing a very large family of submodular functions. Therefore, instead of directly learning an unknown submodular function, which is hard as aforementioned, we aim to learn the component weights of a submodular mixture.

5.4 Learning Algorithms for Submodular Mixtures

We follow the maximum margin approach (Taskar *et al.* , 2005b) to learn the component weights $\boldsymbol{w}$. We focus on supervised learning setup, where given a set of training instances $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$, the goal is to find a score function that scores $\boldsymbol{y}^{(t)}$ higher than all other $\boldsymbol{y} \in \mathcal{Y} \setminus \boldsymbol{y}^{(t)}$ by some margin.

Formally, the learning problem with quadratic regularizer is as follows:

$$\begin{aligned}
&\min_{\boldsymbol{w} \geq 0} \frac{1}{T} \sum_{t=1}^{T} \left(\left(\max_{\boldsymbol{y} \in Y_t} \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y}) \right) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) \right) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2 \\
=& \min_{\boldsymbol{w} \geq 0} \frac{1}{T} \sum_{t=1}^{T} r_t(\boldsymbol{w}) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2 \\
=& \min_{\boldsymbol{w} \geq 0} c(\boldsymbol{w}).
\end{aligned}$$

where

$$\begin{aligned}
r_t(\boldsymbol{w}) &\triangleq \max_{\boldsymbol{y} \in \mathcal{Y}_t} \left(\boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y}) \right) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) \\
c(\boldsymbol{w}) &\triangleq \frac{1}{T} \sum_{t=1}^{T} r_t(\boldsymbol{w}) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2.
\end{aligned}$$

Algorithm 11: Projected perceptron learning for submodular mixtures.

Input : $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ and the number of iterations I.

$w = \mathbf{0}, w_a = \mathbf{0}$;

for $i = 1, \cdots, I$ **do**

  for $t = 1, \cdots, T$ **do**

    Approximate inference: $\hat{\boldsymbol{y}}_t \approx \text{argmax}_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}^{(t)}}} \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y})$;

    if $\hat{\boldsymbol{y}}_t \neq \boldsymbol{y}^{(t)}$ **then**

      $\boldsymbol{w} \leftarrow \max\{\mathbf{0}, \boldsymbol{w} + \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) - \boldsymbol{f}_t(\boldsymbol{y}_t^*)\}$;

      $\boldsymbol{w}_a \leftarrow \boldsymbol{w}_a + \boldsymbol{w}$;

Return: the averaged parameters $\frac{1}{TI}\boldsymbol{w}_a$.

Two issues need to be noted when applying traditional learning algorithms to learn submodular mixtures. First, the component weights are required to be non-negative. Second, we only do approximate inference and infer the approximately highest scored hypothesis. With these two issues in mind, we have the modified perceptron algorithm, cutting plane method, and subgradient descent method for submodular mixture learning in Algorithms 11, 12, and 13, respectively, in which max of two vectors takes dimension-wise maximum, i.e. $\max\{\boldsymbol{a}, \boldsymbol{b}\} = (\max(a_1, b_1), \cdots, \max(a_n, b_n))$ where $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$.

For the first issue, we simply project the weights to the non-negative region whenever doing the updates. Such extra projection step might violate the correctness or the convergence of the original algorithms. Fortunately, for the three algorithms mentioned above, it is easy to show that updates followed by projection onto a non-negative region do not affect the convergence or correctness of the algorithm. Basically, when a point is projected onto the convex set C, it is moved closer to every point in C, including the optimal points. I.e.,

$$\|\boldsymbol{w}_t - \boldsymbol{w}^*\| = \|P(\boldsymbol{v}_t) - \boldsymbol{w}^*\| \leq \|\boldsymbol{v}_t - \boldsymbol{w}^*\|$$

where P is Euclidean projection on C, $\boldsymbol{v}_t$ is the point before projection, and $\boldsymbol{w}^*$ is the optimal point.

Algorithm 12: Cutting plane algorithm for learning submodular mixtures.

Input : $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ and the tolerance ϵ.

$W_t \leftarrow \emptyset$ for all $t = 1, \cdots, T$;

repeat

  for $t = 1, \cdots, T$ **do**

    $H(\boldsymbol{y}) \triangleq \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \ell_t(\boldsymbol{y})$;

    Find the approximate cutting plane: $\hat{\boldsymbol{y}}_t \approx \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{Y}_t} H(\boldsymbol{y})$;

    Determine the value of current slack variable: $\xi_t = \max\{0, \max_{\boldsymbol{y} \in S} H(\hat{\boldsymbol{y}}_t)\}$;

    if $H(\hat{\boldsymbol{y}}_t > \xi_t + \epsilon$ **then**

      $W_t \leftarrow W_t \cup \{\boldsymbol{y}_t^*\}$;

      Update $\boldsymbol{w}$ by solving a QP with constraints based on $\bigcup_t W_t$.

until *no* W_t *has changed during iteration*;

Return: $\boldsymbol{w}$.

Algorithm 13: Projected subgradient descent for learning submodular mixtures.

Input : $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{T}$ and a learning rate sequence $\{\eta_t\}_{t=1}^{T}$.

$w_0 = 0$;

for $t = 1, \cdots, T$ **do**

  Approximate loss augmented inference: $\hat{y}_t \approx \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{Y}_t} \boldsymbol{w}_{t-1}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y})$;

  Compute the subgradient: $\boldsymbol{g}_t = \lambda \boldsymbol{w}_{t-1} + \boldsymbol{f}_t(\boldsymbol{y}^*) - \boldsymbol{f}_t(\boldsymbol{y}^{(t)})$;

  Update the weights with projection: $\boldsymbol{w}_t = \max(\boldsymbol{0}, \boldsymbol{w}_{t-1} - \eta_t \boldsymbol{g}_t)$;

Return: the averaged parameters $\frac{1}{T}\sum_t \boldsymbol{w}_t$.

For the second issue, we need to further analyze whether using good approximate inference would lead to good approximate learning. Before doing so, we note that there are two types of approximation inference algorithms, namely undergenerating and overgenerating approximations.

Consider a maximization problem

$$\max_{\boldsymbol{y} \in \mathcal{Y}} f(\boldsymbol{y}) \triangleq f^*.$$

Undergenerating approximation algorithm always find a solution $\boldsymbol{y} \in \mathcal{Y}$ such that $f(\boldsymbol{y}) \leq f^*$, while overgenerating approximation algorithm always find a solution $\boldsymbol{y} \in \bar{\mathcal{Y}} \supseteq \mathcal{Y}$ such that $f(\boldsymbol{y}) \geq f^*$. The greedy algorithm (Algorithm 4) or loopy belief propagation (Pearl, 1988) are instances of undergenerating approximation algorithms. Relaxation methods, e.g. linear programming relaxation, are overgenerating approximation algorithms. Note that undergenerating algorithms usually produces solution that is within the feasible region of the problem. Overgenerating algorithms, on the other hand, generate solutions that might belong to a superset of the feasible region. Therefore, solutions found by overgenerating algorithms sometimes need to be mapped back to the feasible region (e.g., rounding of linear programing produced solutions) in order to produce a feasible solution, during which the approximation guarantee no longer holds in some cases (Ravikumar *et al.* , 2008). For learning submodular mixtures, we are particularly interested in undergenerating algorithms since the greedy algorithm, one of the undergenerating algorithms, offers near-optimal solutions for submodular maximization with certain (e.g., cardinality, budget, and matroid) constraints (e.g., Theorem 9, Theorem 14).

Again, from a practical point of view, we focus on the quality of prediction obtained from an approximately learned model. I.e., we seek to bound the risk (expected loss) of predictions made by the approximately learned models. A nice bound shall be on the risk gap, that is, the difference between the expected loss of predictions from an approximate (but efficient) scheme and from exact (but intractable) methods. Table 5.1 summaries the results (some of which are new in this thesis) of generalization bounds for approximate learning using perceptron, cutting-plane and subgradient descent algorithms. Generalization bounds for approximate learning with cutting-plane algorithms, with either undergenerating or

Table 5.1: Summary of approximate learning performance.

Approximate algorithms		Risk bound?
Perceptron	undergenerating	
	overgenerating	
Cutting-plane	undergenerating	Yes (Finley & Joachims, 2008)
	overgenerating	Yes (Finley & Joachims, 2008)
Subgradient	undergenerating	Yes (this thesis, Theorem 23)
	overgenerating	Yes (Martins *et al.* , 2009b; Kulesza, 2009)

overgenerating inferences, have been shown in (Finley & Joachims, 2008); the analysis, however, is only on training with a single example, and the risks bounds, rather than bounding w.r.t. the risk under exact learning, are data-depend (actually, they depend on the single training example assumed). For subgradient descent methods, generalization analysis is available, but only for overgenerating cases, in (Martins *et al.* , 2009b; Kulesza, 2009). As far as we know, no generalization analyses are available for approximate learning with perceptrons, or with undergenerating subgradient methods. We fill this gap and offer risk bounds for approximate learning with undergenerating subgradient methods w.r.t the expected risk under exact learning.

5.5 Theoretical Analysis

Definition 9. *Given a class of functions and a maximization problem* $\max_{\boldsymbol{y}\in\mathcal{Y}} f(\boldsymbol{y}) \triangleq f^*$*, we call an (undergenerating) algorithm a* ρ*-approximate algorithm if it finds a solution* $\boldsymbol{y}' \in \mathcal{Y}$ *such that* $f(\boldsymbol{y}') \geq \rho f^*$*, where* $0 \leq \rho \leq 1$*.*

In Algorithm 13, one need to solve the so-called *loss augmented inference* (LAI):

$$\boldsymbol{y}^* \in \operatorname*{argmax}_{\boldsymbol{y}\in\mathcal{Y}_t} \boldsymbol{w}_{t-1}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y}).$$

Note that if ℓ is modular (e.g. hamming loss) or submodular (e.g. see Section 5.6.2), the ρ-approximate inference algorithm can also apply to this loss augmented inference to find a near-optimal solution efficiently. However, when an approximate inference algorithm is used in a learning algorithm, a good approximation of the score might not be sufficient, and it is

possible that the learning can fail even with rigorous approximate guarantees (Kulesza *et al.* , 2007). On the other hand, Ratliff *et al.* (2006b) show that the subgradient algorithm is robust under approximate settings, and the empirical risk experienced during training with approximate subgradients can be bounded in an online setting.

In particular, we have Lemma 7 where $\boldsymbol{w}^*$ is the optimal solution of the convex learning problem, and a vector $\boldsymbol{g}$ is called a γ-subgradient of f at $\boldsymbol{w}$ if for all $\boldsymbol{w}'$, $f(\boldsymbol{w}') \geq f(\boldsymbol{w}) + \boldsymbol{g}^\top(\boldsymbol{w}' - \boldsymbol{w}) - \gamma f(\boldsymbol{w})$ where $0 \leq \gamma \leq 1$.

Lemma 7 (Ratliff *et al.* , 2006b)**.** *Assume $r(\boldsymbol{w})$ is upper-bounded by 1. Let $\hat{\boldsymbol{w}}_t$ be the solution produced at each iteration of Algorithm 13 with γ-approximate subgradient. For $\lambda = \frac{\sqrt{1+\log T}}{\|\boldsymbol{w}^*\|\sqrt{T}}$, $\eta_t = \frac{1}{\lambda t}$, and $S(T) = \|\boldsymbol{w}^*\|\sqrt{T(1+\log T)}$,*

$$\frac{1}{T}\sum_{t=1}^{T} \ell_t(h(\boldsymbol{x}^{(t)}; \hat{\boldsymbol{w}}_t)) \leq \frac{1}{1-\gamma}\left(\frac{1}{T}\sum_{t=1}^{T} r_t(\boldsymbol{w}^*) + \frac{S(T)}{T}\right).$$

Note that a ρ-approximate LAI does not necessary imply any γ-subgradient because the approximate ratio does not apply to the term $-\boldsymbol{w}^\top f_t(\boldsymbol{y}^{(t)})$.

To see that, let $\hat{\boldsymbol{y}}_t$ be the solution found by ρ-approximate LAI, i.e.

$$\boldsymbol{w}_{t-1}^\top \boldsymbol{f}_t(\hat{\boldsymbol{y}}_t) + \ell_t(\hat{\boldsymbol{y}}_t) \geq \rho\left(\boldsymbol{w}_{t-1}^\top \boldsymbol{f}_t(\boldsymbol{y}_t^*) + \ell_t(\boldsymbol{y}_t^*)\right)$$

and let

$$\begin{aligned} c_t(\boldsymbol{w}) &= \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^*) + \ell_t(\boldsymbol{y}^*) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2, \\ \hat{c}_t(\boldsymbol{w}) &= \boldsymbol{w}^\top \boldsymbol{f}_t(\hat{\boldsymbol{y}}_t) + \ell_t(\hat{\boldsymbol{y}}_t) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2, \end{aligned}$$

we have

$$\begin{aligned} c_t(\boldsymbol{w}) &\geq \hat{c}_t(\boldsymbol{w}) \\ &\geq \rho c_t(\boldsymbol{w}) - (1-\rho)\boldsymbol{w}^\top f_t(\boldsymbol{y}^{(t)}) + (1-\rho)\frac{\lambda}{2}\|\boldsymbol{w}\|^2 \end{aligned}$$

and we compute an approximate subgradient using $\hat{\boldsymbol{y}}_t$:

$$\begin{aligned} \hat{\boldsymbol{g}}_t &= \nabla_{\boldsymbol{w}} \hat{c}_t(\boldsymbol{w}) \\ &= \boldsymbol{f}_t(\hat{\boldsymbol{y}}_t) - \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \lambda \boldsymbol{w} \end{aligned}$$

therefore, for any $\boldsymbol{w}'$, we have

$$\begin{aligned}\hat{\boldsymbol{g}}_t^\top(\boldsymbol{w}'-\boldsymbol{w}) &\le \hat{c}_t(\boldsymbol{w}') - \hat{c}_t(\boldsymbol{w}) \\ &\le c_t(\boldsymbol{w}') - \rho c_t(\boldsymbol{w}) + (1-\rho)\boldsymbol{w}^\top f_t(\boldsymbol{y}^{(t)}) - (1-\rho)\frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\ &= c_t(\boldsymbol{w}') - c_t(\boldsymbol{w}) + (1-\rho)c_t(\boldsymbol{w}) + C_t(\boldsymbol{w})\end{aligned}$$

and we see $\hat{\boldsymbol{g}}_t$ does not necessary to be a $(1-\rho)$-subgradient of c_t at $\boldsymbol{w}$ due to $C_t(\boldsymbol{w})$.

To analysis the actual impact of ρ-approximate LAI in the learning procedure when compared with the exact formulation, we provide risk bounds for the approximate learner in Theorem 23.

Theorem 23. *Assume $w_i, f_i, i = 1, \cdots, M$ are all upper-bounded by 1, $r(\boldsymbol{w}) \le B$, and $\|\boldsymbol{g}_t\| \le G$. Let $\hat{\boldsymbol{w}}$ be the solution returned by Algorithm 13 using ρ-approximate LAI with learning rate $\eta_t = \frac{2}{\lambda t}$ and $\lambda = \frac{G}{M}\sqrt{\frac{2(1+\log T)}{T}}$. Then for any $\delta > 0$ with probability at least $1-\delta$,*

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{y}}(h(\boldsymbol{x};\hat{\boldsymbol{w}}))] \le \frac{1}{\rho}\left(\frac{1}{T}\sum_{t=1}^{T} r_t(\boldsymbol{w}^*)\right) + S(T),$$

where

$$S(T) = \frac{MG}{\rho}\sqrt{\frac{2(1+\log T)}{T}} + B\sqrt{\frac{2}{T}\log\frac{1}{\delta}} + \frac{1-\rho}{\rho}M$$

Proof. Let

$$r(\hat{\boldsymbol{w}}) = \hat{\boldsymbol{w}}^\top \boldsymbol{f}(\boldsymbol{x}, h(\boldsymbol{x};\hat{\boldsymbol{w}})) + \ell_{\boldsymbol{y}}(h(\boldsymbol{x};\hat{\boldsymbol{w}})) - \hat{\boldsymbol{w}}^\top \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}).$$

Following (Martins *et al.* , 2009b), we adapt a result in (Cesa-Bianchi *et al.* , 2001) to get for any δ, with probability at least $1-\delta$

$$\mathbb{E}[r(\hat{\boldsymbol{w}})] \le \frac{1}{T}\sum_{t=1}^{T} r_t(\boldsymbol{w}_t) + B\sqrt{\frac{2}{T}\log\frac{1}{\delta}}. \tag{5.14}$$

note that $r(\hat{\boldsymbol{w}}) \ge \ell_{\boldsymbol{y}}(h(\boldsymbol{x};\hat{\boldsymbol{w}}))$ by the definition of h, it suffices to bound the right hand side. Since $c_t(\boldsymbol{w})$ is $\frac{\lambda}{2}$-strongly convex, we have

$$c_t(\boldsymbol{w}^*) \ge c_t(\boldsymbol{w}) + \nabla_{\boldsymbol{w}} c_t(\boldsymbol{w}^* - \boldsymbol{w}) + \frac{\lambda}{4}\|\boldsymbol{w}^* - \boldsymbol{w}\|^2.$$

when an approximate gradient $\hat{\boldsymbol{g}}_t$ is generated with ρ-approximate LAI, we have

$$c_t(\boldsymbol{w}^*) \geq c_t(\boldsymbol{w}_t) + \hat{\boldsymbol{g}}_t^\top(\boldsymbol{w}^* - \boldsymbol{w}_t) + \frac{\lambda}{4}\|\boldsymbol{w}^* - \boldsymbol{w}_t\|^2 - (1-\rho)c_t(\boldsymbol{w}_t) - C_t(\boldsymbol{w}_t).$$

Rearrange to get

$$\rho c_t(\boldsymbol{w}_t) - c_t(\boldsymbol{w}^*) - C_t(\boldsymbol{w}_t) \leq \hat{\boldsymbol{g}}_t^\top(\boldsymbol{w}_t - \boldsymbol{w}^*) - \frac{\lambda}{4}\|\boldsymbol{w}^* - \boldsymbol{w}_t\|^2 \tag{5.15}$$

We upper-bound $\hat{\boldsymbol{g}}_t^\top(\boldsymbol{w}^* - \boldsymbol{w}_t)$. Using the update for $\boldsymbol{w}_{t+1}$, we get

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2 \leq \|\boldsymbol{w}_t - \eta_{t+1}\hat{\boldsymbol{g}}_t - \boldsymbol{w}^*\|^2,$$

and hence

$$\hat{\boldsymbol{g}}^\top(\boldsymbol{w}_t - \boldsymbol{w}^*) \leq \frac{\|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2}{2\eta_{t+1}} + \frac{\eta_{t+1}}{2}G^2.$$

Using above and the fact that $\eta_t = \frac{2}{\lambda t}$, summing Eq. 5.15 from $t = 1, \cdots, T$, we have

$$\begin{aligned}
\sum_{t=1}^{T}(\rho c_t(\boldsymbol{w}_t) - c_t(\boldsymbol{w}^*) - C_t(\boldsymbol{w}_t))) &\leq \sum_{t=1}^{T}\|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2\left(\frac{1}{2\eta_{t+1}} - \frac{1}{2\eta_t} - \frac{\lambda}{4}\right) + \frac{G^2}{2}\sum_{t=1}^{T}\eta_{t+1} \\
&= 0 + G^2\sum_{t=1}^{T}\frac{1}{\lambda(t+1)} \\
&\leq \frac{G^2}{\lambda}(1 + \log T),
\end{aligned}$$

which gives us

$$\begin{aligned}
\rho\frac{1}{T}\sum_{t=1}^{T} r_t(\boldsymbol{w}_t) \leq & \\
&\frac{1}{T}\sum_{t=1}^{T} r_t(\boldsymbol{w}^*) + \frac{\lambda}{2}\left(\|\boldsymbol{w}^*\|^2 - \frac{1}{T}\sum_{t=1}^{T}\|\boldsymbol{w}_t\|^2\right) + \frac{1-\rho}{T}\sum_{t=1}^{T}\boldsymbol{w}_t^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \frac{G^2}{\lambda T}(1 + \log T) \\
&\leq \frac{1}{T}\sum_{t=1}^{T} r_t(\boldsymbol{w}^*) + \frac{M^2\lambda}{2} + \frac{G^2}{\lambda T}(1 + \log T) + M(1-\rho)
\end{aligned}$$

Combining above with Eq. 5.14 and choosing $\lambda = \frac{G}{M}\sqrt{\frac{2(1+\log T)}{T}}$, we have the theorem. □

Note that there are three terms in $S(T)$, while the first two terms vanish as $T \to \infty$, the third term does not. Therefore, the additional risk incurred due to the use of ρ-approximate

LAI for learning is $(1-\rho)(R^*+M)/\rho$, where $R^* = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} r_t(\boldsymbol{w}^*)$. Thus, the better (larger ρ) the approximation, the less the additional risk there will be. And when exact inference is used ($\rho = 1$), the additional risk shrinks to zero.

In practice, as the approximation factor of a greedy algorithm on submodular maximization is usually very close to 1 (Lin & Bilmes, 2010b), one could expect very little additional risk when using Algorithm 13 with approximate inference on sums of submodular functions.

To the best of our knowledge, Theorem 23 is the first approximate learning bound for subgradient algorithms with undergenerating (greedy) inference.

5.6 Learning Submodular Mixtures for Summarization

5.6.1 Submodular components for summarization

In Chapter 4, we have seen how submodularity naturally arises in the task of document summarization, and by designing a class of submodular functions, we achieved better than other results ever reported on several standard benchmark tasks. In this subsection, we generalize the class of submodular functions introduced in Section 4.7 and craft a variety of submodular components that are useful for document summarization. By learning the submodular mixture with these components, we obtain even better results than those reported in Section 4.7.3, and these results are, in fact, the best ever reported as far as we know.

Diversity component

Given a partition $\{P_k\}_{k=1,\cdots,K}$ of the ground set V, we define diversity component as

$$f_{\text{diversity}}(S) = \frac{\sum_{k=1}^{K}\left(\sum_{i\in S\cap P_k} r_i\right)^{\alpha}}{\sum_{k=1}^{K}\left(\sum_{i\in P_k} r_i\right)^{\alpha}}, \tag{5.16}$$

where $0 < \alpha \leq 1$ is the curvature and $r_i \in [0,1]$ is the singleton reward of element $i \in V$. This function is similar to the diversity reward function in Eqn. 4.28 except that it is normalized. In fact, when $\alpha = 0.5$, $f_{\text{diversity}}(S) = \mathcal{R}(S)/\mathcal{R}(V)$ where $\mathcal{R}$ is the diversity reward function defined in Eqn. 4.28. Therefore, the diversity component models the diversity of a summary set S, by diminishing the benefit of choosing elements from the same cluster.

A variety of diversity components can be produced by varying the curvature α, the clusterings (partition), or the singleton rewards.

Clustered facility location component

Given a partition $\{P_k\}_{k=1,\cdots,K}$ of the ground set V, we define clustered facility-location like component as

$$f_{\text{c-facility}}(S) = \frac{1}{K}\sum_{k=1}^{K}\max_{i\in S\cap P_k} r_i, \tag{5.17}$$

where $r_i \in [0,1]$ is the singleton reward of element $i \in V$. This function has a similar form to the well known submodular function, facility location function, but defined on a partition of the ground set. We thus call it clustered facility location function. If a summary contains multiple elements from a same cluster, the element with largest singleton reward will be regarded as the "representative" of this cluster, and only the reward of this representative will be counted into the final score. This again diminishes returns of choosing elements from the same cluster and therefore $f_{\text{c-facility}}$ is submodular.

A variety of clustered facility location components can be produced by changing the clusterings (partition), or the singleton rewards.

Fidelity component

Given a ground set V, we define fidelity component as

$$f_{\text{fidelity}}(S) = \frac{1}{|V|}\sum_{i\in V}\min\left\{\frac{\mathcal{C}_i(S)}{\mathcal{C}_i(V)}, \beta\right\}, \tag{5.18}$$

where $0 < \beta \leq 1$ is the threshold and $\mathcal{C}_i : 2^V \to \mathbb{R}$ is a *monotone* submodular function modeling how S covers the information contained in i. This function is the normalized version of the coverage function defined in Eqn. 4.26. Basically, the saturation threshold controls how much information needed to be covered for an element; once $\mathcal{C}_i(S)$ is large enough such that the ratio of it over the its largest possible value ($\mathcal{C}_i(V)$) is over the saturation threshold, covering more information in i does not help increasing the function value. Therefore, a larger value of f_{fidelity} tends to have more $i \in V$ well covered (to the extent controlled by β).

A variety of fidelity components can be generated by varying the value of β, or the types of $\mathcal{C}_i$ functions.

5.6.2 Loss function in summarization

One problem of applying large margin approach describe above to NLP is that the cost (loss) functions at evaluation time are not always well-defined and usually require human annotations. In the document summarization task, the human generated summaries are sometimes abstractive, and therefore even with human summarises, exact labels for training automatic extractive summarizers are not always available. On the other hand, the widely accepted evaluation criteria for summarization is ROUGE score, which is basically a submodular function that counts n-gram recall rate over human summaries.

Let S be the candidate summary (a set of sentences extracted from the ground set V), $c_e : 2^V \to \mathbb{Z}_+$ be the number of times n-gram e occurs in summary S, and R_i be the set of n-grams contained in the reference summary i (suppose we have K reference summaries, i.e., $i = 1, \cdots, K$). Then ROUGE-N (Lin, 2004) can be written as the following set function:

$$f_{\text{ROUGE-N}}(S) \triangleq \frac{\sum_{i=1}^{K} \sum_{e \in R_i} \min(c_e(S), r_{e,i})}{\sum_{i=1}^{K} \sum_{e \in R_i} r_{e,i}},$$

where $r_{e,i}$ is the number of times n-gram e occurs in reference summary i.

$f_{\text{ROUGE-N}}(S)$ is submodular (as shown in Theorem 22) but cannot be directly used as loss function since it basically measures "accuracy" rather than loss.

An alternative is to use $1 - f_{\text{ROUGE-N}}(S)$ as loss function, which is, however, supermodular. Note that in order to have the risk of the approximated learned model bounded, performance guarantees are required for the approximation algorithms used in inference, or loss augmented inference. When using $1 - f_{\text{ROUGE-N}}$, which is supermodular as a loss function, the objective function in the loss augmented inference (Eqn. 5.8) is at most supermodular. On the other hand, we want to use a submodular mixture as the score function to best match the diminishing return property that naturally arises in the task of summarization. The resulting objective function for the loss augmented inference is then a submodular function plus a supermodular function. While an approximate algorithm (e.g., submodular-supermodular procedure (Narasimhan & Bilmes, 2005b)) is available to

approximately optimize the sum of a submodular function and a supermodular function, performance guarantees usually do not exist for these algorithms. In fact, any set function can be represented as sum of submodular and supermodular functions. It is unlikely that a polynomial time algorithm can solve an optimization problem for any set functions near-optimally. Therefore, using one minus ROUGE as the loss function, although intuitively sounded, is not algorithmically well-supported: the greedy algorithm no longer supplies near-optimal solution when applying to the non-submodular objective, and the risk bound shown in Theorem 23 no longer holds.

To address this issue, we propose a ROUGE-like loss function that measures the "complement recall":

$$\ell_{\text{ROUGE}}(S) \triangleq \frac{\sum_{e\in\bar{R}} \min(c_e(S), r_e)}{\sum_{e\in\bar{R}} r_e}, \tag{5.19}$$

where

$$\bar{R} = N \backslash \bigcup_i R_i, \tag{5.20}$$

N is the set of all the n-grams occur in the documents, and r_e is the number of times n-gram e occurs in the documents. Instead of counting with respect to human summary counts, it counts the n-gram overlaps of a candidate summary S to the *complement* of human summaries. In particular, given the ground set, the set of n-grams contained in the ground set is also given, i.e. N is fixed. Thus $\bar{R}$ is basically the n-grams that are *not* covered by any human references.

Intuitively, we want a summary S to cover as many reference n-grams as possible such that it would get a high ROUGE-score; this is equivalent to having S overlap as little as possible with the n-grams that are *not* in human references. In this sense, ℓ_{ROUGE} measures the portion of how many n-grams in the complement of the reference ngrams set are covered (i.e., "complement recall"), and when comparing summaries with the same size, the smaller ℓ_{ROUGE} is, the better. The extreme case is that the human reference itself would have ℓ_{ROUGE} equal to 0.

Obviously, a summary that is empty would also have ℓ_{ROUGE} equal to 0 but it is not a good summary. It is worth noting that ℓ_{ROUGE} only makes sense when comparing summaries that are close to the same budget. Fortunately, most summarization algorithms would try to consume every bit of the budget and contain as much information as possible under the budget constraint. For summaries produced in this way, ℓ_{ROUGE} offers a fair indicator of their quality: the smaller the loss value is, the larger reference n-gram overlaps there are, and therefore the better the summary is. Since the greedy nature of our algorithm (Algorithm 4) makes it always output summary candidates that closely meet the budget, ℓ_{ROUGE} can sever as a loss function for learning submodular mixtures for summarization tasks.

The major advantage of using ℓ_{ROUGE} as a loss function is algorithmic. Note that all the theoretical analysis of submodular learning introduced in Section 5.5 relies on the fact that a ρ-approximation algorithm is available for the loss augmented information. Recall that the objective to optimize in LAI is the sum of a score function and the corresponding loss:

$$s(\boldsymbol{x}, \boldsymbol{y}) + \ell_t(\boldsymbol{y}).$$

Since we use submodular score function for summarization, the above objective will be submodular if the loss function is submodular. Fortunately, similar to $f_{\text{ROUGE-N}}$, the proposed loss for summarization, ℓ_{ROUGE}, is also monotone submodular. Therefore, the LAI in submodular mixture learning for summarization is exactly the budgeted submodular maximization problem (Problem 4), and efficient and near-optimal algorithms (e.g., Algorithm 4) are then available. Consequently, all the theoretical analyses in Section 5.5 apply.

Not only theoretical well sounded, using the ROUGE-like loss function proposed here moreover empirically outperforms directly using $1 - f_{\text{ROUGE}}$ as a loss functions, as we will see in Section 5.6.4.

5.6.3 Related work

Recently, Sipos *et al.* (2012) studied large margin learning of submodular score functions for extractive document summarization. In particular, two types of submodular score functions

were considered. The first one is the submodular function that we proposed in (Lin *et al.* , 2009; Lin & Bilmes, 2010b), which is based on inter-sentence similarity (f_{penalty} in Eqn. 4.5), and the second one is the canonical coverage function where each word is assumed to have an importance score, and the score for a summary is the sum of the importance scores of words that the summary contains (f_{concept} in Eqn. 4.23 and see Section 4.6.2 for details). Sipos *et al.* (2012) parameterize submodular score functions by using linear models for inter-sentence similarity and word importance. In other words, similarity or word importance is modeled as weighted sum of features, and the weights are then the model parameters to learn.

The representation of submodular score function in (Sipos *et al.* , 2012) turns out to be a special case of a submodular mixture. To see this, denote the pairwise similarity between element i and j as $\sigma_{i,j}$, and word importance for word v as ω_v. A weighted parameterization is then

$$\sigma_{i,j} = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}, i, j),$$
$$\omega_v = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}, v),$$

where $\boldsymbol{w}$ is the weight vector to be learned, and $\boldsymbol{\phi} \in \mathbb{R}^d$ are feature vectors. Since f_{penalty} is linear in $\sigma_{i,j}$ and f_{concept} is linear in ω_v, the weight vector on σ or ω can be extracted out and serves as the weights for submodular components while the original feature vector is then converted into a new feature vector with values generated from submodular components. Precisely, recall that

$$f_{\text{penalty}}(S) = \sum_{i \in V \setminus S} \sum_{j \in S} \sigma_{i,j} - \lambda \sum_{i,j \in S : i \neq j} \sigma_{i,j},$$

we have

$$
\begin{aligned}
f_{\text{penalty}}(S) &= \sum_{i \in V \setminus S} \sum_{j \in S} \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}, i, j) - \lambda \sum_{i,j \in S : i \neq j} \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}, i, j) \\
&= \boldsymbol{w}^\top \left(\sum_{i \in V \setminus S} \sum_{j \in S} \boldsymbol{\phi}(\boldsymbol{x}, i, j) - \lambda \sum_{i,j \in S : i \neq j} \boldsymbol{\phi}(\boldsymbol{x}, i, j) \right) \\
&= \sum_k w_k \left(\sum_{i \in V \setminus S} \sum_{j \in S} \phi_k(\boldsymbol{x}, i, j) - \lambda \sum_{i,j \in S : i \neq j} \phi_k(\boldsymbol{x}, i, j) \right) \\
&= \sum_k w_k f^k_{\text{penalty}}(S),
\end{aligned}
$$

which is exactly a submodular mixture with components being submodular functions with different inter-similarity measures ($\phi_k(\boldsymbol{x}, i, j)$). Similar analogy can be applied to f_{concept} as well. I.e.,

$$
\begin{aligned}
f_{\text{concept}}(S) &= \sum_{v \in \Gamma(S)} \omega_v \\
&= \sum_{v \in \Gamma(S)} \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}, v) \\
&= \boldsymbol{w}^\top \sum_{v \in \Gamma(S)} \boldsymbol{\phi}(\boldsymbol{x}, v) \\
&= \sum_k w_k \left(\sum_{v \in \Gamma(S)} \phi_k(\boldsymbol{x}, v) \right) \\
&= \sum_k w_k f^k_{\text{concept}}(S).
\end{aligned}
$$

Our submodular mixture framework is more general and flexible than the framework proposed in (Sipos *et al.* , 2012). Although the authors claim that their approach applies to all submodular summarization models, there are many submodular functions useful for summarization that are not linear on either pairwise similarity or singleton importance. For example, in the diversity reward function, we have a concave function over the sums of singleton rewards, and even if using linear model for singleton rewards, the score function is non-linear over parameters since the weights can not be linearly extracted, and thus

the algorithms in (Sipos *et al.* , 2012) does not apply. Viewing each feature as input to a submodular component, on the other hand, preserves the linearity on the parameters, whereas all the algorithms introduced in Section 5.4 apply. Moreover, representing score function using a submodular mixture is very powerful in terms of expressiveness, as we have shown in Section 5.3.

In (Sipos *et al.* , 2012), a cutting plane algorithm with one minus ROUGE F-measure as loss function was used to learn model weights. When doing the loss augmented inference, they use our greedy algorithm (Algorithm 4) to approximately optimize the objective. Their objective function, however, is not submodular, due to the non-submodular loss function they use. Therefore, the LAI inference is no longer guaranteed to be near-optimal, and the performance of approximate learning with their cutting plane algorithm is no longer guaranteed (Finley & Joachims, 2008). Also, Sipos *et al.* (2012) apparently neglect the fact that the learned similarity (or importance) should be non-negative, which are necessary ingredients to preserve the submodularity of the learned score function. One simple way to ensure this is to constrain the weights to be non-negative, as we do for submodular mixtures.

5.6.4 Results

We evaluated learning submodular mixture approach on DUC data 2003-2007, and demonstrate results on both generic and query-focused summarization. See Section 4.7.3 for details about the DUC data and the preprocessing steps.

Generating pseudo labels

The automatic summarizers of interest here are extractive. On the other hand, although human references are available for all DUC tasks, they are not necessary extractive. In other words, annotators had freedom to use their own words when summarizing a document cluster, and were not restricted to use the exact sentences from the documents to be summarized. Therefore, labels are not available for DUC tasks in the sense of supervised training. To address this issue, we generate "pseudo labels" using the human reference. Since ROUGE-N is monotone submodular (Theorem 22), we can use the greedy algorithm

to optimize it near-optimally, given the human references. The outputs of Algorithm 4 over ROUGE-N score functions are then used as pseudo labels. The pseudo labels are required for certain type of loss functions, e.g., the hamming loss. For the ROUGE-like loss function we proposed in Section 5.6.2, no extractive labels are required, and thus pseudo labels we generated were only used for hamming loss in our experiments.

Query-independent summarization

Table 5.2: ROUGE-1 recall (R) and F-measure (F) results (%) on DUC-04. DUC-03 was used as development set.

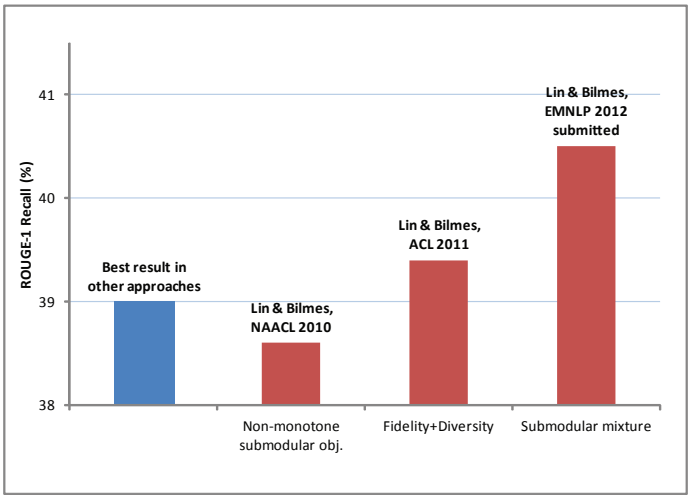
DUC-04	R	F
Takamura & Okumura (2009)	38.50	-
Wang *et al.* (2009)	39.07	-
Lin & Bilmes (2010b)	-	38.39
Best system in DUC-04 (peer 65)	38.28	37.94
Lin & Bilmes (2011a)	39.35	38.90
Submodular Mixture	**40.43**	**39.78**

The summarization tasks in DUC-03 and DUC-04 are generic summarization tasks. We used DUC-03 as the training set for submodular mixture learning. There are in total 60 document clusters in the DUC-03 task, therefore we have 60 training examples in total. We used a submodular mixture with 15 fidelity components for this task. In particular, the $\mathcal{C}_i$ function we used is

$$\mathcal{C}_i(S) = \sum_{j \in V} \delta_{i,j}$$

where $\delta_{i,j}$ is the pairwise sentence similarity between sentence i and j. We used three types of sentence similarities. The first two are cosine similarities with unigram and bigram TF-IDF vectors respectively (see Section 4.7.3 for more details about the similarity generation). The third similarity is again cosine similarity but on the vector generated by latent semantic analysis. For each of the similarity measure, we use five different saturation thresholds

Figure 5.1: Performance evolvement of submodular summarization approach.



($\beta = 0.01, \cdots, 0.05$), and thus we have 15 fidelity components in total. We used the projected subgradient algorithm (Algorithm 13) with ROUGE-like loss (Eqn. 5.19) to learn this submodular mixture. The ROUGE-1 results are shown in Table 5.2. As we can see, the result of the learned submodular mixture significantly outperforms all other previous reported results, and is the best result ever reported on DUC-04 as far as we know.

Figure 5.1 shows how the performance of our submodular summarization approaches have evolved in recent years.

We also compared the performance of using different loss functions in the subgradient algorithm. Two other loss functions are considered: the hamming loss (which uses pseudo labels), and the one-minus-ROUGE loss (non-submodular objective). The ROUGE-1 recall score improvement as the number of iterations increases in the subgradient algorithm is shown in Figure 5.2. Both the two ROUGE related loss functions outperform the Hamming loss with pseudo labels, while the proposed ROUGE-like loss shows superiority over one-minus-ROUGE loss.

Figure 5.2: Convergence of projected subgradient algorithm with different loss functions (training set: DUC-03, test set: DUC-04).

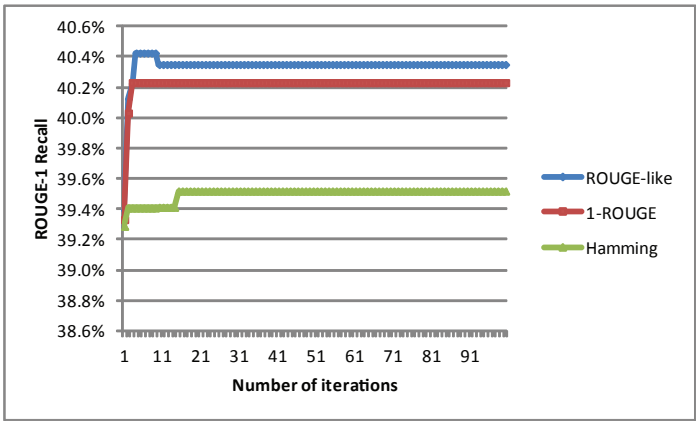
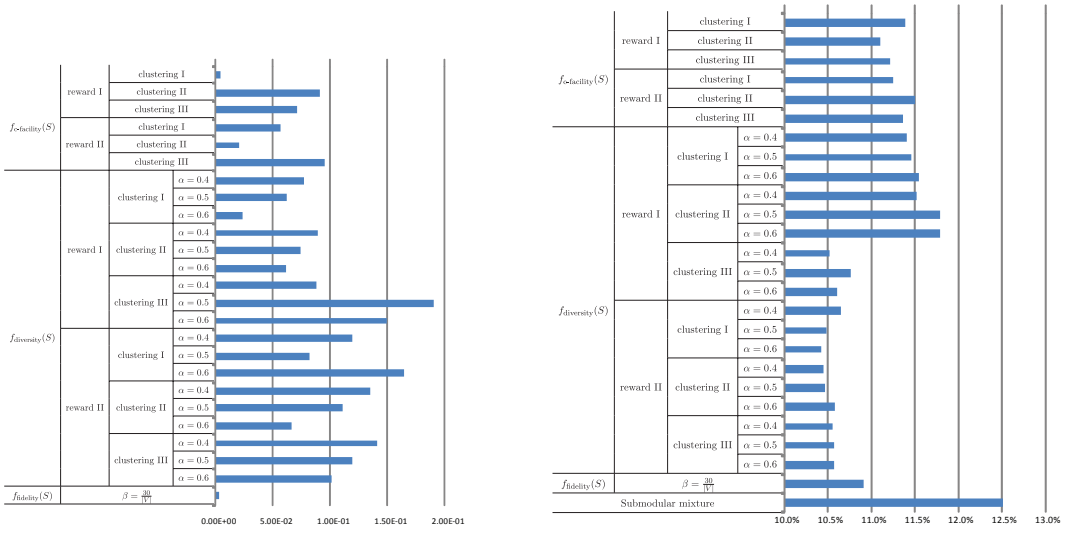


Figure 5.3: Visualization of component weights and ROUGE-2 recall scores.



(a) Component weight

(b) Component ROUGE score

Table 5.3: Component weights of the submodular mixture trained on DUC-05 and DUC-06 and single component performance (ROUGE-2 recall %) trained on DUC-05,06 and tested on DUC-07.

Submodular component				Component weight	Single comp. ROUGE-2-R
$f_{\text{c-facility}}(S)$ Eqn. 5.17	reward I	clustering I		0.005	11.39
		clustering II		0.091	11.10
		clustering III		0.071	11.22
	reward II	clustering I		0.057	11.25
		clustering II		0.028	11.50
		clustering III		0.095	11.36
$f_{\text{diversity}}(S)$ Eqn. 5.16	reward I	clustering I	$\alpha = 0.4$	0.077	11.41
			$\alpha = 0.5$	0.062	11.46
			$\alpha = 0.6$	0.024	11.55
		clustering II	$\alpha = 0.4$	0.090	11.52
			$\alpha = 0.5$	0.074	11.79
			$\alpha = 0.6$	0.061	11.79
		clustering III	$\alpha = 0.4$	0.088	10.52
			$\alpha = 0.5$	0.190	10.76
			$\alpha = 0.6$	0.150	10.60
	reward II	clustering I	$\alpha = 0.4$	0.119	10.76
			$\alpha = 0.5$	0.082	10.49
			$\alpha = 0.6$	0.165	10.42
		clustering II	$\alpha = 0.4$	0.135	10.45
			$\alpha = 0.5$	0.111	10.46
			$\alpha = 0.6$	0.067	10.68
		clustering III	$\alpha = 0.4$	0.141	10.55
			$\alpha = 0.5$	0.119	10.57
			$\alpha = 0.6$	0.101	10.57
$f_{\text{fidelity}}(S)$ Eqn. 5.18	$\beta = \frac{30}{\lvert V \rvert}$			0.003	10.91
Submodular Mixture				-	12.51

Query-focused summarization

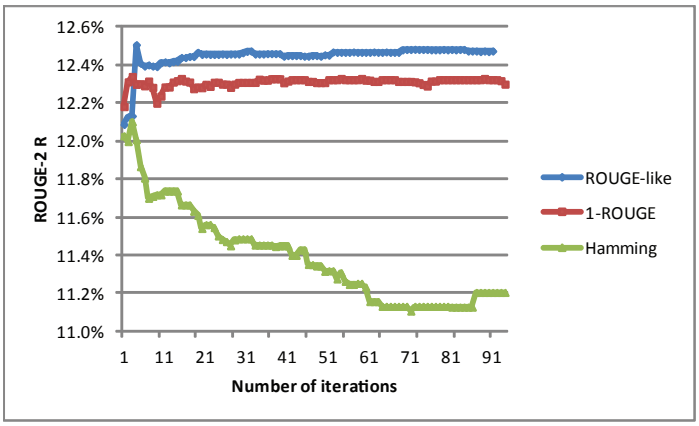
Since 2004, DUC summarization evaluation had started focusing on query-focused summarization. See Section 4.7.3 for details about the data and our preprocessing procedures.

We tested our submodular mixture approach for summarization on DUC-05, DUC-06 and DUC-07 data. In particular, we used DUC-06 and DUC-07 as the training set for the DUC-05 task, DUC-05 and DUC-07 as the training set for the DUC-06 task, and DUC-05 and DUC-06 as the training set for the DUC-07 task.

We used diversity components, clustered facility location components, and fidelity components to form the submodular mixture for query-focused summarization. For the diversity and clustered facility location components, clusterings were generated by CLUSTO (refer to Section 4.7.3 for details). Three clusterings with different numbers of clusters were created ($K = 0.1|V|, 0.2|V|, 0.3|V|$). As for the singleton rewards, we used both query-independent and query-dependent singleton rewards. The query-independent reward for i is simply the summation of the pairwise similarities of other other elements in V to i. For the query-dependent reward, we simply used the number of terms (up to a bi-gram) that sentence j overlaps the query Q, where the IDF weighting is not used (i.e., every term in the query, after stop word removal, was treated as equally important). Therefore, in total, we have 6 clustered facility location components. We further used three curvatures in diversity components ($\alpha = 0.5, 0.6, 0.7$), which gives 18 diversity components in total. With one additional fidelity component, we have 25 components in sum.

All the components, along with their ROUGE-2 recall scores, are listed in Table 5.3. Where single component performance is refer to the ROUGE-2 recall score we got by training each component on DUC-05 and DUC-06 and then testing them on DUC-07. The single component's performance is further visualized in Figure 5.3(b). In general, components with query-dependent rewards do not perform well by themselves. However, when combined with other query-independent components, they play significant roles and boot the performance of the mixture to a level that is much better than any single components. This can be seen from the component weights (listed in Table 5.3 and also visualized in Figure 5.3(a)) of the final learned submodular mixture, where the weights on components with query-dependent

Figure 5.4: Convergence of projected subgradient algorithm with different loss functions (training sets: DUC-05 and DUC-06, test set: DUC-07).



information take major mass of the weight distribution.

We also compared the performance of using different loss functions in the subgradient algorithm. Similar to what have been observed in the generic summarization task, ROUGE-like loss achieves better ROUGE score than those obtained by using the one-minus-ROUGE score. In this case, we see performance of using Hamming loss decreases as the number of iteration increases, presumably because the pseudo labels were in low quality, and also optimizing (convex relaxation) of the Hamming loss does not necessary imply better ROUGE scores.

When compared to other results reported in literature (See Table 5.4, Table 5.5 and Table 5.6), our submodular mixture approach achieves best results ever reported on DUC-05, DUC-06, and DUC-07.

Table 5.4: ROUGE-2 recall (R) and F-measure (F) results on DUC-05 (%). We used DUC-06 and DUC-07 as training sets.

DUC-05	R	F
Daumé III & Marcu (2006)	6.98	-
Best system in DUC-05 (peer 15)	7.44	7.43
Lin & Bilmes (2011a)	7.82	7.72
Submodular Mixture	**8.44**	**8.39**

Table 5.5: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-06, where DUC-05 and DUC-07 were used as training sets.

DUC-06	R	F
Celikyilmaz & Hakkani-tür (2010)	9.10	-
Shen & Li (2010)	9.30	-
Best system in DUC-06 (peer 24)	9.51	9.51
Lin & Bilmes (2011a)	9.75	9.77
Submodular Mixture	**9.92**	**9.93**

Table 5.6: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-07. DUC-05 and DUC-06 were used as training sets.

DUC-07	R	F
Toutanova *et al.* (2007)	11.89	11.89
Haghighi & Vanderwende (2009)	11.80	-
Celikyilmaz & Hakkani-tür (2010)	11.40	-
Best system in DUC-07 (peer 15)	12.45	12.29
Lin & Bilmes (2011a)	12.38	12.33
Submodular Mixture	**12.51**	**12.40**

Chapter 6

WORD ALIGNMENT VIA SUBMODULAR MAXIMIZATION OVER MATROIDS

We cast the word alignment problem as maximizing a submodular function under matroid constraints. Our framework is able to express complex interactions between alignment components while remaining computationally efficient, thanks to the power and generality of submodular functions. We show that submodularity naturally arises when modeling word fertility. Experiments on the English-French Hansards alignment task show that our approach achieves lower alignment error rates compared to conventional matching based approaches.

6.1 Introduction

Word alignment is a key component in most statistical machine translation systems. While classical approaches for word alignment are based on generative models (e.g., IBM models (Brown *et al.* , 1993) and HMMs (Vogel *et al.* , 1996)), word alignment can also be viewed as a matching problem (see Figure 6.1 for an example), where each word pair is associated with a score reflecting the desirability of aligning that pair, and the alignment is then the highest scored matching under some constraints.

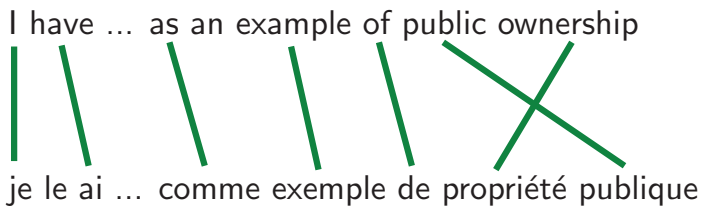


Figure 6.1: An example of word alignment between an English sentence and a French sentence.

Several matching-based approaches have been proposed in the past. Melamed (2000) introduces the competitive linking algorithm which greedily constructs matchings under the one-to-one mapping assumption. In (Matusov *et al.* , 2004), matchings are found using an algorithm for constructing a maximum weighted bipartite graph matching (Schrijver, 2003), where word pair scores come from alignment posteriors of generative models. Similarly, Taskar *et al.* (2005a) cast word alignment as a maximum weighted matching problem and propose a framework for learning word pair scores as a function of arbitrary features of that pair. These approaches, however, have two potentially substantial limitations: words have fertility of at most one, and interactions between alignment decisions are not representable.

Lacoste-Julien *et al.* (2006) address this issue by formulating the alignment problem as a quadratic assignment problem, and off-the-shelf integer linear programming (ILP) solvers are used to solve to optimization problem. While efficient for some median scale problems, ILP-based approaches are limited since when modeling more sophisticated interactions, the number of variables (and/or constraints) required grows polynomially, or even exponentially, making the resultant optimization impractical to solve.

In this chapter, we treat the word alignment problem as maximizing a submodular function subject to matroid constraints. Submodular objective functions can represent complex interactions among alignment decisions, and essentially extend the modular (linear) objectives used in the aforementioned approaches. While our extensions add expressive power, they do *not* result in a heavy computational burden. This is because maximizing a monotone submodular function under a matroid constraint can be solved efficiently using a simple greedy algorithm. The greedy algorithm, moreover, is a constant factor approximation algorithm that guarantees a near-optimal solution. In this paper, we moreover show that submodularity naturally arises in word alignment problems when modeling word fertility (see Section 6.3). Experiment results on the English-French Hansards alignment task show that our approach achieves lower alignment error rates compared to the maximum weighted matching approach, while being at least 50 times faster than an ILP-based approach.

Backgrounds on submodularity and matroids can be found in Chapter 2.

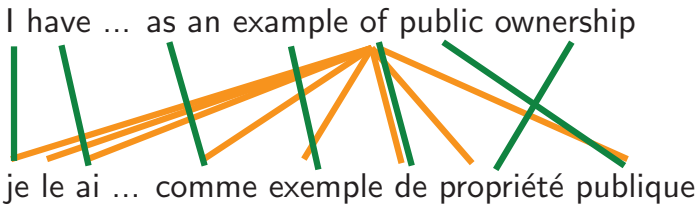


Figure 6.2: An example of modeling fertility using intersection of sets. Green lines represent an alignment A, orange lines represent P_i^F where i indicates word "of" in the English sentence, and $|A \cap P_i^F|$ suggests how many words in alignment A are aligned to word "of".

6.2 Approach

We are given a source language (English) string $e_1^I = e_1, \cdots, e_i, \cdots, e_I$ and a target language (French) string $f_1^J = f_1, \cdots, f_j, \cdots, f_J$ that have to be aligned. Define the word positions in the English string as set $E \triangleq \{1, \cdots, I\}$ and positions in the French string as set $F \triangleq \{1, \cdots, J\}$. An alignment A between the two word strings can then be seen as a subset of the Cartesian product of the word positions, i.e., $A \subseteq \{(i,j) : i \in E, j \in F\} \triangleq V$, and $V = E \times F$ is the ground set. For convenience, we refer to element $(i,j) \in A$ as an *edge* that connects i and j in alignment A.

Restricting the fertility of word f_j to be at most k_j is mathematically equivalent to having $|A \cap P_j^E| \leq k_j$, where $A \subseteq V$ is an alignment and $P_j^E = E \times \{j\}$. Intuitively, P_j^E is the set of all possible edges in the ground set that connect to j, and the cardinality of the intersection between A and P_j^E indicates how many edges in A are connected to j.

Similarly, we can impose constraints on the fertility of English words by constraining the alignment A to satisfy $|A \cap P_i^F| \leq k_i$ for $i \in E$ where $P_i^F = \{i\} \times F$. See Figure 6.2 for an example, where green lines represent an alignment A, orange lines represent P_i^F where i indicates word "of" in the English sentence, and $|A \cap P_i^F|$ suggests how many words in alignment A are aligned to word "of".

Note that either of $\{P_j^E : j \in F\}$ or $\{P_i^F : i \in E\}$ constitute a partition of V. Therefore, alignments A that satisfy $|A \cap P_j^E| \leq k_j, \forall j \in F$, are independent in the *partition matroid*

$\mathcal{M}_E = (V, \mathcal{I}_E)$ with

$$\mathcal{I}_E = \{A \subseteq V : \forall j \in F, |A \cap P_j^E| \leq k_j\},$$

and alignments A that satisfy $|A \cap P_i^F| \leq k_i, \forall i \in E$, are independent in matroid $\mathcal{M}_F = (V, \mathcal{I}_F)$ with

$$\mathcal{I}_F = \{A \subseteq V : \forall i \in E, |A \cap P_i^F| \leq k_i\}.$$

Suppose we have a set function $f : 2^V \rightarrow \mathbb{R}_+$ that measures quality (or scores) of an alignment $A \subseteq V$, then when also considering fertility constraints, we can treat the word alignment problem as maximizing a set function subject to matroid constraint:

Problem 14. $\max_{A \subseteq V} f(A)$, *subject to:* $A \in \mathcal{I}$,

where $\mathcal{I}$ is the set of independent sets of a matroid (or it might be the set of independent sets simultaneously in two matroids, as we shall see later).

Independence in partition matroids generalizes the typical matching constraints for word alignment, where each word aligns to at most one word ($k_j = 1, \forall j$) in the other sentence (Matusov *et al.* , 2004; Taskar *et al.* , 2005a). Our matroid generalizations provide flexibility in modeling fertility, and also strategies for solving the word alignment problem efficiently and near-optimally. In particular, when f is monotone submodular, near-optimal solutions for Problem 14 can be efficiently guaranteed.

For example, in (Fisher *et al.* , 1978), a simple greedy algorithm for monotone submodular function maximization with a matroid constraint is shown to have a constant approximation factor. Precisely, the greedy algorithm finds a solution A such that $f(A) \geq \frac{1}{m+1} f(A^*)$ where A^* is the optimal solution and m is number of matroid constraints. When there is only one matroid constraint, we get an approximation factor $\frac{1}{2}$. Constant factor approximation algorithms are particularly attractive since the quality of the solution does not depend on the size of the problem, so even very large size problems do well. It is also important to note that this is a worst case bound, and in most cases the quality of the solution obtained will be much better than this bound suggests.

Vondrák (2008) shows a continuous greedy algorithm followed by pipage rounding with approximation factor $1 - 1/e$ (≈ 0.63) for maximizing a monotone submodular function

subject to a matroid constraint. Lee *et al.* (2009c) improve the $\frac{1}{m+1}$-approximation result in (Fisher *et al.* , 1978) by showing a local-search algorithm has approximation guarantee of $\frac{1}{m+\epsilon}$ for the problem of maximizing a monotone submodular function subject to m matroid constraints ($m \geq 2$ and $\epsilon > 0$). In this paper, however, we use the simple greedy algorithm for the sake of efficiency. We outline our greedy algorithm for Problem 14 in Algorithm 14, which is slightly different from the one in (Fisher *et al.* , 1978) as in line 14 of Algorithm 14, we have an additional requirement on a such that the increment of adding a is *strictly* greater than zero. This additional requirement is to maintain a higher precision word alignment solution. The theoretical guarantee still holds as f is monotone — i.e., Algorithm 14 is a $\frac{1}{2}$-approximation algorithm for Problem 14 (only one matroid constraint) when f is monotone submodular.

Algorithm 14: A greedy algorithm for Problem 14.

input : $A = \emptyset, N = V$.

begin

  while $N \neq \emptyset$ **do**

    $a \leftarrow \text{argmax}_{e \in N} f(A \cup \{e\}) - f(A)$;

    if $A \cup \{a\} \in \mathcal{I}$ **and** $f(A \cup \{a\}) - f(A) > 0$ **then**

      $A \rightarrow A \cup \{a\}$

    $N \rightarrow N \setminus \{a\}$.

Algorithm 14 requires $O(|V|^2)$ evaluations of f. In practice, the argmax in Algorithm 14 can be efficient implemented with priority queue when f is submodular (Minoux, 1978), which brings the complexity down to $O(|V| \log |V|)$ oracle function calls.

6.3 Submodular Fertility

We begin this section by demonstrating that submodularity arises naturally when modeling word fertility. To do so, we borrow an example of fertility from (Melamed, 2000). Suppose a trained model estimates $s(e_1, f_1) = .05, s(e_1, f_2) = .02$ and $s(e_2, f_2) = .01$, where $s(e_i, f_j)$ represents the score of aligning e_i and f_j. To find the correct alignment (e_1, f_1) and (e_2, f_2),

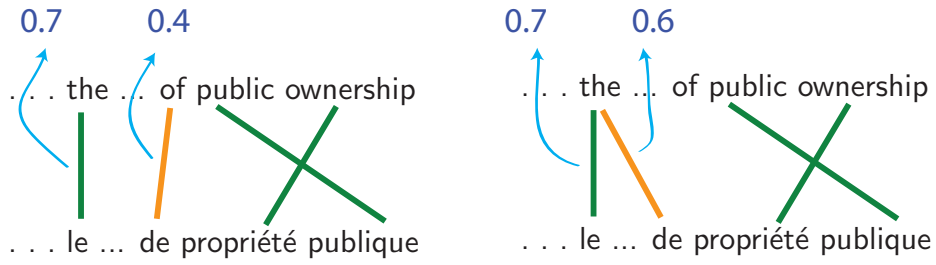


Figure 6.3: An example of diminishing returns when modeling word fertility.

the competitive linking algorithm in (Melamed, 2000) poses a one-to-one assumption to prevent choosing (e_1, f_2) over (e_2, f_2). The one-to-one assumption, however, limits the algorithm's capability of handling models with fertility larger than one. Alternatively, we argue that the reason for choosing (e_2, f_2) rather than (e_1, f_2) is that the benefit of aligning e_1 and f_2 *diminishes* after e_1 is already aligned with f_1 — this is exactly the property of diminishing returns, and therefore, it is natural to use submodular functions to model alignment scores.

To illustrate this further, we use another real example taken from the trial set of English-French Hansards data. The scores estimated from the data for aligning word pairs (*the, le*), (*the, de*) and (*of, de*) are $0.68, 0.60$ and 0.44 respectively. Given an English-French sentence pair: "*I have stressed the CDC as an example of creative, aggressive effective public ownership*" and "*je le ai cité comme exemple de propriété publique créatrice, dynamique et efficace*", an algorithm that allows word fertility larger than 1 might choose alignment (*the, de*) over (*of, de*) since $0.68 + 0.60 > 0.68 + 0.44$, regardless the fact that *the* is already aligned with *le*. Now if we use a submodular function to model the score of aligning an English word to a set of French words, we might obtain the correct alignments (*the, le*) and (*of, de*) by incorporating the diminishing returns property (i.e., the score gain of (*the, de*), which is 0.60 out of context, could diminish to something less than 0.44 when evaluated in the context of (*the, le*)).

Formally, for each i in E, we define a mapping $\delta_i : 2^V \rightarrow 2^F$ with

$$\delta_i(A) = \{j \in F | (i, j) \in A\}, \tag{6.1}$$

i.e., $\delta_i(A)$ is the set of positions in F that are aligned with position i in alignment A.

We use function $f_i : 2^F \rightarrow \mathbb{R}_+$ to represent the benefit of aligning position $i \in E$ to a set of positions in F. Given score $s_{i,j}$ of aligning i and j, we could have, for $S \subseteq F$,

$$f_i(S) = \left(\sum_{j \in S} s_{i,j} \right)^{\alpha}, \tag{6.2}$$

where $0 < \alpha \leq 1$, i.e., we impose a concave function over a modular function, which produces a submodular function. The value of α determines the rate that the marginal benefit diminishes when aligning a word to more than one words in the other string.

Summing over alignment scores in all positions in E, we obtain the total score of an alignment A:

$$f(A) = \sum_{i \in E} f_i(\delta_i(A)), \tag{6.3}$$

which is again, monotone submodular. By diminishing the marginal benefits of aligning a word to more than one words in the other string, $f(A)$ encourages the common case of low fertility while allowing fertility larger than one. For instance in the aforementioned example, when $\alpha = \frac{1}{2}$, the score for aligning both *le* and *de* to *the* is $\sqrt{0.68 + 0.60} \approx 1.13$, while the score of aligning *the* to *le* and *of* to *de* is $\sqrt{0.68} + \sqrt{0.44} \approx 1.49$, leading to the correct alignment.

6.4 Experiments

We evaluated our approaches using the English-French Hansards data from the 2003 NAACL shared task (Mihalcea & Pedersen, 2003). This corpus consists of 1.1M automatically aligned sentences, and comes with a test set of 447 sentences, which have been hand-aligned and are marked with both "sure" and "possible" alignments (Och & Ney, 2003). Using these alignments, *alignment error rate* (AER) is calculated as:

$$AER(A, S, P) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|} \tag{6.4}$$

where S is the set of sure gold pairs, and P is the set of possible gold pairs. We followed the work in (Taskar *et al.* , 2005a) and split the original test set into 347 test examples, and 100 training examples for parameters tuning.

In general, the score of aligning i to j can be modeled as a function of arbitrary features. Although parameter learning in our framework would be another interesting topic to study, we focus herein on the inference problem. Therefore, only one feature (Eq. 6.5) was used in our experiments in order for no feature weight learning to be required. In particular, we estimated the score of aligning i to j as

$$s_{i,j} = \frac{p(f_j|e_i) \cdot p(i|j, I)}{\sum_{j' \in F} p(f_{j'}|e_i) \cdot p(i|j', I)}, \tag{6.5}$$

where the translation probability $p(f_j|e_i)$ and alignment probability $p(i|j, I)$ were obtained from IBM model 2 trained on the 1.1M sentences. The IBM 2 models gives an AER of 21.0% with French as the target, in line with the numbers reported in Och & Ney (2003) and Lacoste-Julien *et al.* (2006).

We tested two types of partition matroid constraints. The first is a global matroid constraint:

$$A \in \{A' \subseteq V : \forall j \in F, |A' \cap P_j^E| \leq b\}, \tag{6.6}$$

which restricts fertility of *all* words on F side to be at most b. This constraint is denoted as $\text{Fert}_F(A) \leq b$ in Table 6.1 for simplicity. The second type, denoted as $\text{Fert}_F(A) \leq k_j$, is word dependent:

$$A \in \{A' \subseteq V : \forall j \in F, |A' \cap P_j^E| \leq k_j\}, \tag{6.7}$$

where the fertility of word on j is restricted to be at most k_j. Here $k_j = \max\{b : p_b(j) \leq \theta, b \in \{0, 1, \ldots, 5\}\}$, where θ is a threshold and $p_b(j)$ is the probability that French word j was aligned to at most b English words based on the IBM 2 alignment.

As mentioned in Section 6.2, matroid constraints generalize the matching constraint. In particular, when using two matroid constraints, $\text{Fert}_E(A) \leq 1$ and $\text{Fert}_F(A) \leq 1$, we have the matching constraint where fertility for both English and French words are restricted to be at most one. Our setup 1 (see Table 6.1) uses these two constraints along with a modular objective function, which is equivalent to the maximum weighted bipartite matching problem. Using greedy algorithm to solve this problem, we get AER 21.0% (setup 1 in Table 6.1) – no significant difference compared to the AER (20.9%) achieved by the

Table 6.1: AER results

ID	Objective function	Constraint	AER(%)
1	modular: $f(A) = \sum_{i \in E} \sum_{j \in \delta_i(A)} s_{i,j}$	$\text{Fert}_F(A) \leq 1, \text{Fert}_E(A) \leq 1$	21.0
2		$\text{Fert}_F(A) \leq 1$	23.1
3		$\text{Fert}_F(A) \leq k_j$	22.1
4	submodular: $f(A) = \sum_{i \in E} \left(\sum_{j \in \delta_i(A)} s_{i,j} \right)^{\alpha}$	$\text{Fert}_F(A) \leq 1$	19.8
5		$\text{Fert}_F(A) \leq k_j$	**18.6**
Generative model (IBM 2, E→F)			21.0
Maximum weighted bipartite matching			20.9
Matching with negative penalty on fertility (ILP)			19.3

exact solution (maximum weighted bipartite matching approach), illustrating that greedy solutions are near-optimal. Note that the bipartite matching approach does not improve performance over IBM 2 model, presumably because only one feature was used here.

When allowing fertility of English words to be more than one, we see a significant AER reduction using a submodular objective (setup 4 and 5) instead of a modular objective (setup 2 and 3), which verifies our claim that submodularity lends itself to modeling the marginal benefit of growing fertility. In setup 2 and 4, while allowing larger fertility for English words, we restrict the fertility of French words to be most one. To allow higher fertility for French words, one possible approach is to use constraint $\text{Fert}_F(A) \leq 2$, in which all French words are allowed to have fertility up to 2. This approach, however, results in a significant increase of false positive alignments since all French words tend to collect as many matches as permitted. This issue could be alleviated by introducing a symmetric version of the objective function in Eq. 6.3 such that marginal benefit of higher fertility of French words are also compressed. Alternatively, we use the second type of matroid constraint in which fertility upper bounds of French words are word dependent instead of global. With $\theta = .8$, about 10 percent of the French words have k_j equal to 2 or greater. By using the word dependent matroid constraint (setup 3 and 5), AERs are reduced compared to those using global matroid constraints. In particular, 18.6% AER is achieved by setup 5, which

significantly outperforms the maximum weighted bipartite matching approach.

We also compare our method with model of Lacoste-Julien *et al.* (2006) which also allows fertility larger than one by penalizing different levels of fertility. We used $s_{i,j}$ as an edge feature and $p_b(f)$ as a node feature together with two additional features: a bias feature and the bucketed frequency of the word type. The same procedures for training and decoding as in (Lacoste-Julien *et al.* , 2006) were performed where MOSEK was used as the ILP solver. As shown in Table 6.1, performance of setup 5 outperforms this model and moreover, our approach is at least 50 times faster: it took our approach only about half a second to align all the 347 test set sentence pairs whereas using the ILP-based approach took about 40 seconds.

6.5 Discussion

We have presented a novel framework where word alignment is framed as submodular maximization subject to matroid constraints. Our framework extends previous matching-based frameworks in two respects: submodular objective functions generalize modular (linear) objective functions, and matroid constraints generalize matching constraints. Moreover, such generalizations do not incur a prohibitive computational price since submodular maximization over matroids can be efficiently solved with performance guarantees. As it is possible to leverage richer forms of submodular functions that model higher order interactions, we believe that the full potential of our approach has yet to be explored. Our approach might lead to novel approaches for machine translation as well.

Chapter 7

OPTIMAL SELECTION OF LIMITED VOCABULARY SPEECH CORPORA

We address the problem of finding a subset of a large speech data corpus that is useful for accurately and rapidly prototyping novel and computationally expensive speech recognition architectures. To solve this problem, we express it as an optimization problem over submodular functions. Quantities such as vocabulary size (or quality) of a set of utterances, or quality of a bundle of word types are submodular functions which make finding the optimal solutions possible. We, moreover, are able to express our approach using graph cuts leading to a very fast implementation even on large initial corpora. We show results on the Switchboard-I corpus, demonstrating improved results over previous techniques for this purpose. We also demonstrate the variety of the resulting corpora that may be produced using our method.

7.1 Introduction

Large vocabulary spontaneous conversational speech recognition is one of the most challenging tasks in speech processing and is one of the most computationally demanding in all machine learning. In recent times, very large amounts of transcribed data, with both many tokens and many types, have become available. Some corpora have a vocabulary size as large as one million and as many as 230 billion tokens (Chelba *et al.* , 2010)! While having such a wealth of training data is useful from the perspective of producing better speech recognition systems (there is no data like more data), the data size itself presents a serious problem for novel speech recognition research.

Novel ASR systems are often not highly optimized or tuned, including at the implementation level (where low-level coding tricks and years of human effort can have a significant speed and memory benefit) and also at the algorithmic level (where different or new algorithms can later be discovered to more efficiently solve the same underlying problem). The

more novel the idea, the more effort it takes to get it working on a large system since there is less chance of potential implementation reuse from a pre-existing system. In general, it is important to be able to test a novel idea quickly, without investing enormous amounts of time on the engineering effort to make the ideas perform well, and if a new idea ends up performing poorly, knowing this sooner rather than later will avoid futile work.

Novel speech recognition systems, moreover, deserve rich data on which to be evaluated. For example, novel systems might not show their benefit on data lacking the characteristics the novel system is designed to address. Now, large corpora are useful not simply because they are large, but because they contain information simply unavailable in typical smaller corpora. For example, a large corpus can contain not only rich phonetic variety but also a full representation of that variety. That is, a large corpus has many samples of high probability word pronunciations (certain pronunciations might even be over-represented, and less data is sufficient to produce for an accurate model). Even low probability pronunciations, however, might have a sufficient number of samples in a very large corpus to produce a good statistical pronunciation model.

On the other hand, recent large data sets are unkind to novel ASR systems simply because they are so large. ASR systems often have complexity that is linear in the number of tokens and polynomial in the number of types (e.g., decoding using a trigram language model with size-N vocabulary has, in the worst case, a complexity of at least $O(N^3)$). A challenge is determining how to quickly test a new system on large data sets.

One way to address this problem is to produce a smaller version of the corpus, and one way to do this is to draw a subset uniformly at random. Any such subset, however, might not possess the richness mentioned above. It should moreover be possible to produce a smaller corpus that removes over-representation while retaining proper representation for every speech unit. Ideally, we would like a process that can take a large corpus and produce a subset that satisfies a particular purpose. For example, when vocabulary size is the key attribute hindering the rapid evaluation of novel acoustic method, we might choose a limited vocabulary subset of data of maximal size. On the other hand, we may wish to correct for some other quality, such as imposing a bias against certain word forms. We moreover wish the results from the corpus to be an accurate reflection of results from the entire corpus.

In this chapter we address this problem by formulating it as an optimization problem via the use of submodular functions (Fujishige, 2005a). As we will see, this allows us to express the problem of corpus subset selection in a variety of flexible ways, suiting the needs of an individual novel ASR system and its designer, and allows us to find the optimal solution for our objective, improving on a previously proposed method for this purpose (King *et al.* , 2005).

7.2 Why Not Greedy

Our goal is to create a corpus of spontaneous conversational speech with limited (small) vocabulary. We do this by selecting utterances from a large vocabulary conversational speech corpus (e.g., Swithboard). The question is: how can we select as much and as rich acoustic data as possible while limiting the vocabulary size?

One straightforward and simple way to solve this corpus subset selection problem is to use a greedy algorithm: Here, we start with an empty vocabulary. In each greedy step, we add an out-of-vocabulary (OOV) word into the vocabulary if the amount of data containing **only and nothing other than** this new vocabulary is maximized (this is made formal below via function f_{svb}). The algorithm stops when the desired vocabulary size is reached. This algorithm was used in (King *et al.* , 2005), for instance.

The greedy algorithm, although conceptually simple, could perform arbitrarily poorly for the corpus subset selection problem. To see this, we first formalize the greedy approach as follows. Let F be the set of distinct words (i.e., vocabulary, or types) in the original (large) corpus. For $Y \subseteq F$, let the function $\zeta(Y)$ denote the set of utterances that contain *only* words in Y. Given an utterance v, let t_v represent the amount of data contained in x. For instance, in (King *et al.* , 2005), t_v is the number of word tokens in utterance v. The greedy algorithm then attempts to select a set of words such that a set function $f_{\text{svb}} : 2^F \rightarrow \mathbb{R}$ is maximized, where

$$f_{\text{svb}}(Y) \triangleq \sum_{v \in \zeta(Y)} t_v. \tag{7.1}$$

Now it can be shown that f_{svb} is a supermodular (Narayanan, 1997b) set function (see Appendix for a proof). Therefore, the greedy algorithm above attempts to solve the problem

of maximizing a supermodular function subject to a cardinality constraint. Unfortunately, the greedy algorithm in this case has an unboundedly poor approximation factor. For instance, let $F = \{a, b, c\}, f(\{a\}) = 1, f(\{b\}) = f(\{c\}) = 0, f(\{a, b\}) = f(\{a, c\}) = 1$, $f(\{b, c\}) = p > 1$ and the cardinality is constrained to be at most 2. This function f is supermodular. Greedily maximizing f leads to a solution $\{a, b\}$ with objective function value 1, while the true optimal objective function value is p. Since p is arbitrary, the approximation factor for this example is unboundedly poor. This is unsurprising, as it is known that greedy algorithm works near-optimally only when maximizing a *submodular* function subject to cardinality (knapsack) constraint (Nemhauser *et al.* , 1978), and this nice property has been used in our previous work on selecting good unlabeled training data to transcribe (Lin & Bilmes, 2009) and for document summarization (Lin & Bilmes, 2010b, 2011a).

7.3 Problem Setup

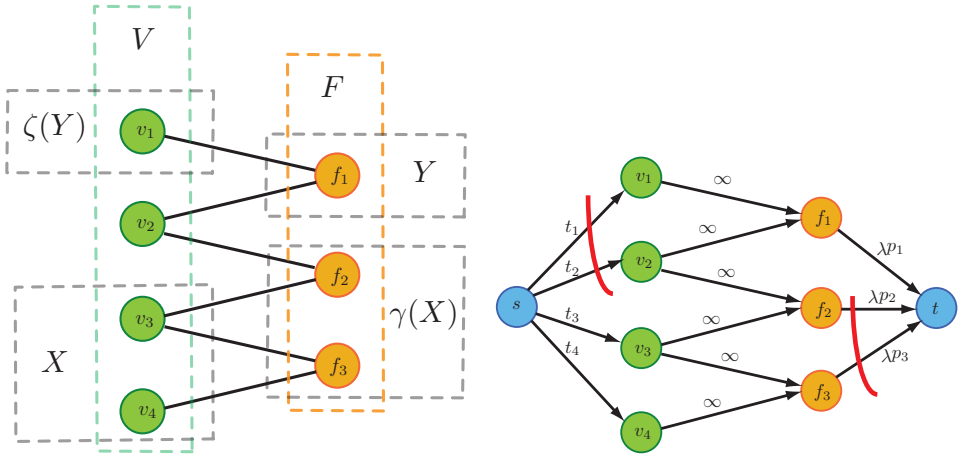
In this chapter, we treat the corpus creation problem as finding a subset of utterances that simultaneously minimizes the vocabulary size and maximizes the total amount of data, measured as either the number of utterances, the number of tokens in the utterances, or duration of speech. The problem can be seen as a combinatorial optimization problem defined on a bipartite graph. Let V be a set of corpus utterances, and let F be the vocabulary (set of distinct words) contained collectively in these utterances. We define a bipartite graph $G = (V, F, E)$ where $E \subseteq V \times F$ are the set of edges. Each $(v, f) = e \in E$ is an edge between an utterance $v \in V$ and a word $f \in F$ if utterance v contains word f (see Figure 7.1(a) for example). The problem of interest is then:

Problem 15. *We find $X \subseteq V$ that maximizes the following objective function*

$$w(X) - \lambda\Gamma(X) \tag{7.2}$$

where $w(X)$ measures the amount of data contained in utterances X, $\Gamma(X)$ represents the vocabulary size associated with utterances X, and $\lambda \geq 0$ is a tradeoff coefficient.

Intuitively, maximizing Eq. 7.2 simultaneously maximizes the total amount of data



(a) Bipartite graph (b) s-t graph

Figure 7.1: In subfigure (a), $V = \{v_1, v_2, v_3, v_4\}$ represents four utterances, which might be (from top to bottom) "yes", "oh yes", "oh right, right" and "right"; $F = \{f_1, f_2, f_3\}$ is the vocabulary, i.e., f_1, f_2, f_3 represent words "yes", "oh" and "right" respectively. For $X = \{v_3, v_4\}$, $\gamma(X) = \{f_2, f_3\}$. For $Y = \{f_1\}$, $\zeta(Y) = \{v_1\}$. In (b), the s-t graph corresponding to Eq. 7.6.

$(w(X))$ and minimizes the vocabulary size $(\Gamma(X))$. Note that we *optimize* the vocabulary size instead of having hard constraints on it. By changing the value of λ, we can produce corpora with different vocabulary sizes, where each vocabulary size is determined during the optimization process and is optimal in terms of balancing with the amount of the associated data. In general, the larger λ is, the smaller the resulting vocabulary size will be.

There are at least two advantages of formulating the corpus subset selection problem as maximizing Eq. 7.2. First of all, this allows us to express the problem of corpus subset selection in a variety of flexible ways, suiting the needs of an individual novel ASR system its designer. Secondly, unlike the problem setup in (King *et al.* , 2005) where only a sub-optimal solution is available, our optimization problem can be solved exactly and efficiently by leveraging techniques of submodular function minimization. We discuss these two aspects in more detail in the following two sections.

7.4 Objective Functions

When designing a corpus for novel ASR system development, certain properties of the data as well as certain word forms might be preferred, both of which can be easily modeled in our approach by using different forms of the objective function (Eq. 7.2). In particular, we may have different w functions depending on our needs. For instance, when w is the cardinality function, i.e. $w(X) = \sum_{v \in X} 1 = |X|$, it measures the number of the utterances. In this case, maximizing Eq. 7.2 will give us a corpus that favors a large number of utterances. We can also have a weight on each utterance where the weight indicates how important the corresponding utterance is, i.e., we can have $w(X) = \sum_{v \in X} s_v$, where s_v is the weight for utterance x. When s_v represents the speech time-length of utterance x, $\sum_{x \in X} s_v$ measures the total speech duration of the utterances X, in which by maximizing Eq. 7.2, we will have a bias on utterances that contain more speech. On the other hand, we can also have different forms of $\Gamma(X)$ to impose a bias against certain word forms. First, we define $\gamma(X)$ to be the distinct words that appear in utterances X. That is

$$\gamma(X) \triangleq \{f \in F : \exists v \in X \text{ s.t. } (v, f) \in E\}. \tag{7.3}$$

In other words, $\gamma(X)$ is the vocabulary of utterances (corpus) X, and the cardinality of $\gamma(X)$ (i.e., $|\gamma(X)|$) is the size of the vocabulary (see Figure 7.1(a)). Then we can have $\Gamma_1(X) = |\gamma(X)|$, which represents the collective vocabulary size of utterances in set X. We can also have

$$\Gamma_2(X) = \sum_{f \in \gamma(X)} p_f, \tag{7.4}$$

where p_f indicates the unimportance of word f. A larger p_f states that word f is less important. This allows certain desirable properties of the vocabulary of the resultant corpus to be expressed (e.g., words with more syllables might be preferred). Note if $p_f = 1, \forall f$, then $\Gamma_2 = \Gamma_1$.

In some case, incorporation of stop words (e.g., “about”, “after”, “all”, “am”, etc.) is less desirable compared to content words that convey the semantic meaning of an utterance. Let (F_s, F_f) be a partition of F into stopwords F_s and F_f non-stop (function) words $F_f = F \backslash F_s$.

We can use the above modular function (i.e. m in Γ_2) to put a preference against stop words. I.e., for any $f \in F_s$ and $f' \in F_f$, we would have that $m(f) \geq m(f')$. So far (empirically) this has ended up performing poorly since it effectively reduces the vocabulary size down only to the function words.

Alternatively, we can use a submodular function that has a preference away from stop words — the function is a mixture of a modular function over stop words and concave-submodular function over content words. I.e., we define

$$\Gamma_3(X) = m(\gamma(X) \cap F_s) + \alpha\sqrt{m(\gamma(X) \cap F_f)} \tag{7.5}$$

where α is a tradeoff coefficient. This function is such that once we start including non-stop words, they become cheaper but the stop words retain their expense.

By using different forms of objective function, corpora suiting different needs can be created. Table 7.1 illustrates several objective functions that we used in our experiments, where for instance, Corpus D was created by maximizing the total speech duration in the resultant corpus while limiting the vocabulary with preference for words with more syllables.

7.5 Algorithm

Note that maximizing Eq. (7.2) is identical to finding X that minimizes

$$L(\lambda, X) \triangleq w(V \setminus X) + \lambda\Gamma(X). \tag{7.6}$$

For a given λ, if $L(\lambda, X)$ is a submodular function on X, then $\min_{X \subseteq V} L(\lambda, X)$ can solved exactly in polynomial time (Fujishige, 2005a) Fortunately, all the aforementioned w functions are modular (both submodular and supermodular), and all Γ functions are submodular, making $L(\lambda, X)$ submodular in all our cases. Therefore we can solve our corpus creation problem optimally by leveraging submodular function minimization techniques.

To create a corpus with the desired property (e.g., a fixed upper limit on vocabulary size), different values of the trade-off coefficient λ must be tried, where multiple calls of the optimization algorithm are required. In other words, we do not have direct control over the vocabulary constraint, only indirect control via λ. In our case, however, *all* possible solutions for $\min_{X \subseteq V} L(\lambda, X)$ for *all* possible $\lambda \geq 0$ can be found in the same complexity

as the complexity of solving $\min_{X \subseteq V} L(\lambda, X)$ for a *single* λ, and moreover there are only a finite (no more than $|V|$) number of distinct values of λ that makes a difference, all thanks to submodularity.

Finding all distinct λ values (and minimizing sets) can be done using parametric submodular function minimization. When using a push-relabel framework (Fleischer & Iwata, 2000), finding solutions for all λ requires only the same asymptotic running time as a *single* submodular function minimization. Note that the push-relabel framework was firstly introduced in (Gallo *et al.* , 1989) for network flow (graph cut) problems.

Now, interestingly, the submodular functions used in this chapter are all graph-representable. In other words, we can convert our submodular minimization problem to the problem of finding minimum s-t cuts in a graph, and therefore a fast parametric flow algorithm (Gallo *et al.* , 1989) can be used to find all the solutions for the corpus creation problem for all possible values trade-off coefficient λ, while only requiring the computational complexity of running a single minimum s-t cut, which can be solved very efficiently even on very large graphs.

We next describe how we convert our minimization problem into a minimum s-t graph cut problem. Take $\min_{X \subseteq V} w(V \setminus X) + \lambda \Gamma_2(X)$ for example. We build an s-t graph as follows. Add a source node s and connect it to every node $v \in V$ with weight t_v. Add a sink node t and connect to it every node in $f \in F$ with weight λp_f. Use an infinite weight for every original graph edge $(v, f) \in E$. Now in such a graph, due to the max-flow/min-cut theorem, any minimum cut cannot have any edge $(v, f) \in E$ since that edge has infinite capacity, and therefore, any minimum cut will consist of either:

1. all the edges $\{(s, v) : v \in V\}$ having a cut value of $\sum_{v \in V} t_v = w(V)$;

2. all of the edges $\{(f, t) : f \in F\}$ having a cut value of $\lambda \sum_{f \in F} p_f = \lambda \Gamma_2(V)$; or

3. the edges $\{(s, v) : v \in V \setminus X\} \cup \{(f, t) : f \in \gamma(X)\}$, with a cut value of $\sum_{v \in V \setminus X} t_v + \lambda \sum_{f \in \gamma(X)} p_f = w(V \setminus X) + \lambda \Gamma_2(X)$.

Note that the first case has $X = \emptyset$, and the second case $X = V$, both of which are special

Table 7.1: Objective functions used for Corpora A, B, C and D. t_v and s_v are the number of word tokens, and the duration of speech in utterance v respectively. q_f is the number of phonemes in the pronunciation of word f.

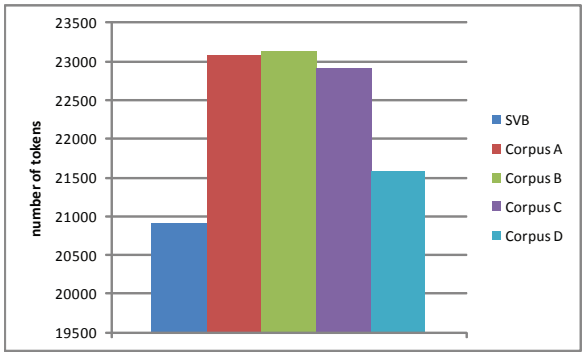
Corpus ID	Objective Function	
	$w(X)$	$\Gamma(X)$
A	$\|X\|$	$\|\gamma(X)\|$
B	$\sum_{v \in X} t_v$	$\|\gamma(X)\|$
C	$\sum_{v \in X} s_v$	$\|\gamma(X)\|$
D	$\sum_{v \in X} s_v$	$\sum_{f \in \gamma(X)} \frac{100}{q_f}$

cases of the third case. The transformation is shown in Figure 7.1(b). Therefore, finding the minimum s-t cut will minimize our objective $w(V \setminus X) + \lambda\Gamma_2(X)$.

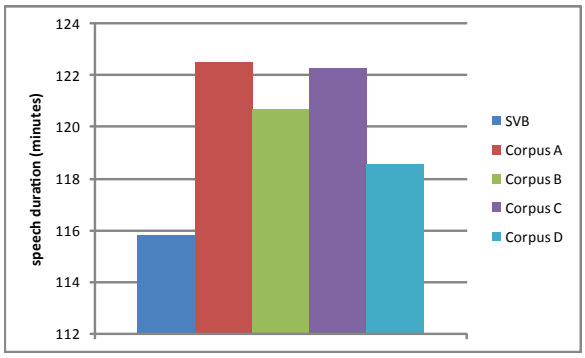
7.6 Experiments

We tested our corpus subset selection approach on Switchboard I. To be comparable with (King *et al.* , 2005), we followed the same experimental setup. In particular, each side of the long conversations in the Switchboard-I corpus was divided into shorter segments. The initial cuttings used were based on the segmentations produced by Mississippi State University (Ganapathiraju *et al.* , 1998). These segments were further divided into smaller utterances at every silence longer than 500ms. The resulting utterances were pruned by removing those containing disfluency and filler-model words: i.e., all word fragments (e.g. sim[ilar]-), words ending in a digit, uh, [noise], i-, yeah, [laughter], huh, hm, [laughter-*], uh-huh, um-hum hum, huh-uh, um.

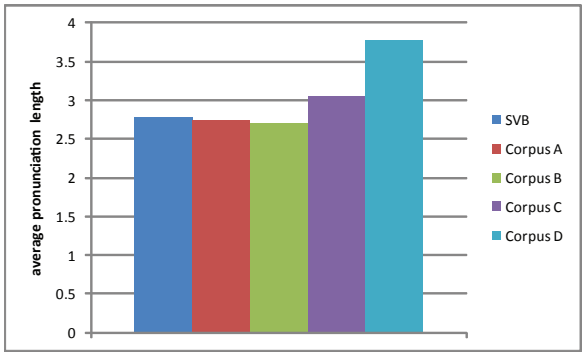
We investigated several types of w and Γ function, and produced corpora with different properties. In particular, we created four corpora (corpus A, B, C and D) using objective functions illustrated in Table 7.1, and compare them to SVitchboard (SVB, corpora created using the greedy algorithm (King *et al.* , 2005).) Note that in our approach the vocabulary sizes of the resulting corpora are naturally determined by the objective rather than predefined, and we choose those that are as close as possible to the SVB vocabulary sizes (10, 25, 50, 100, and 500) for comparison.



(a) toal number of tokens



(b) total speech duration (minutes)



(c) average number of phones per pronunciation

Figure 7.2: Corpora statistics on the total number of tokens in the utterances, the duration of speech, and the average number of phones in the pronunciations of the corpus vocabulary. The plots correspond to a vocabulary size of 50.

Corpus A was produced with the objective function that maximizes the number of utterances while restricting the vocabulary size. It turns out that corpus A indeed includes more utterances compared to other corpora (created with objectives that are not maximizing the number of utterances) given the same vocabulary size. For instance, with vocabulary size 10, corpus A contains 7615 utterances while SVB has 6775. It even contains more utterances (26165 vs. 23670) with a smaller vocabulary (489 vs. 500), compared to SVB.

Corpora B and C were produced in a similar way as corpus A except that utterance weights were used. In particular, for corpus B, each utterance was weighted by the number of tokens it contains (i.e. $\sum_{x \in X} t_x$ measures the number of tokens in utterances X), while in corpus C, each utterance was weighted by the duration of the non-silence speech (i.e. $\sum_{x \in X} s_x$ measures the total speech duration of X). The resulting corpora B and C do have the desired property. As illustrated in Figure 7.2(a), with vocabulary size 50, there are 23124 tokens in corpus B, which is larger than those in SVB, or in corpora C and D; there are about 122 minutes of speech in corpus C, which is more than those in SVB, or in corpora B and D, as shown in Figure 7.2(b). Note that for corpus A, the vocabulary size closest to 50 is 51, but all the remaining corpora have a λ yielding a vocabulary size of exactly 50. While corpus C (at 50 types) is optimized for speech duration, corpus A is slightly larger in this measure, due to it having a 51 types. Corpus C, however, also had a λ corresponding to 51 types and in this case, a 51-type corpus C had a total speech duration of 123.17, beating the 51-type corpus A with its total speech duration of 122.49.

In some situations, we may be concerned not only with the amount of data and the vocabulary size, but also wish for the resulting vocabulary to possess certain properties. Our method can handle such scenarios naturally and efficiently. For instance, a corpus with a limited vocabulary but rich phonetic variety could be useful for research on novel pronunciation modeling, and this is how we produced corpus D. We approximated the phonetic richness by the number of phoneme tokens in the pronunciation of a word (Ganapathiraju *et al.* , 1998). We used an objective function as shown in Table 7.1. Intuitively, by using this objective function, words with more phones will have a lower weight and therefore a higher chance of being selected. The resulting corpus D is well balanced in terms of corpus size and vocabulary variety. Compared to SVB at vocabulary size 50, corpus D not only has a

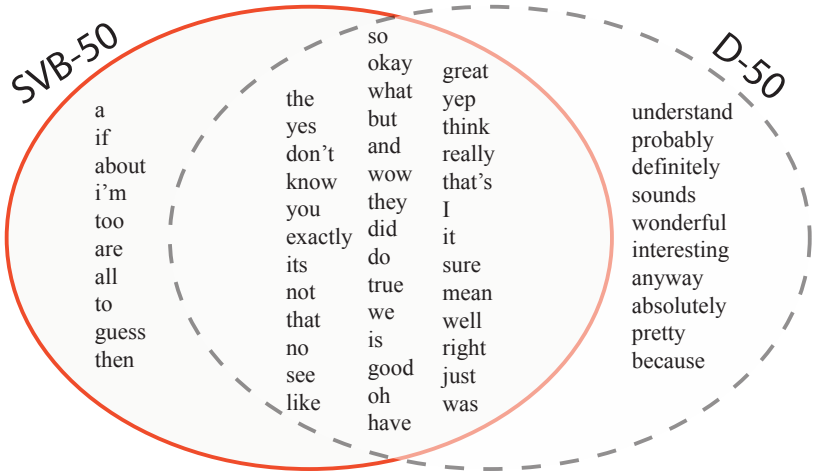


Figure 7.3: Venn diagram showing the vocabulary difference between SVB-50 (King *et al.* , 2005) and our D-50.

vocabulary with longer average pronunciation length (Figure 7.2(c)) but also includes more tokens (Figure 7.2(a)) and more acoustic speech (Figure 7.2(b)). We compare the complete vocabulary of SVB-50 and D-50 using a Venn diagram in Figure 7.3, clearly showing that D-50 has a richer lexicon (with words like "absolutely" and "definitely").

7.7 Conclusions

We have presented a framework for selecting a limited vocabulary subset of a large speech corpora. We show that a previous approach (King *et al.* , 2005) for this purpose is theoretically unjustified, while the selection process in our new approach is optimal in the sense that formulating the subset selection problem as an optimization problem over submodular functions leads to an optimal solution. Our approach can be scaled to very large initial corpora, thanks to the efficiency of finding a minimum graph cut. Experimental results on Switchboard show that our approach indeed produces limited vocabulary corpora with both additional and richer acoustic data.

Chapter 8

CONCLUSION AND FUTURE WORK

This chapter summarizes the main contribution of this dissertation and suggests directions for future research.

8.1 Main Contribution

The contributions of this thesis are tailored into two types. On the algorithm side, the contributions are

- A modified greedy algorithm for submodular maximization with knapsack constraints is presented. The algorithm is cheap enough to be applicable to large scale NLP problems. Moreover, we generalize Khuller *et al.* (1999)'s theoretical result on optimizing set cover function under knapsack constraint to budgeted optimization problem with any monotone submodular objectives, and show that our modified algorithm is a $(1 - \frac{1}{\sqrt{e}})$-approximation algorithm for budgeted maximization of a monotone submodular function. A scaling parameter on the cost of ground set element is also introduced to account for the uncalibrated issue in the ratio-based greedy heuristic (ratio of objective function gain over the cost), and we also show that with non-unity scaling parameter, performance of the proposed algorithm is still bounded.

- Minimum-norm-point (MN) algorithm is the widely regarded as the most efficient algorithm for general submodular function minimization. In this thesis, we offer the first empirical study of the complexity of MN algorithm when applied to a real-world application scale problem of practical interest. We show that the empirical complexity of MN algorithm on a particular type of submodular functions that arises in practice is not as good as the complexity of the combinatorial algorithms for submodular functions minimization, giving some implications for the theoretical analysis of the

worse-case complexity of MN algorithm (as the complexity of MN algorithm is still an open question). Moreover, acceleration methods are proposed which speed up MN algorithm phenomenally in practise.

- A pilot study is presented for the submodular knapsack problem. We show the hardness of this problem, with a theorem saying that there is no (ρ, σ)-bicriteria approximation algorithm for submodular knapsack problem, even with monotone submodular function, for any ρ and σ with $\frac{\rho}{\sigma} = o\left(\sqrt{\frac{\ln n}{n}}\right)$ where n is the ground set size.

- Learning algorithms for learning submodular mixtures are presented, with contribution on analysis of the performance of approximate learning when approximate inference (decoding) is used. In particular, we show that the risk of approximate learning is bounded by the risk of exact learning where exact inference is used.

Application-wise, the contributions are:

- We show that submodularity naturally arises in document summarization. In particular, we show that a lot of well-established methods for document summarization, including the widely used maximum marginal relevance (Carbonell & Goldstein, 1998) method, are actually corresponding to submodular function optimization, without being realized for decades before this thesis. Moreover, we prove that the widely-used ROUGE score (Lin, 2004) for automatic summarization evaluation, and the score used in recently popular Pyramid evaluation (Nenkova & Passonneau, 2004), are both monotone submodular, giving further evidence that submodular functions are natural fit for the score function of document summarization.

- We introduce a class of submodular functions that is not only monotone but also models relevance and diversity simultaneously for document summarization. This class of submodular functions is further generalized to a mixture of submodular components, where each component might either models the relevance or models the diversity, and might differs either in function forms or in function parameters. When evaluated on the standard benchmark task for document summarization, namely Document

Understanding Conference (DUC), we achieve *best results ever reported* on DUC-2004, DUC-2005, DUC-2006, and DUC-2007.

- We provide a new angle of viewing word alignment problem in machine translation by casting it as an optimization problem over matroid constraints. We show that submodularity naturally arises when modeling word fertility. Experiments on the English-French Hansards alignment task show that our approach achieves lower alignment error rates compared to conventional matching based approaches as well as an ILP based approach.

- We address the problem of finding a subset of a large training data set (corpus) that is useful for accurately and rapidly prototyping novel and computationally expensive machine learning architectures. To solve this problem, we express it as an minimization problem over a weighted sum of modular functions and submodular functions. Quantities such as number of classes (or quality) in a set of samples, or quality of a bundle of classes are submodular functions which make finding the optimal solutions possible. We apply the principal partition to our problem such that solutions for all possible trade-offs between a modular function and a submodular function can be found efficiently. We show results for speech recognition on the Switchboard-I speech recognition corpus, demonstrating improved results over previous techniques for this purpose. We also demonstrate the variety of the resulting corpora that may be produced using our method.

8.2 Future work

From the theoretical and algorithmic perspective, there are several open questions for the work that presented in this thesis.

First, we have empirically shown some hard examples for minimum-norm-point algorithm for submodular minimization. The open question is what is the complexity of MN algorithm. Is it the case that, similar to the simplex algorithm for linear programing (Klee & Minty, 1972), MN algorithm is efficient in practise but requires exponentially many steps

in the worst case? This might be a hard question to answer, but on the other hand, it might be possible to show some lower bound of the MN's complexity by constructing a submodular function such that the runtime of MN algorithm on this function can be analytically analyzed. For instance, the Iwata's function is known to be ideal for MN algorithm (Fujishige & Isotani, 2011), since only two major iterations, regardless the ground set size, are needed for the MN when minimizing this function. Similarly, it might be possible to construct a least ideal function for MN, upon whom the number of MN's major iterations can be analytically analyzed. The hard examples we found in this thesis might give some clue on constructing such examples. Regarding to the acceleration method that we propose, the open question is whether there is a general way to identify the resolution of a submodular function. Also, it has been shown that warm-start usually helps for many optimization problem, and the question is then whether there is some effective warm-start strategy for MN algorithm as well.

Second, we have addressed the problem of approximate learning for submodular mixtures by theoretical analysis the learning performance. The analysis depends on the fact that the loss augmented inference can be solved approximately and efficiently, which requires the loss augmented inference objective to be submodular. However, for many applications arises in practise, the loss function that best models the problem is supermodular (e.g., the ROUGE score for summarization), or neither submodular nor supermodular. The question is then: is there a general way to handle a loss augmented objective that is not submodular, while at the same time preserves the performance guarantee? One possible direction of research for this problem is to use submodular surrogates for the true loss, analogs to the convex surrogates for non-convex loss functions that people have been using.

Application-wise, there are some immediate applications from what have been shown in this thesis. The approaches we proposed to query-focused summarization can be directly applied to information retrieval. In particular, there is a growing interest on *diversifying search results*. The motivation is that when an ambiguous query is received, a sensible approach is to diversify the results retrieved for this query, in the hope that at least one of the interpretations of the query intent will satisfy the user. In this scenario, our query-focused summarization approach is directly applicable since the goals are exactly the same:

finding the most relevant documents and at the same time being as diverse as possible.

In general, many beautiful theory and algorithms have been developed for the submodular optimization, NLP problems are essentially combinatorial optimization problems, and we believe that there are many more NLP problems can benefit from submodularity and submodular optimizations. What we have shown, e.g., the success in document summarization by leveraging submodularity, is just a tip of an iceberg. We am very excited at the prospect of continuing research on exploring submodularity in natural language processing.

BIBLIOGRAPHY

Ageev, AA, & Sviridenko, MI. 2004. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, **8**(3), 307–328.

Agrawal, R., Gollapudi, S., Halverson, A., & Ieong, S. 2009. Diversifying search results. *Pages 5–14 of: Proceedings of the Second ACM International Conference on Web Search and Data Mining*. ACM.

Allan, J., Wade, C., & Bolivar, A. 2003. Retrieval and novelty detection at the sentence level. *Page 321 of: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*.

Anderson, B., & Moore, AW. 2005. Active learning for hidden markov models: Objective functions and algorithms. *Page 9 of: Machine Learning-International workshop*, vol. 22.

Bach, F. 2010. Structured sparsity-inducing norms through submodular functions. *Advances in Neural Information Processing Systems*.

Bach, F. 2011. Learning with Submodular Functions: A Convex Optimization Perspective. *Arxiv preprint arXiv:1111.6453*.

Balcan, M.F., & Harvey, N.J.A. 2011. Learning submodular functions. *Pages 793–802 of: Proceedings of the 43rd annual ACM symposium on Theory of computing*. ACM.

Bilmes, Jeff. 2008. *Fisher Kernel for DBN*. Tech. rept. University of Washington.

Bilmes, Jeff, & Bartels, Chris. 2005. Graphical Model Architectures for Speech Recognition. *IEEE Signal Processing Magazine*, **22**(5), 89–100.

Brin, S., & Page, L. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, **30**(1-7), 107–117.

Brown, P.F., Pietra, V.J.D., Pietra, S.A.D., & Mercer, R.L. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, **19**(2), 263–311.

Calinescu, G., Chekuri, C., Pál, M., & Vondrák, J. 2007. Maximizing a submodular set function subject to a matroid constraint. *Pages 182–196 of: Proc. of 12th IPCO*. Citeseer.

Carbonell, J., & Goldstein, J. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. *In: Proc. of SIGIR*.

Celikyilmaz, A., & Hakkani-tür, D. 2010. A Hybrid Hierarchical Model for Multi-Document Summarization. *Pages 815–824 of: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics.

Cesa-Bianchi, N., Conconi, A., & Gentile, C. 2001. On the Generalization Ability of On-Line Learning Algorithms. *Pages 359–366 of: NIPS*.

Chapelle, Olivier, Ji, Shihao, Liao, Ciya, Velipasaoglu, Emre, Lai, Larry, & Wu, Su-Lin. 2011. Intent-based diversification of web search results: metrics and algorithms. *Information Retrieval*, May, 1–21.

Charniak, E., & Johnson, M. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. *Pages 173–180 of: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Chelba, C., Brants, T., Neveitt, W., & Xu, P. 2010 (September). Study on interaction between entropy pruning and Kneser-Ney smoothing . *In: Proc. of Interspeech*.

Clarke, J., & Lapata, M. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, **31**(1), 399–429.

Cohn, D., Atlas, L., & Ladner, R. 1994. Improving generalization with active learning. *Machine Learning*, **15**(2), 201–221.

Collins, M. 2000. Discriminative Reranking for Natural Language Parsing. *Pages 175–182 of: Proc 17th International Conf on Machine Learning*, vol. 31.

Collins, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Pages 1–8 of: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics.

Collins, P.L.B.M., & McAllester, B.T.D. 2004. Exponentiated gradient algorithms for large-margin structured classification. *Page 113 of: Advances in neural information processing systems*, vol. 17.

Conn, A.R., & Cornuejols, G. 1990. A projection method for the uncapacitated facility location problem. *Mathematical programming*, **46**(1), 273–298.

Conroy, J.M., Schlesinger, J.D., Goldstein, J., & O'leary, D.P. 2004. Left-brain/right-brain multi-document summarization. *In: Proceedings of the Document Understanding Conference (DUC 2004)*.

Cornuejols, G., Fisher, M., & Nemhauser, G.L. 1977. On the uncapacitated location problem. *Pages 163–177 of: Studies in Integer Programming: Proceedings of the Institute of Operations Research Workshop*, vol. 1. North Holland.

Cover, T.M., & Thomas, J.A. 2006. *Elements of information theory*. Wiley-Interscience.

Culotta, A., & McCallum, A. 2005. Reducing labeling effort for structured prediction tasks. *In: Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Cunningham, W.H. 1984. Testing membership in matroid polyhedra. *Journal of Combinatorial Theory, Series B*, **36**(2), 161–188.

Dagan, I., & Engelson, S.P. 1995. Committee-based sampling for training probabilistic classifiers. *Pages 150–157 of: ICML*. Morgan Kaufmann.

Dang, H.T. 2005. Overview of DUC 2005. *In: Proceedings of the Document Understanding Conference*.

Daumé, H., Langford, J., & Marcu, D. 2009. Search-based structured prediction. *Machine learning*, **75**(3), 297–325.

Daumé III, H., & Marcu, D. 2006. Bayesian query-focused summarization. *Page 312 of: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*.

Dughmi, S. 2009. Submodular Functions: Extensions, Distributions, and Algorithms. A Survey. *Arxiv preprint arXiv:0912.0322*.

Edmonds, J. 1970. *Combinatorial Structures and their Applications*. Gordon and Breach. Chap. Submodular functions, matroids and certain polyhedra, pages 69–87.

Erkan, G., & Radev, D.R. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, **22**, 457–479.

Fadaei, Salman, Safari, MohammadAli, & Fazli, MohammadAmin. 2011. Maximizing Submodular Set Functions Subject to Different Constraints: Combined Algorithms. *CoRR*.

Feige, U. 1998a. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, **45**(4), 634–652.

Feige, U. 1998b. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, **45**(4), 634–652.

Feige, U., Mirrokni, V., & Vondrak, J. 2007. Maximizing non-monotone submodular functions. *In: Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.

Fellbaum, C. 1998. *WordNet: An electronic lexical database*. The MIT press.

Filatova, E., & Hatzivassiloglou, V. 2004. Event-based extractive summarization. *In: Proceedings of ACL Workshop on Summarization*, vol. 111.

Finley, T., & Joachims, T. 2008. Training structural SVMs when exact inference is intractable. *Pages 304–311 of: Proceedings of the 25th international conference on Machine learning*. ACM.

Fisher, M.L. 1981. The Lagrangian relaxation method for solving integer programming problems. *Management science*, 1–18.

Fisher, ML, Nemhauser, GL, & Wolsey, LA. 1978. An analysis of approximations for maximizing submodular set functions—II. *Polyhedral combinatorics*, 73–87.

Fleischer, L., & Iwata, S. 2000. A push-relabel framework for submodular function minimization and applications to parametric opt imization. *Page 116 of: Symposium on Theory of Computing*, vol. 107.

Ford, L.R., & Fulkerson, D.R. 1958. Network ow and systems of representatives. *Canadian Journal of Mathematics*, **10**.

Fujii, A., Tokunaga, T., Inui, K., & Tanaka, H. 1998. Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, **24**(4), 573–597.

Fujishige, S. 2005a. *Submodular functions and optimization*. Elsevier Science Ltd.

Fujishige, S. 2005b. *Submodular Functions and Optimization*. 2nd edn. Annals of Discrete Mathematics, no. 58. Elsevier Science.

Fujishige, S., & Isotani, S. 2011. A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, **7**, 3–17.

Fujishige, S., Hayashi, T., & Isotani, S. 2006. The minimum-norm-point algorithm applied to submodular function minimization and linear programming. *Research Institute for Mathematical Sciences Preprint RIMS-1571, Kyoto University, Kyoto Japan*.

Gallo, G., Grigoriadis, M. D., & Tarjan, R. E. 1989. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, **18**(1), 30–55.

Ganapathiraju, N., Deshmukh, A., Gleeson, A., Hamakera, A., & Pic one, J. 1998. Resegmentation of SWITCHBOARD. *In: Proc. ICSLP*.

Gharan, Shayan Oveis, & Vondrák, Jan. 2011. Submodular Maximization by Simulated Annealing. *Pages 1098–1116 of: SODA*.

Gillick, Dan, Riedhammer, Korbinian, Favre, Benoit, & Hakkani-Tür, Dilek. 2009. A Global Optimization Framework for Meeting Summarization. *In: Proc. of ICASSP.*

Goel, G., , Karande, C., Tripathi, P., & Wang, L. 2009. Approximability of Combinatorial Problems with Multi-agent Submodular Cost Functions. *In: FOCS.*

Goemans, M.X., & Williamson, D.P. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, **42**(6), 1115–1145.

Goemans, M.X., Harvey, N.J.A., Iwata, S., & Mirrokni, V. 2009. Approximating submodular functions everywhere. *Pages 535–544 of: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics.

Grötschel, M., Lovász, L., & Schrijver, A. 1981. The Ellipsoid Algorithm and its consequences in combinatorial optimization. *Combinatorica*, **1**, 499–513.

Grötschel, M., Lovász, L., & Schrijver, A. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Pages 169–197 of: Combinatorica*, vol. 1. Springer.

Guillory, Andrew, & Bilmes, Jeff. 2010. Interactive Submodular Set Cover. *In: International Conference on Machine Learning (ICML).*

Gupta, A., Roth, A., Schoenebeck, G., & Talwar, K. 2010. Constrained Non-Monotone Submodular Maximization: Offline and Secretary Algorithms. *Arxiv preprint arXiv:1003.1517.*

Haghighi, A., & Vanderwende, L. 2009. Exploring Content Models for Multi-Document Summarization. *Pages 362–370 of: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Boulder, Colorado: Association for Computational Linguistics.

Hakkani-tür, Dilek, & Gorin, Allen. 2002. Active learning for automatic speech recognition. *Pages 3904–3907 of: in Proceedings of the ICASSP.*

Hoi, S.C.H., Jin, R., Zhu, J., & Lyu, M.R. 2006. Batch mode active learning and its application to medical image classification. *Pages 417–424 of: ICML*. ACM New York, NY, USA.

Huang, L., & Sagae, K. 2010. Dynamic programming for linear-time incremental parsing. *Pages 1077–1086 of: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Iwata, S. 2003. A Faster Scaling Algorithm for Minimizing Submodular Functions. *SIAM Journal on Computing*, **32**, 833.

Iwata, S., Fleischer, L., & Fujishige, S. 2001a. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, **48**(4), 761–777.

Iwata, S., Fleischer, L., & Fujishige, S. 2001b. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, **48**, 761–777.

Iwata, S., McCormick, S.T., & Shigeno, M. 2006. A strongly polynomial cut canceling algorithm for minimum cost submodular flow. *SIAM Journal on Discrete Mathematics*, **19**(2), 304–320.

Iwata, Satoru, & Nagano, Kiyohito. 2009. Submodular Function Minimization under Covering Constraints. *Pages 671–680 of: FOCS'09*.

Jaakkola, T.S., & Haussler, D. 1999. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, 487–493.

Jagarlamudi, J., Pingali, P., & Varma, V. 2006. Query independent sentence scoring approach to DUC 2006. *In: DUC 2006*.

Janin, Adam, Baron, Don, & et al., Jane Edwards. 2003. The ICSI Meeting Corpus. *In: Proc. of ICASSP*.

Jegelka, S., & Bilmes, J. A. 2011 (June). Submodularity beyond submodular energies: coupling edges in graph cuts. *In: Computer Vision and Pattern Recognition (CVPR)*.

Jegelka, S., Lin, H., & Bilmes, J. 2011. On fast approximate submodular minimization: Extended Version. *In: NIPS*.

Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming. *Pages 302–311 of: Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM.

Kelmans, BN, *et al.* . 1983. Multiplicative submodularity of a matrix's principal minor as a function of the set of its rows and some combinatorial applications. *Discrete Mathematics*, **44**(1), 113–116.

Kempe, D., Kleinberg, J., & Tardos, E. 2003. Maximizing the spread of influence through a social network. *In: Proceedings of the 9th Conference on SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Khachiyan, LG. 1979. Polinomialnyı algoritm v lineınom programmirovanii [Russian]. *Doklady Akademii Nauk SSSR*, **244**, 1093–1096.

Khuller, S., Moss, A., & Naor, J. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, **70**(1), 39–45.

King, S., Bartels, C., & Bilmes, J. 2005. SVitchboard 1: Small Vocabulary Tasks from Switchboard. *In: Ninth European Conference on Speech Communication and Technology*. ISCA.

Klee, V., & Minty, G.J. 1972. How Good Is the Simplex Algorithm? *Inequalities: proceedings*, **3**, 159.

Kolmogorov, V., & Zabin, R. 2004. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(2), 147–159.

Koo, T., Rush, A.M., Collins, M., Jaakkola, T., & Sontag, D. 2010. Dual decomposition for parsing with non-projective head automata. *Pages 1288–1298 of: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Krause, A., & Guestrin, C. 2005a. A note on the budgeted maximization of submodular functions. *Technical Rep. No. CMU-CALD-05*, **103**.

Krause, A., & Guestrin, C. 2005b. Near-optimal nonmyopic value of information in graphical models. *In: Proc. of Uncertainty in AI*.

Krause, A., McMahan, H.B., Guestrin, C., & Gupta, A. 2008. Robust submodular observation selection. *Journal of Machine Learning Research*, **9**, 2761–2801.

Krause, Andreas R. 2008. *Optimizing Sensing: Theory and Applications*. Ph.D. thesis, Carnegie Mellon University.

Kruskal, J.B. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, **7**(1), 48–50.

Kulesza, A. 2009. Approximate learning for structured prediction problems. *UPenn WPE-II Report*.

Kulesza, A., Pereira, F., *et al.* . 2007. Structured learning with approximate inference. *Advances in neural information processing systems*, **20**, 785–792.

Kulik, A., Shachnai, H., & Tamir, T. 2009. Maximizing submodular set functions subject to multiple linear constraints. *Pages 545–554 of: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics.

Kupiec, J., Pedersen, J., & Chen, F. 1995. A trainable document summarizer. *Pages 68–73 of: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*.

Lacoste-Julien, S., Taskar, B., Klein, D., & Jordan, M.I. 2006. Word alignment via quadratic assignment. *Pages 112–119 of: Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics.

Lafferty, J., McCallum, A., & Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Pages 282–289 of: MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*.

Lamel, L.F., Kassel, R.H., & Seneff, S. 1989. Speech database development: Design and analysis of the acoustic-phonetic corpus. *In: Speech Input/Output Assessment and Speech Databases*. ISCA.

Lee, J., Mirrokni, V.S., Nagarajan, V., & Sviridenko, M. 2009a. Non-monotone submodular maximization under matroid and knapsack constraints. *Pages 323–332 of: Proceedings of the 41st annual ACM symposium on Symposium on theory of computing*. ACM New York, NY, USA.

Lee, J., Sviridenko, M., & Vondrák, J. 2009b. Submodular maximization over multiple matroids via generalized exchange properties. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 244–257.

Lee, J., Sviridenko, M., & Vondrák, J. 2009c. Submodular maximization over multiple matroids via generalized exchange properties. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 244–257.

Lee, K.F., & Hon, H.W. 1989. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**(11), 1641–1648.

Levenshtein, V.I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Pages 707–710 of: Soviet Physics Doklady*, vol. 10.

Lewis, D.D., & Gale, W.A. 1994. A sequential algorithm for training text classifiers. *Pages 3–12 of: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag New York, Inc. New York, NY, USA.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. *In: Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Lin, H., & Bilmes, J. 2010a. An Application of the Submodular Principal Partition to Training Data Subset Selection. *In: NIPS workshop on Discrete Optimization in Machine Learning.*

Lin, H., & Bilmes, J. 2010b (June). Multi-document Summarization via Budgeted Maximization of Submodular Functions. *In: North American chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL/HLT-2010).*

Lin, H., & Bilmes, J. 2011a (June). A Class of Submodular Functions for Document Summarization. *In: The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT).*

Lin, H., & Bilmes, J. 2011b. Optimal Selection of Limited Vocabulary Speech Corpora. *In: Proc. Interspeech.*

Lin, H., & Bilmes, J. 2011c (June). Word Alignment via Submodular Maximization over Matroids. *In: The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT).*

Lin, H., Bilmes, J., Vergyri, D., & Kirchhoff, K. 2007. OOV detection by joint word/phone lattice alignment. *Pages 478–483 of: IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU).*

Lin, H., Bilmes, J., & Xie, S. 2009 (December). Graph-based Submodular Selection for Extractive Summarization. *In: Proc. IEEE Automatic Speech Recognition and Understanding (ASRU).*

Lin, Hui, & Bilmes, Jeff A. 2009 (September). How to Select a Good Training-data Subset for Transcription: Submodular Active Selection for Sequences. *In: Proc. of Interspeech.*

Lovász, L. 1983. Submodular functions and convexity. *Mathematical programming-The state of the art,(eds. A. Bachem, M. Grotschel and B. Korte) Springer*, 235–257.

Martins, A.F.T., & Smith, N.A. 2009. Summarization with a joint model for sentence extraction and compression. *Pages 1–9 of: Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*. Association for Computational Linguistics.

Martins, A.F.T., Smith, N.A., & Xing, E.P. 2009a. Concise integer linear programming formulations for dependency parsing. *ACL*.

Martins, A.F.T., Smith, N.A., & Xing, E.P. 2009b. Polyhedral outer approximations with application to natural language parsing. *In: Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA.

Matusov, E., Zens, R., & Ney, H. 2004. Symmetric word alignments for statistical machine translation. *Page 219 of: Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics.

McCormick, S. T. 2006. *Submodular Function Minimization*. Elsevier. updated version 3a (2008).

McDonald, R. 2007. A study of global inference algorithms in multi-document summarization. *Lecture Notes in Computer Science*, **4425**, 557.

McLachlan, G.J., & Basford, K.E. 1988. Mixture models. Inference and applications to clustering. *Statistics: Textbooks and Monographs, New York: Dekker, 1988*, **1**.

Melamed, I.D. 2000. Models of translational equivalence among words. *Computational Linguistics*, **26**(2), 221–249.

Mihalcea, R. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. *In: Proceedings of the ACL 2004 (companion volume)*.

Mihalcea, R., & Pedersen, T. 2003. An evaluation exercise for word alignment. *Pages 1–10 of: Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond-Volume 3*. Association for Computational Linguistics.

Mihalcea, R., & Tarau, P. 2004. TextRank: bringing order into texts. *In: Proceedings of EMNLP*.

Mihalcea, R., Tarau, P., & Figa, E. 2004. PageRank on semantic networks, with application to word sense disambiguation. *In: Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*.

Minoux, M. 1978. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, 234–243.

MOSEK, A.S. 2010. The MOSEK optimization software. *Online at http://www. mosek. com*.

Murray, Gabriel, Renals, Steve, & Carletta, Jean. 2005. Extractive summarization of meeting recordings. *In: Proc. of Interspeech*.

Nagano, K., Kawahara, Y., & Aihara, K. 2011. Size-constrained Submodular Minimization through Minimum Norm Base. *In: ICML*.

Narasimhan, M., & Bilmes, J. 2005a. A Submodular-Supermodular Procedure with Applications to Discriminative Structure Learning. *In: Proc. Conf. Uncertainty in Artifical Intelligence*. Edinburgh, Scotland: Morgan Kaufmann Publishers.

Narasimhan, M., & Bilmes, J. 2005b. A submodular-supermodular procedure with applications to discriminative structure learning. *In: Proc. Conf. Uncertainty in Artifical Intelligence, Edinburgh, Scotland, July. Morgan Kaufmann Publishers*.

Narasimhan, M., & Bilmes, J. 2007 (January). Local Search for Balanced Submodular Clusterings. *In: Twentieth International Joint Conference on Artificial Intelligence (IJCAI07)*.

Narayanan, H. 1997a. *Submodular functions and electrical networks*. North-Holland.

Narayanan, H. 1997b. *Submodular Functions and Electrical Networks*. North-Holland, Amsterdam.

Nash-Williams, C.St.J.A. 1961. Edge-disjoint spanning trees of nite graphs. *The Journal of the London Mathematical Society*, **36**, 445–450.

Nemhauser, G.L., Wolsey, L.A., & Fisher, M.L. 1978. An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, **14**(1), 265–294.

Nenkova, A., & Passonneau, R. 2004. Evaluating content selection in summarization: The pyramid method. *Pages 145–152 of: Proceedings of HLT-NAACL*, vol. 2004.

Nguyen, H.T., & Smeulders, A. 2004. Active learning using pre-clustering. *In: ICML*. ACM New York, NY, USA.

Och, F.J., & Ney, H. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, **29**(1), 19–51.

Orlin, J. B. 2009. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, **118**(2), 237–251.

Oxley, J. 2011. *Matroid theory*. Vol. 21. Oxford Univ Pr.

Passonneau, R.J., Nenkova, A., McKeown, K., & Sigelman, S. 2005. Applying the pyramid method in DUC 2005. *In: Proceedings of the Document Understanding Conference (DUC 05), Vancouver, BC, Canada*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Petrov, S., Barrett, L., Thibaux, R., & Klein, D. 2006. Learning accurate, compact, and interpretable tree annotation. *Pages 433–440 of: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Pingali, P., Rahul, K., & Varma, V. 2007. IIIT Hyderabad at DUC 2007. *Proceedings of DUC 2007*.

Qazvinian, V., Radev, D.R., & Ozgür, A. 2010. Citation Summarization Through Keyphrase Extraction. *Pages 895–903 of: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*.

Queyranne, M. 1998. Minimizing symmetric submodular functions. *Mathematical Programming*, **82**(1), 3–12.

Ratliff, N., Bagnell, J.A., & Zinkevich, M. 2006a. Subgradient methods for maximum margin structured learning. *In: ICML Workshop on Learning in Structured Output Spaces*.

Ratliff, N.D., Bagnell, J.A., & Zinkevich, M.A. 2006b. (Online) Subgradient Methods for Structured Prediction. *In: AISTATS*, vol. 2007. Citeseer.

Ratnaparkhi, A. 1996. A maximum entropy model for part-of-speech tagging. *Pages 133–142 of: EMNLP*, vol. 1.

Ravikumar, P., Agarwal, A., & Wainwright, M.J. 2008. Message-passing for graph-structured linear programs: proximal projections, convergence and rounding schemes. *Pages 800–807 of: Proceedings of the 25th international conference on Machine learning*. ACM New York, NY, USA.

Reynar, J.C., & Ratnaparkhi, A. 1997. A maximum entropy approach to identifying sentence boundaries. *Pages 16–19 of: Proceedings of the fifth conference on Applied natural language processing*. Association for Computational Linguistics.

Riedhammer, K., Favre, B., & Hakkani-Tür, D. 2010. Long story short-Global unsupervised models for keyphrase based meeting summarization. *Speech Communication*.

Roth, D., & Yih, W. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. *In: In Proceedings of CoNLL-2004*.

Roth, D., & Yih, W. 2005. Integer linear programming inference for conditional random fields. *Page 743 of: Proceedings of the 22nd international conference on Machine learning*. ACM.

Roy, N., & McCallum, A. 2001. Toward optimal active learning through sampling estimation of error reduction. *Pages 441–448 of: ICML.*

Rush, A.M., Sontag, D., Collins, M., & Jaakkola, T. 2010. On dual decomposition and linear programming relaxations for natural language processing. *Pages 1–11 of: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics.

Scheffer, T., Decomain, C., & Wrobel, S. 2001. Active hidden markov models for information extraction. *Pages 309–318 of: Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis.* Springer-Verlag London, UK.

Schrijver, A. 2000a. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, **80**(2), 346–355.

Schrijver, A. 2000b. A combinatorial algorithm minimizing submodular functions in strong ly polynomial time. *J. Combin. Theory Ser. B*, **80**, 346–355.

Schrijver, A. 2003. *Combinatorial optimization: polyhedra and efficiency.* Springer Verlag.

Settles, Burr. 2009. *Active Learning Literature Survey.* Tech. rept. University of Wisconsin–Madison.

Settles, Burr, & Craven, Mark. 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. *In: EMNLP.*

Shapley, L.S. 1971. Cores of convex games. *International Journal of Game Theory*, **1**(1), 11–26.

Shen, C., & Li, T. 2010. Multi-Document Summarization via the Minimum Dominating Set. *Pages 984–992 of: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010).* Beijing, China: Coling 2010 Organizing Committee.

Shor, N.Z., Kiwiel, K.C., & Ruszczynski, A. 1985. *Minimization methods for non-differentiable functions.* Springer-Verlag Berlin.

Sipos, R., Shivaswamy, P., & Joachims, T. 2012. Large-Margin Learning of Submodular Summarization Methods. *In: Proceedings of the 13th conference of the European chapter of the Association for Computational Linguistics*.

Stobbe, P., & Krause, A. 2010. Efficient Minimization of Decomposable Submodular Functions. *In: NIPS*.

Sviridenko, M. 2004. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, **32**(1), 41–43.

Svitkina, Z., & Fleischer, L. 2008. Submodular approximation: Sampling-based algorithms and lower bounds. *Pages 697–706 of: Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*. IEEE.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., & Rother, C. 2006. A comparative study of energy minimization methods for markov random fields. *Computer Vision–ECCV 2006*, 16–29.

Takamura, H., & Okumura, M. 2009. Text summarization model based on maximum coverage problem and its variant. *Pages 781–789 of: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Taskar, B., Guestrin, C., & Koller, D. 2004. Max-margin Markov networks. *Advances in neural information processing systems*, **16**, 25–32.

Taskar, B., Lacoste-Julien, S., & Klein, D. 2005a. A discriminative matching approach to word alignment. *Pages 73–80 of: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Taskar, B., Chatalbashev, V., Koller, D., & Guestrin, C. 2005b. Learning structured prediction models: A large margin approach. *Pages 896–903 of: Proceedings of the 22nd international conference on Machine learning*. ACM.

Taskar, B., Lacoste-Julien, S., & Jordan, M.I. 2006. Structured prediction, dual extragradient and Bregman projections. *The Journal of Machine Learning Research*, **7**, 1627–1653.

Topkis, D.M. 1998. *Supermodularity and complementarity.* Princeton Univ Pr.

Toutanova, K., Brockett, C., Gamon, M., Jagarlamudi, J., Suzuki, H., & Vanderwende, L. 2007. The PYTHY summarization system: Microsoft research at DUC 2007. *In: the proceedings of Document Understanding Conference.*

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. 2006. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, **6**(2), 1453.

Tür, G., Schapire, RE, & Hakkani-Tür, D. 2003. Active learning for spoken language understanding. *In: ICASSP*, vol. 1.

Tür, G., Hakkani-Tür, D., & Schapire, R.E. 2005. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, **45**(2), 171–186.

Varadarajan, Balakrishnan, Yu, Dong, Deng, Li, & Acero, Alex. 2009. Maximizing Global Entropy Reduction for Active Learning in Speech Recognition. *In: ICASSP.*

Vives, X. 2001. *Oligopoly pricing: old ideas and new tools.* The MIT press.

Vogel, S., Ney, H., & Tillmann, C. 1996. HMM-based word alignment in statistical translation. *Pages 836–841 of: Proceedings of the 16th conference on Computational linguistics-Volume 2.* Association for Computational Linguistics.

Vondrák, J. 2008. Optimal approximation for the submodular welfare problem in the value oracle model. *Pages 67–74 of: Proceedings of the 40th annual ACM symposium on Theory of computing.* ACM.

Vondrák, J. 2009. Symmetry and approximability of submodular maximization problems. *Pages 651–670 of: 2009 50th Annual IEEE Symposium on Foundations of Computer Science.* IEEE.

Vondrák, Jan. 2010. Polyhedral techniques in combinatorial optimization. *Lecture notes*.

Wainwright, MJ, Jaakkola, TS, & Willsky, AS. 2005. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, **51**(11), 3697–3717.

Wang, D., Zhu, S., Li, T., & Gong, Y. 2009. Multi-Document Summarization using Sentence-based Topic Models. *Pages 297–300 of: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Suntec, Singapore: Association for Computational Linguistics.

Welsh, D.J.A. 1976. *Matroid theory*. Vol. 5. Academic Press London, New York, San Francisco.

Whitney, H. 1935. On the Abstract Properties of Linear Dependence. *American Journal of Mathematics*, **57**(3), 509–533.

Wolfe, P. 1976. Finding the nearest point in a polytope. *Mathematical Programming*, **11**(1), 128–149.

Wolsey, L.A. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, **2**(4), 385–393.

Wu, Yi, Zhang, Rong, & Rudnicky, A. 2007 (Dec.). Data selection for speech recognition. *Pages 562–565 of: ASRU*.

Xie, Shasha, Favre, Benoit, Hakkani-Tür, Dilek, & Liu, Yang. 2009. Leveraging Sentence Weights in a Concept-based Optimization Framework for Extractive Meeting Summarization. *In: Proc. of Interspeech*.

Zhang, Justin Jian, Chan, Ho Yin, & Fung, Pascale. 2007. Improving Lecture Speech Summarization Using Rhetorical Information. *In: Proc. of ASRU*.

Zhao, P., Rocha, G., & Yu, B. 2009. Grouped and hierarchical model selection through composite absolute penalties. *Annals of Statistics*, **37**(6A), 3468–3497.

Zhu, Qifeng, Stolcke, Andreas, Chen, Barry, & Morgan, Nelson. 2005. Using MLP Features in SRI's Conversational Speech Recognition System. *In: Proc. of Interspeech*.

Zhu, Xiaodan, & Penn, Gerald. 2006. Summarization of spontaneous conversations. *In: Proc. of Interspeech.*

Zivný, S., & Jeavons, P.G. 2008. Classes of submodular constraints expressible by graph cuts. *Pages 112–127 of: Proceedings of the 14th International Conference on Principles and Practice of Contraint Programming (CP08)*, vol. 5202.

Appendix A

OMITTED PROOFS

Proof of f_{svb} is supermodular.

Proof. Note that

$$\zeta(Y \cup \{k\}) \setminus \zeta(Y) = \{v \in V : (v,k) \in E \\ \text{and } (v,f') \notin E, \forall f' \in F \setminus Y \cup \{k\}\}.$$

For any $A \subseteq B \subseteq F$ and $k \notin B$, we have

$$\zeta(A \cup \{k\}) \setminus \zeta(A) \subseteq \zeta(B \cup \{k\}) \setminus \zeta(B)$$

since if $\forall f' \in F \setminus A \cup \{k\}$, $(v,f') \notin E$, then it is also true that $\forall f' \in F \setminus B \cup \{k\}, (v,f') \notin E$. Therefore, we have

$$f_{\text{svb}}(A \cup \{k\}) - f_{\text{svb}}(A) = \sum_{i \in \zeta(A \cup \{k\}) \setminus \zeta(A)} m_i \\ \leq \sum_{i \in \zeta(B \cup \{k\}) \setminus \zeta(B)} t_i = f_{\text{svb}}(B \cup \{k\}) - f_{\text{svb}}(B)$$

□

Lemma 8. *if f is non-decreasing concave, for $a,b,c,d \in \mathbb{R}$ such that $a \leq b$, $c \leq d$, and $b - a \geq d - c$, we have $f(b) - f(a) \geq f(d) - f(c)$.*

Proof. Find b' such that $b' - a = d - c$. Since $b \geq b'$ and f is non-decreasing, we have

$$f(b) - f(a) \geq f(b') - f(a).$$

By Cauchy's mean value theorem, $\exists x \in [a,b'], \exists y \in [c,d]$ such that

$$f(b') - f(a) = (b' - a) f'(x)$$
$$f(d) - f(c) = (d - c) f'(y)$$

If $b' \leq c$, then $x \leq y$; we have $f'(x) \geq f'(y)$ because of concavity. Therefore $f(b')-f(a) \geq f(d) - f(c)$.

If $b' \geq c$, then $a \leq c \leq b' \leq d$, and $\exists x' \in [a, c], \exists y' \in [b', d]$ such that

$$f(c) - f(a) = f'(x')(c - a) = f'(x')(d - b') \geq$$
$$f'(y')(d - b') = f(d) - f(b').$$

Therefore, $f(b') - f(a) \geq f(d) - f(c)$ always holds, which indicates that

$$f(b) - f(a) \geq f(d) - f(c)$$

□

Proof of Theorem 8.

Proof. For any $A \subseteq B \subseteq V$ and $k \notin B$, by submodularity of f we have

$$f(A \cup \{k\}) - f(A) \geq f(B \cup \{k\}) - f(B).$$

Since f is non-decreasing, we have $f(A) \leq f(B)$. By Lemma 1, we have

$$g(f(A \cup \{k\})) - g(f(A))$$
$$\geq g(f(B \cup \{k\})) - g(f(B))$$

□

Proofs of $f_{\text{MMR-I}}$ is submodular.

Proof. Since $\sum_{i \in S} \texttt{Rel(i)}$ is modular, we only need to show $\sum_{i,j \in S, i<j} \texttt{Red}(i, j)$ is supermodular. Denote $\texttt{Red}(i, j)$ as $a_{i,j}$ for simplicity. Since $\sum_{i,j \in S, i<j} a_{i,j} = \frac{1}{2} \sum_{i,j \in S, i \neq j} a_{i,j}$, we have the increment of adding $k \in V \setminus S \setminus \{k\}$ to S as

$$\sum_{i,j \in S \cup \{k\}, i<j} a_{i,j} - \sum_{i,j \in S, i<j} a_{i,j} =$$
$$\frac{1}{2} \left(\sum_{i \in S \setminus \{k\}} a_{i,k} + \sum_{j \in S \setminus \{k\}} a_{k,j} \right)$$

which is monotone increasing. □

Proof of Theorem 18:

Proof. When f is not monotone, by submodularity, we have

$$f(S^*) \leq f(G_{i-1}) + \sum_{u \in S^* \setminus G_{i-1}} \rho_u(G_{i-1}) - \sum_{u \in G_{i-1} \setminus S^*} \rho_u(S^* \cup G_{i-1} - \{u\}). \tag{A.1}$$

Our proof still holds for non-monotone function if

$$\forall G_{i-1}, \sum_{u \in G_{i-1} \setminus S^*} \rho_u(S^* \cup G_{i-1} - \{u\}) \geq 0. \tag{A.2}$$

Since $G_1 \subset G_2 \subset ... \subset G$, by submodularity, the above (Eqn. (A.2)) is true if

$$\sum_{u \in G \setminus S^*} \rho_u(S^* \cup G - \{u\}) \geq 0. \tag{A.3}$$

Let A be the event that our algorithm performs near-optimally for non-monotone functions. We have:

$$\begin{aligned}
&\mathbf{Pr}(A) \\
\geq &\mathbf{Pr}\left(\sum_{u \in G \setminus S^*} \rho_u(S^* \cup G - \{u\}) \geq 0\right) \\
\geq &\mathbf{Pr}\left(\forall u \in G \setminus S^*, \rho_u(S^* \cup G - \{u\}) \geq 0\right) \\
= &\mathbf{Pr}\left(\bigwedge_{u \in G \setminus S^*} \rho_u(S^* \cup G - \{u\}) \geq 0\right) \\
= &1 - \mathbf{Pr}\left(\bigvee_{u \in G \setminus S^*} \rho_u(S^* \cup G - \{u\}) < 0\right) \\
\geq &1 - K\mathbf{Pr}\left(\rho_u(S^* \cup G - \{u\}) < 0, u \in G\right)
\end{aligned}$$

where the last inequality follows from the union bound and K is the largest possible size of a feasible solution (i.e. within the budget constraint).

In the following, we bound $\mathbf{Pr}(\rho_u(S) < 0)$ for our objective function f_{MMR}.

Let

$$\begin{aligned} X &= \rho_u(S) \\ &= \sum_{i \in V} w_{i,u} - (2\lambda + 2) \sum_{j \in S} w_{u,j} - (2\lambda + 1) w_{u,u} \\ &= \sum_{i \in V \setminus S} w_{i,u} - \alpha \sum_{j \in S} w_{j,u} \end{aligned}$$

where $\alpha = 2\lambda + 1$ and we assume $w_{u,u} = 0$ (i.e. self similarity does not count.)

Assume that all edges weights are bounded: $w_{i,j} \in [0, 1]$ (which is the case for cosine similarity). Also assume that edges weights are independently identically distributed with mean μ, i.e. $\mathbb{E}(w_{i,j}) = \mu$. We have

$$\mathbb{E}(X) = (|V| - (\alpha + 1)|S|)\mu. \tag{A.4}$$

Applying Hoeffding's inequality we have for any $t \geq 0$,

$$\mathbf{Pr}(X - \mathbb{E}(X) \leq -t) \leq \exp\left\{-\frac{2t^2}{|V| + (\alpha^2 - 1)|S|}\right\} \tag{A.5}$$

Let $t = \mathbb{E}(X)$ (which is greater than zero when $|S| \ll |V|$, we have

$$\mathbf{Pr}(X \leq 0) \leq \exp\left\{-\frac{2(|V| - (\alpha + 1)|S|)^2\mu^2}{|V| + (\alpha^2 - 1)|S|}\right\} \tag{A.6}$$

Since $|G \cup S^* - \{u\}| \leq 2K - 1 = \beta \ll |V|$, we have

$$\mathbf{Pr}(\rho_u(S^* \cup G - \{u\}) \leq 0) \leq \exp\left\{-\frac{2(|V| - (\alpha + 1)\beta)^2\mu^2}{|V| + (\alpha^2 - 1)\beta}\right\} \tag{A.7}$$

In sum, we have

$$\begin{aligned} \mathbf{Pr}(A) \geq & 1 - K\mathbf{Pr}\left(\rho_u(S^* \cup G - \{u\}) < 0, u \in G\right) && \text{(A.8)} \\ \geq & 1 - K \exp\left\{-\frac{2(|V| - (\alpha + 1)\beta)^2\mu^2}{|V| + (\alpha^2 - 1)\beta}\right\} && \text{(A.9)} \\ = & 1 - \exp\left\{-\frac{2(|V| - (\alpha + 1)\beta)^2\mu^2}{|V| + (\alpha^2 - 1)\beta} + \ln K\right\} && \text{(A.10)} \end{aligned}$$

□

Proof of Thorem 19.

Proof. $\forall A, B \in V$, we have

$$\begin{aligned}\mathcal{G}(A) + \mathcal{G}(B) &= f(\delta(A)) + f(\delta(B)) \\ &\geq f(\delta(A) \cup \delta(B)) + f(\delta(A) \cap \delta(B)) \\ &\geq f(\delta(A \cup B)) + f(\delta(A \cap B)) \\ &= \mathcal{G}(A \cup B) + \mathcal{G}(A \cap B),\end{aligned}$$

where the first and last equation is by definition in Eqn. 4.19, the first inequality is based on the submodularity of f, and the second inequality is due to the non-decreasing property of f.

Note that we use the fact that $\delta(A \cup B) = \delta(A) \cup \delta(B)$ and also that $\delta(A \cap B) \subseteq \delta(A) \cap \delta(B)$ (i.e., $\forall v \in A \cap B$, we have $\delta(v) \in \delta(A)$ and $\delta(v) \in \delta(B)$, and therefore $\delta(v) \in \delta(A) \cap \delta(B)$). However, $\delta(A \cap B) \supseteq \delta(A) \cap \delta(B)$ is not necessary true, so we are not able to achieve equality here and thus the non-decreasing condition on f is necessary to achieve this second inequality. □