Learning Mixtures of Submodular Shells with Application to Document Summarization

Hui Lin
University of Washington
hlin@ee.washington.edu

Jeff Bilmes
University of Washington
bilmes@ee.washington.edu

Abstract

We introduce a method to learn a mixture of submodular "shells" in a large-margin setting. A submodular shell is an abstract submodular function that can be instantiated with a ground set and a set of parameters to produce a submodular function. A mixture of such shells can then also be so instantiated to produce a more complex submodular function. What our algorithm learns are the mixture weights over such shells. We provide a risk bound guarantee when learning in a large-margin structured-prediction setting using a projected subgradient method when only approximate submodular optimization is possible (such as with submodular function maximization). We apply this method to the problem of multi-document summarization and produce the best results reported so far on the widely used NIST DUC-05 through DUC-07 document summarization corpora.

1 Introduction

Submodular functions [10] are those that satisfy the property of *diminishing returns*: given a finite ground set V, for any $A \subseteq B \subseteq V \setminus v$, a submodular function f must satisfy $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$, i.e., the incremental "value" of v decreases as the context in which v is considered grows from A to B. Submodular functions share a number of properties in common with convex and concave functions [29], 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, the weighted sum of a collection of submodular functions $\{f_i\}_i$, $f = \sum_i w_i f_i$ where w_i are nonnegative weights, is also submodular. While they have a long history in operations research, game theory, econometrics, and electrical engineering, they are still beginning to be studied in machine learning for a variety of tasks including sensor placement [19, 20], structure learning of graphical models [34], document summarization [27, 28] and social networks [18].

The problem of learning submodular functions has also recently been addressed. For example, in [15], it is asked "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 n is the ground set size and for what function $g : \mathbb{N} \to \mathbb{R}$ is the possible?". Among many results, they show that even with adaptive queries and monotone functions, one cannot learn better than an $\Omega(\sqrt{n}/\log n)$ approximation of a given fixed submodular function. Similarly, [1] addressed the submodular function learning problem from a learning theory perspective, given a distribution on subsets. They provide strong negative results including that one can not approximate in this setting to within a constant factor. In general, therefore, learning submodular functions is hard.

While learning over all possible submodular functions may be hard, this does not preclude learning submodular functions with known forms with unknown parameters. For example, given a finite set of M fixed submodular components $\{f_i\}_{i=1}^M$ where $f_i : 2^V \to \mathbb{R}$ is submodular over V, then learning a conical mixture $\sum_{i=1}^M w_i f_i$, where $(w_1, w_2, \ldots, w_M) = \boldsymbol{w} \in \mathbb{R}_+^M$, has not in the above been ruled out to have approximate guarantees. Such mixtures might span a very large set of submodular functions, depending on the diversity of the component set. We call such a problem "learning submodular mixtures."

In this paper we extend this one step further, to an approach we call learning "submodular shells." An instantiated submodular shell is a function $f_{\alpha,(V,\beta)} : 2^V \to \mathbb{R}$ indexed by a pair of parameter vectors α, β and a ground set V. The parameters β are associated with a particular ground set V but the parameters α of the shell apply to *any* ground-set vector pair (V,β). A *shell* has the form $f_{\bar{\alpha},(\cdot,\cdot)}$, which does not have (V,β) instantiated. We might learn, for example, that a particular value $\bar{\alpha}$ produces a good shell $f_{\bar{\alpha},(\cdot,\cdot)}$ for any instantiation of that shell with a particular (V,β). In this sense, a submodular shell might be seen as a form of "structured submodularity." Moreover, we might learn $\boldsymbol{w}$ in a mixture of such shells $\sum_i w_i f_{\alpha_i,(\cdot,\cdot)}$ based on data consisting only of training tuples of the form $\{((V^{(t)},\beta^{(t)}),S^{(t)})\}_t$ where $S^{(t)} \subseteq V^{(t)}$ and which can be used in an objective that involves the instantiated mixture of shells. Note that here, training tuples may consist of a sequence of different ground sets, and the goal is from these training tuples, and given a finite and fixed set of shells parameterized by $\{\alpha_i\}$, identify the conical weights $\boldsymbol{w}$ that optimize an objective. Of course, if all training ground sets are identical, then learning submodular shell mixtures reduces to learning submodular mixtures. In practice, however, the training ground sets are not usually identical. E.g., in extractive document summarization, the training data could be a set of documents and the corresponding human summaries, where the ground sets, those sentences that constitute the documents, are not identical between training samples. In particular, for a document t, the ground set $V^{(t)} = \{1,\cdots,n_t\}$ where n_t is the number of sentences in document t, and $\beta^{(t)}$ could be the term-frequency vectors for the sentences (see Section 5.2 for more instances of this).

We introduce a method whereby submodular shell mixtures may be learnt in a max-margin structured-prediction setting, and provide a risk-bound guarantee under approximate submodular maximization (Theorem 1). We apply our method to the extractive document summarization problem and, as mentioned in the abstract, this yields extremely good results.

2 Structured Prediction

A main goal of learning is to identify a function $h : \mathcal{X} \to \mathcal{Y}$, that maps from an input domain $\mathcal{X}$ to an output domain $\mathcal{Y}$. Structured prediction problems [51, 50, 14] are those where these domains may consist of combinatorial structures. Often, given an input $\boldsymbol{x} \in \mathcal{X}$, only a subset of $\mathcal{Y}$, say $\mathcal{Y}_{\boldsymbol{x}}$, is valid. For example, if $\boldsymbol{x}$ is a sentence, and $\mathcal{Y}$ is the set of all parse trees, than only a subset of possible parse trees $\mathcal{Y}_{\boldsymbol{x}}$ might be valid. This offers little solace, however, as $\mathcal{Y}_{\boldsymbol{x}}$ is typically still exponentially large. Usually, one forms a score function $s : \mathcal{X} \times \mathcal{Y}$ that measures how good an output $\boldsymbol{y}$ is for a given input $\boldsymbol{x}$. The decision problem is, given an $\boldsymbol{x}$, find one of the outputs with the highest score, i.e., $\boldsymbol{y}^* \in \operatorname{argmax}_{\boldsymbol{y}\in\mathcal{Y}_{\boldsymbol{x}}} s(\boldsymbol{x},\boldsymbol{y})$.

One of the primary challenges of structured prediction is that one needs to handle $\mathcal{Y}_{\boldsymbol{x}}$, whose cardinality is finite but exponentially large. A common way to address this issue is to make assumptions on both $\mathcal{Y}$ and s. For instance, if s decomposes over the parts of $\mathcal{Y}$ and moreover only depends on "local" parts, one could employ dynamic programming or integer programming algorithm to find the optimal solution. Another approach is to explore combinatorial structures upon which efficient algorithms are available. Examples of this sort include the Hungarian method [21] for maximum weighted bipartite matching and the Chu-Liu-Edmonds algorithm [6, 9] for optimal branchings. Alternatively, as we do in the sequel, one could resort to approximate solutions with approximation guarantees.

3 Submodular Shell Scores

In this paper, we propose to explore the structured prediction problem using submodular shell score functions. That is, given $\boldsymbol{x}$, $s(\boldsymbol{x},\cdot) : \mathcal{Y}_{\boldsymbol{x}} \to \mathbb{R}$ is a submodular function where $\mathcal{Y}_{\boldsymbol{x}}$ is considered to be the finite ground set associated with $\boldsymbol{x}$ — there is a finite ground set $V_{\boldsymbol{x}}$ associated with $\boldsymbol{x}$ where $\mathcal{Y}_{\boldsymbol{x}} \subseteq 2^{V_{\boldsymbol{x}}}$. There are at least two benefits of using such submodular score functions. First, submodular functions are natural and expressive with the capability of modeling decisions beyond local or linear interactions among parts. That is, submodular functions may allow for global direct interactions over a structured object unlike score functions that must decompose in some way. Second, this expressive power does require prohibitive computation as would an arbitrary score function. The reason is that the submodular maximization problem, even under many constrained settings, can be solved efficiently and near-optimally with rigorous performance guarantees [36, 13, 25, 2, 11].

The hypothesis function we consider takes the following linear discriminant form:

$$\begin{aligned} h(\boldsymbol{x};\boldsymbol{w}) &= \operatorname*{argmax}_{\boldsymbol{y}\in\mathcal{Y}_{\boldsymbol{x}}} s(\boldsymbol{x},\boldsymbol{y}) = \operatorname*{argmax}_{Y\in\mathcal{Y}_{\boldsymbol{x}}} \boldsymbol{w}^\top \boldsymbol{f}_{\boldsymbol{x}}(Y) \\ &= \operatorname*{argmax}_{Y\in\mathcal{Y}_{\boldsymbol{x}}} \sum_i w_i f_{\alpha_i,(V_{\boldsymbol{x}},\beta_{\boldsymbol{x}})}(Y) \end{aligned}$$

where $\boldsymbol{w} \in \mathbb{R}_+$ is a non-negative weight vector, and $f_{\alpha,(V,\beta)} : 2^V \to \mathbb{R}$ is submodular over subsets of V for every valid value of α and (V,β). Since the weights are non-negative, the score function is also submodular. We call $f_{\alpha,(V,\beta)}$ a *submodular shell*, abstracting a set of submodular functions characterized by some α,β and a ground set V. The score function is called a *submodular shell mixture*, and each $f_{\alpha_i,(V,\beta)}$ is a *shell component* of the mixture. Note that the parameters β are associated with a particular ground set V but the parameters α of the shell apply to *any* pair of ground set V and β value.

While we discuss our learning process in Section 4, we introduce the learning problem here to improve clarity. 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 $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ restricted in such a way that $\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}$ for any pair $(\boldsymbol{x}, \boldsymbol{y})$. $\boldsymbol{x}^{(t)}$ corresponds to a finite ground set $V_{\boldsymbol{x}^{(t)}}$ and a set of parameters $\beta_{\boldsymbol{x}^{(t)}}$ and so we say that $\boldsymbol{x}^{(t)} = (V_{\boldsymbol{x}^{(t)}}, \beta_{\boldsymbol{x}^{(t)}})$. We also have a finite collection of M submodular shells $\{f_{\alpha_i,(\cdot,\cdot)}\}_{i=1}^{M}$ each of which is instantiated into a submodular function for any $\boldsymbol{x} \in \mathcal{X}$. That is, $f_{\alpha_i,\boldsymbol{x}^{(t)}} : 2^{V_{\boldsymbol{x}^{(t)}}} \to \mathbb{R}$ is a submodular function. We are also given a loss function and $\ell_{\boldsymbol{x},\boldsymbol{y}}(\hat{\boldsymbol{y}})$ that measures the loss of predicting $\hat{\boldsymbol{y}} \in \mathcal{Y}_{\boldsymbol{x}}$ when the true label is $\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}$. The empirical risk minimization problem then is to find a conical mixture $\boldsymbol{w} \in \mathbb{R}_{+}^{M}$ such that the risk $\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{x},\boldsymbol{y}}(h(\boldsymbol{x};\boldsymbol{w}))]$ is minimized. It is important to realize that what is learnt is the mixture coefficients $\boldsymbol{w}$ over a set of shells that may be instantiated into a weighted sum of submodular functions.

Before we discuss learning in detail, we consider the expressive power of mixtures of such submodular shells.

3.1 Submodular Shell Mixtures

A fairly rich subclass of submodular functions can be instantiated from submodular shell mixtures with components being simpler submodular shells. Consider, for example, truncation functions, mixtures of which can represent many submodular functions of interest. Given a modular function[1] $c : 2^V \to \mathbb{R}$, define a truncation function $t_{\alpha,(V,c)} : 2^V \to \mathbb{R}$ as

$$t_{\alpha,(V,c)}(A) = \min\left\{c(A), \alpha \max_{B \subseteq V} c(B)\right\},$$

which is submodular, where $\alpha \in \mathbb{R}$ is the truncation threshold. A modular function can also be written as a truncation function with $\alpha = \infty$. We note that (V, β) above takes the form (V, c) here, and we might have a collection of such functions $\{t_{\alpha_i,(V,c)}\}_i$, and learning a mixture $\boldsymbol{w}$ would mean that w_i indicates the importance of truncation threshold α_i.

In fact, many submodular functions can be written as mixtures of truncation functions. For example, coverage type functions, canonical examples of submodular functions, can be written as submodular mixtures. Precisely, let a collection of sets be $\{A_i\}_{i\in\{1,\cdots,|V|\}}$ on a ground set E. Then the set cover function becomes

$$f : 2^V \to \mathbb{R}_+, \quad f(B) = \left|\bigcup_{i\in B} A_i\right|.$$

We can define costs $c_{i,j} = 1$ if A_i contains $j \in E$ and $c_{i,j} = 0$ otherwise. We then have

$$f(B) = \left|\bigcup_{i\in B} A_i\right| = \sum_{j=1}^{|E|} \min\left\{\sum_{i\in B} c_{i,j}, 1\right\},$$

which is a mixture of truncation functions. Moreover sums of concave functions applied to cardinality functions [47] can be represented as mixtures of truncation functions — any sum of such functions can be expressed as a sum of a modular function and nonnegative linear combinations of truncation functions.

Another interesting class is weighted matroid rank functions [32]. Given matroids [56, 38] $\mathcal{M}_i = (V, \mathcal{I}_i)$, and modular functions $m_i : 2^V \to \mathbb{R}_+$, we get:

$$\sum_{i=1}^{M} w_i f^i_{m_i,(V,\mathcal{M}_i)} = \sum_{i=1}^{M} w_i \max\{m_i(I) : I \subseteq S, I \in \mathcal{I}_i\}$$

which includes cover-like functions and many others.

The submodular functions introduced in [28] for document summarization can also been seen as a mixture of submodular shells, with components being either the coverage or the diversity shell that can be instantiated for a particular document.

In general, by using rich enough families of shell components, a submodular shell mixture could be very expressive, representing a very large family of submodular functions. Moreover, another advantage of using submodular shell mixture representations is that, since we assume each component is given and only the component weights are unknown, the learning problem can be addressed by using well-established methods. And by learning shell mixtures, we can then apply the learnt mixture to structured problems even over different underlying ground sets.

4 Learning Submodular Shell Mixtures

While there might be many ways of learning shell mixtures, in this paper we take a large-margin approach, standard in structured prediction. In other words, we want to minimize the risk of making predictions (decisions) when using the submodular shell mixture as a score function. Of course, the standard maximization in learning structured prediction is only approximate in this case, since maximizing submodular functions in NP-hard (although constant-factor approximable). Moreover, we need to identify a valid loss function that dose not violate submodularity. We wish also to ensure that the learnt parameters have quality guarantees (e.g., a risk bound). All of this is done in the below.

[1] f is modular if both f and $-f$ are submodular.

4.1 Large Margin Learning

We consider the learning problem defined in Section 3. For concision, we denote

$$\mathcal{Y}_t \triangleq \mathcal{Y}_{\boldsymbol{x}^{(t)}}, \quad \boldsymbol{f}_t(\boldsymbol{y}) \triangleq \boldsymbol{f}_{\boldsymbol{x}^{(t)}}(\boldsymbol{y}), \text{ and } \ell_t(\boldsymbol{y}) \triangleq \ell_{\boldsymbol{x}^{(t)},\boldsymbol{y}^{(t)}}(\boldsymbol{y}).$$

We follow the maximum margin approach [51] to learn the component weights $\boldsymbol{w}$, where 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 is as follows:

$$\min_{\boldsymbol{w} \geq 0} \frac{1}{T} \sum_{t=1}^{T} \hat{\ell}_t(\boldsymbol{w}) + \frac{\lambda}{2} \|\boldsymbol{w}\|^2, \tag{1}$$

where the generalized hinge loss

$$\hat{\ell}_t(\boldsymbol{w}) \triangleq \max_{\boldsymbol{y} \in \mathcal{Y}_t} \left(\boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y}) \right) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) \tag{2}$$

and a quadratic regularizer is minimized.

Algorithm 1: Projected subgradient descent for learning submodular shell mixtures.

```
Input  : $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^T$ and a learning rate
         sequence $\{\eta_t\}_{t=1}^T$.
$\boldsymbol{w}_0 = 0$;
for $t = 1, \cdots, T$ do
    Approximate loss augmented inference:
    $\hat{\boldsymbol{y}}_t \approx \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{Y}_t} \boldsymbol{w}_{t-1}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y})$;
    Compute the subgradient:
    $\boldsymbol{g}_t = \lambda \boldsymbol{w}_{t-1} + \boldsymbol{f}_t(\hat{\boldsymbol{y}}_t) - \boldsymbol{f}_t(\boldsymbol{y}^{(t)})$;
    Update the weights with projection:
    $\boldsymbol{w}_t = \max(\boldsymbol{0}, \boldsymbol{w}_{t-1} - \eta_t \boldsymbol{g}_t)$;
Return : the averaged parameters $\frac{1}{T} \sum_t \boldsymbol{w}_t$.
```

Many algorithms have been proposed for the large margin learning problem, including those based on the exponentiated gradient method [7], the dual extragadient method [52], the cutting-plane algorithm [54], and the subgradient descent method [42, 43]. We adopt a subgradient descent algorithm to learn submodular shell mixtures, as illustrated in Algorithm 1, where 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 $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$.

Note, to preserve submodularity, the component weights must be non-negative. We thus simply project the weights to the non-negative orthant whenever doing the updates, and 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 as when a point is projected back into the convex set, it is moved closer to every point in the set including the optimal points.

Second, whether the learning can be done efficiently depends on whether the so called *loss augmented inference* (LAI) problem,

$$\max_{\boldsymbol{y} \in \mathcal{Y}_t} \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}) + \ell_t(\boldsymbol{y}), \tag{3}$$

can be solved efficiently. The LAI problem has a term that precisely matches the prediction problem whose parameters we are trying to learn but also has an additional term corresponding to the loss. Tractability of LAI therefore not only depends on the tractability of the prediction problem but also on the form of loss functions. When using submodular shell mixtures as the score function, we use approximate inference with a performance guarantee. Therefore, we must perform approximate inference for the LAI problem as well. We thus in general refer to this as approximate learning.

4.2 Approximate learning

Since some form of inference is a dominant subroutine in many learning algorithms for structured prediction, it is natural to use good approximate inference techniques to make the learning problem tractable. When learning submodular shell mixtures, we inevitably must use approximate learning since, by using a more expressive class of score functions that need not decompose, the inference problem is intractable to do exactly. Using approximate inference as a drop-in replacement for exact inference in learning, however, could mislead the learning algorithm and result in poorly learnt models. This is analyzed in [24], where it is pointed out that approximate learning could fail even with an approximate inference method having approximation guarantees. In general, therefore, it is problematic to assume that an arbitrary choice of approximate inference method will lead to useful results when the learning method expects exact feedback. Choosing compatible inference and learning procedures is therefore crucial.

In the following, we aim to leverage the approximation guarantees of submodular optimization such that the performance of approximate learning of submodular shell 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 utility if good prediction cannot be made from them. Alternatively, we can 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.

4.3 Analysis

Hence, we must further analyze whether using good approximate inference will lead us to good approximate learning in our case. 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 $f(\boldsymbol{y}) \geq f^*$. A greedy algorithm or the loopy belief propagation [40] are instances of undergenerating approximation algorithms. Relaxation methods, e.g. linear programming relaxation, are overgenerating approximation algorithms. Note that undergenerating algorithms usually produce solutions that are within the feasible region of the problem. Overgenerating algorithms, on the other hand, generate solutions that might lie outside this feasible region. Therefore, solutions found by overgenerating algorithms sometimes need to be mapped back to the feasible region (e.g., rounding of linear programming produced solutions) in order to produce a feasible solution, during which the approximation guarantee no longer holds in some cases [44]. For learning submodular shell mixtures, we are particularly interested in undergenerating algorithms since the greedy algorithm, one of the undergenerating algorithms, offers near-optimal solutions for submodular maximization under certain (e.g., cardinality, budget, and matroid) constraints [36, 13].

Generalization bounds for approximate learning with cutting-plane algorithms, with either undergenerating or overgenerating inferences, have been shown in [12]. For subgradient descent methods, generalization analysis is available, but only for overgenerating cases, in [30, 22]. As far as we know, no generalization analyses are available for approximate learning with undergenerating subgradient methods. We fill this gap and offer risk bounds for approximate learning with undergenerating subgradient methods.

We need two definitions before moving on.

Definition 1 (ρ-approximate algorithm). *Given $f : \mathcal{Y} \to \mathbb{R}^+$ 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 $0 \leq \rho \leq 1$.*

Definition 2 (γ-approximate subgradient). *Given $f : \mathbb{R}^M \to \mathbb{R}$, a vector $\boldsymbol{g} \in \mathbb{R}^M$ 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 $0 \leq \gamma \leq 1$ and $\boldsymbol{w}, \boldsymbol{w}' \in \mathbb{R}^M$.*

In Algorithm 1, one needs to solve the LAI in Eqn (3). Note that if ℓ is modular (e.g. hamming loss) or submodular (e.g. see Section 5.3), a ρ-approximate inference algorithm can also apply to this loss augmented inference to find a near-optimal solution efficiently. However, as mentioned above, 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 [24]. On the other hand, Ratliff et al. [43] show that the subgradient algorithm is robust under approximate settings, and the risk experienced during training with γ-approximate subgradients can be bounded.

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 analyze the actual impact of ρ-approximate LAI in the learning procedure when compared with the exact formulation, then, we provide risk bounds for the approximate learner in Theorem 1.

Theorem 1. *Assume $w_i, f_i, i = 1, \cdots, M$ are all upper-bounded by 1, $\hat{\ell}_t(\boldsymbol{w}) \leq B$, and $\|\boldsymbol{g}_t\| \leq G$. Let $\boldsymbol{w}^*$ and $\hat{\boldsymbol{w}}$ be the solutions returned by Algorithm 1 using exact and ρ-approximate LAI, respectively, 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 $1-\delta$,*

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell_{\boldsymbol{y}}(h(\boldsymbol{x};\hat{\boldsymbol{w}}))] \leq \frac{1}{\rho}\left(\frac{1}{T}\sum_{t=1}^{T}\hat{\ell}_t(\boldsymbol{w}^*)\right) + S(T),$$

where

$$S(T) = \frac{MG}{\rho}\sqrt{\frac{2(1+\log T)}{T}} + B\sqrt{\frac{2}{T}\log\frac{1}{\delta}} + \frac{1-\rho}{\rho}M.$$

The proof is given in the supplementary material. 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 $(R^* + (1-\rho)M)/\rho$, where $R^* = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\hat{\ell}_t(\boldsymbol{w}^*)$ and can be seen as the risk of predictions using models learnt with exact inference. Thus, the better (larger ρ) the approximation, the less the additional risk there will be when using an approximate LAI. And when exact inference is used ($\rho = 1$), the additional risk shrinks to zero. Note that Theorem 1 applies to any loss function and any score function which is not necessary to be submodular. When addition assumptions are made (e.g., assumption that the loss function is linearly realizable as in [57]), a better bound might be possible where the additional risk shrinks to zero as T grows. On the other hand, when the LAI objective is monotone submodular, a simple and efficient greedy algorithm performs near-optimally with approximation factor $1 - 1/e$ [36]. In practice, moreover, as the approximation factor of the greedy algorithm on submodular maximization is usually very close to 1 [27], one could expect very little additional risk when using Algorithm 1 with approximate inference on learning submodular shell mixtures.

To the best of our knowledge, Theorem 1 is the first approximate learning bound for subgradient algorithms with undergenerating (greedy) inference.

5 Application to Document Summarization

Submodular mixtures could be applied to many structured prediction problems of practical interest. In this paper, we apply submodular shell mixture learning to extractive document summarization as a case study.

5.1 Submodularity in Document Summarization

Extractive document summarization can be seen as a subset selection problem [27]. Given a ground set of sentences V, the task of extractive document summarization is selecting a subset of sentences, say S, that best represents the whole document. In other words, we want to find $A \subseteq V$ such that

$$A \in \operatorname*{argmax}_{B \subseteq V} f(B) \text{ subject to: } \sum_{i \in B} c_i \leq b, \qquad (4)$$

where $c_i \in \mathbb{R}^+$ is the cost of sentence i (e.g., it could be the number of words in the sentence), $b \in \mathbb{R}^+$ is the total budget (e.g., it could the largest number of words allowed in a summary), and $f : 2^V \to \mathbb{R}$ is a set function that models the quality of a summary. Eqn (4) is known as the problem of submodular maximization subject to knapsack constraints [31] which NP-complete [35]. However, when f is *monotone submodular*, Eqn (4) can be solved efficiently and near-optimally with a theoretical guarantee via greedy algorithms [48, 27].

One can always force f to be submodular, leading to an objective function that can be optimized well but might on the other hand poorly represent a given problem. One attractive property of submodularity, like convexity in continuous domain, is that it arises naturally in many applications. One such applications is document summarization. As pointed out in [28], many well-established methods, including the widely used maximum margin relevance method [3], actually correspond to submodular optimization. Moreover, it is shown that the commonly used ROUGE score [26] for automatic summarization evaluation is monotone submodular [28], giving further evidence that submodular functions are natural for document summarization.

In this paper, we further show that not only is the ROUGE score submodular, the score used in the Pyramid method [37], one of the manual evaluation metrics that has been used in recent TAC summarization track[2], is also monotone submodular.

Theorem 2. *The modified score in Pyramid method is monotone submodular.*

The proof is in Appendix C in the supplement.

The remaining question is how to design (or ideally learn) a good submodular function for summarization. Lin and Bilmes [28] proposed a class of submodular functions that models the coverage as well as the diversity of summary. In this paper, we further generalize their class of submodular functions and propose to use *submodular shell mixtures* for document summarization.

5.2 Submodular shells for summarization

Diversity shell components

We define a diversity shell component as

$$f^{\text{diversity}}_{(a,K,\mathcal{A}),(V,\boldsymbol{r})}(S) = \frac{\sum_{k=1}^{K} \left(\sum_{i \in S \cap P_k} r_i\right)^a}{\sum_{k=1}^{K} \left(\sum_{i \in P_k} r_i\right)^a}, \qquad (5)$$

where $0 \leq a \leq 1$ is the curvature, $K \in \mathbb{Z}^+$ is the number of clusters (partitions), $\mathcal{A}$ is a clustering algorithm, and $\{P_k\}_{k=1,\cdots,K}$ is a partition of the ground set V generated by $\mathcal{A}$, and $\boldsymbol{r} = \{r_i\}_{i=1}^{|V|}$ with $r_i \in [0, 1]$ is the vector of singleton reward of element $i \in V$. The diversity component models the diversity of a summary set S, by diminishing the benefit of choosing elements from the same cluster.

Note that the α parameter of a submodular shell here takes the form $(a, K, \mathcal{A})$. By using different values of a and K, and different clustering algorithms $\mathcal{A}$, we can produce a variety of submodular shells. The (V, β) parameter of a submodular shell takes the form of $(V, \boldsymbol{r})$. When a document (ground set) is given, rewards of each sentence (i.e., r_i) can be computed, and the diversity shell component is then instantiated into a submodular function that measures the diversity of a summary for this particular document.

Clustered facility location shell components

We define clustered facility-location like components as

$$f^{\text{c-facility}}_{(K,\mathcal{A}),(V,\boldsymbol{r})}(S) = \frac{1}{K} \sum_{k=1}^{K} \max_{i \in S \cap P_k} r_i, \qquad (6)$$

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

where $K \in \mathbb{Z}^+$ is the number of clusters (partitions), $\mathcal{A}$ is a clustering algorithm, and $r_i \in [0,1]$ is the singleton reward of element $i \in V$. This function has a similar form to the well known submodular facility location function, but defined on a partition of the ground set. We thus call it clustered facility location. If a summary contains multiple elements from the 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.

Fidelity shell components

Given a ground set V, we define fidelity components as

$$f^{\text{fidelity}}_{\alpha,\left(V,\{C_i\}_{i=1}^{|V|}\right)}(S) = \frac{1}{|V|}\sum_{i\in V}\min\left\{\frac{\mathcal{C}_i(S)}{\mathcal{C}_i(V)}, \alpha\right\}, \qquad (7)$$

where $0 < \alpha \leq 1$ is a saturation 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 [28]. Basically, the saturation threshold controls how much of a given element $i \in V$ should be covered; once $\mathcal{C}_i(S)$ is large enough such that the ratio of it over its largest possible value ($\mathcal{C}_i(V)$) is above threshold, covering more of i does not further increase the function value. Therefore, a larger value of f^{fidelity} tends to have more $i \in V$ well covered. When a document is given, we can instantiate different submodular shells using a variety of C_i.

5.3 A Submodular Loss Function

The most widely used evaluation criteria for summarization is the 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 [26] 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 [28], but cannot be used as a loss function since it basically measures "accuracy" rather than loss.

An alternative is to use $1 - f_{\text{ROUGE-N}}(S)$ as a loss function, but this is 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 loss augmented inference. When using $1 - f_{\text{ROUGE-N}}$, which is supermodular as a loss function, in the objective function for loss augmented inference (Eqn. (3)) along with a submodular shell mixture as the score function, the resulting objective function for LAI is then a submodular function plus a supermodular function. While an algorithm (e.g., submodular-supermodular procedure [33, 17]) is available to approximately optimize the sum of a submodular function and a supermodular function, performance guarantees usually do not exist for these algorithms (although this strategy might work well in practice and should ultimately be tested). Therefore, when using one-minus-ROUGE as the loss function, the greedy algorithm no longer provides a near-optimal solution when applied to the non-submodular objective, and the risk bound shown in Theorem 1 no longer holds.

To address this issue, we propose a ROUGE-like loss function that measures the "complement recall":

$$\ell_{\text{ROUGE}}(S) \triangleq \frac{\sum_{e\in\bar{R}}\omega_e c_e(S)}{\sum_{e\in\bar{R}}\omega_e r_e}, \qquad (8)$$

where $\bar{R} = N \backslash \bigcup_i R_i$, and N is the set of all the n-grams occur in the set of documents, and $r_e = c_e(V)$ is the number of times n-gram e occurs in all the documents, ω_e is a non-negative weight for e, and $\bar{R}$ is the set of n-grams that are *not* covered by any human reference summary. Instead of counting with respect to a reference summary, ℓ_{ROUGE} counts the n-grams of a candidate summary S w.r.t. the *complement* of reference summaries.

Intuitively, we want a summary S to cover as many reference n-grams as possible so that it will get a high ROUGE-score; this is similar to having S be large and overlapping 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 n-grams set are covered, and when comparing summaries with the same size, the smaller ℓ_{ROUGE} is, the better. The best case, i.e., the human reference itself, will have ℓ_{ROUGE} equal to 0.

Obviously, a poor summary that would also have ℓ_{ROUGE} equal to 0 is an empty 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 try to consume every bit of the budget in order to consume 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, the larger the number of reference n-gram overlaps there are, and therefore the better the summary. We use the greedy algorithm for solving the summarization problems (Eqn. (4)). Due to its greedy nature, the algorithm always outputs summary candidates whose costs are close to the budget, and therefore ℓ_{ROUGE} can serve as a reasonable surrogate loss function for learning submodular shell mixtures for summarization tasks.

The major advantage of using ℓ_{ROUGE} as a loss function, of course, is algorithmic. The submodular learning analysis in Theorem 1 relies on the fact that a ρ-approximation algorithm is available for the loss augmented information. Since we use a 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 (in fact modular). Therefore, the LAI in submodular shell mixture learning for summarization is exactly the budgeted submodular maximization problem (Eqn (4)), and efficient and near-optimal algorithms are then available. Consequently, all the theoretical analyses in Section 4.3 apply.

We will soon see (Section 7) that using the ROUGE-like loss function proposed here empirically outperforms using $1 - f_{\text{ROUGE}}$ as a loss functions.

6 Related work

Recently, Yue and Guestrin [57] studied a linear submodular bandits problem in an online learning setting for optimizing a general class of feature-rich submodular utility models in diversified retrieval, and their theoretical result is based on the assumption that the award function has a special (linear) form. Raman et al. [41] also propose an online learning model and algorithm for learning rankings that exploiting feedback to maximize any submodular utility measure. While our approach and analysis could also be applied to the online setting, the focus and analysis in this paper are more on the batch learning. The submodular utility models in [57, 41], however, can both be seen as special instances of submodular shell mixtures.

More closely related to our work is that of [46] where large-margin learning of submodular score functions for extractive document summarization is studied. The representation of a submodular score function in [46], again, turns out to be a special case of a submodular shell mixture. Moreover, our submodular shell mixture framework is more general and flexible than the framework proposed in [46]. 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 [46] do not apply. Viewing each feature as input to a submodular component, on the other hand, preserves the linearity on the parameters, whereas Algorithm 1 applies. Moreover, representing a score function using a submodular shell mixture is expressive, as we have shown in Section 3.1.

In [46], 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 the greedy algorithm introduced in [27] 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 [12]. Also, Sipos et al. [46] apparently neglect the fact that the learned similarity (or importance) should be non-negative, which is a necessary ingredient 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 shell mixtures by project the weights to the non-negative orthant (Algorithm 1).

7 Experiments

We evaluated our approach on NIST's DUC[3] data 2003-2007, and demonstrate results on both generic and query-focused summarization. DUC data were created by national institution of standard technology (NIST). The DUC evaluation is one of the standardized benchmark evaluations for document summarization and researchers continue to publish results on these data sets.

7.1 Query-independent summarization

Table 1: ROUGE-1 recall (R) and F-measure (F) results (%) on DUC-04. DUC-03 was used as development set.

DUC-04	R	F
Takamura and Okumura [49]	38.50	-
Wang et al. [55]	39.07	-
Lin and Bilmes [27]	-	38.39
Lin and Bilmes [28]	39.35	38.90
Kulesza and Taskar [23]	38.71	38.27
Best system in DUC-04 (peer 65)	38.28	37.94
Submodular Shell Mixture	**40.43**	**39.78**

The summarization tasks in DUC-03 and DUC-04 are

[3]http://duc.nist.gov/

generic summarization tasks. We used DUC-03 as the training set for submodular shell 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 shell 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. The third similarity is again cosine similarity but on the vector generated by latent semantic analysis. For each of the similarity measures, we use five different saturation thresholds ($\alpha = 0.01, \cdots, 0.05$), and thus we have 15 fidelity components in total. We used Algorithm 1 with the ROUGE-like loss (Eqn. (8)) with $\omega_e = 1$ for all e to learn this submodular shell mixture. The ROUGE-1 results are shown in Table 1. As we can see, the result of the learned submodular shell mixture significantly outperforms all other previous reported results. Note that the experiment in [46] used a non-standard setup where DUC-04 data were divided into training, development, and test sets with only 5 documents in their test set. Therefore, results reported in [46] are not directly comparable to other results (e.g., our results) that follow standard DUC evaluation setup.

7.2 Query-focused summarization

Table 2: 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 and Marcu [8]	6.98	-
Lin and Bilmes [28]	7.82	7.72
Best system in DUC-05 (peer 15)	7.44	7.43
Submodular Shell Mixture	**8.44**	**8.39**

Table 3: 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 and Hakkani-tür [4]	9.10	-
Shen and Li [45]	9.30	-
Lin and Bilmes [28]	9.75	9.77
Best system in DUC-06 (peer 24)	9.51	9.51
Submodular Shell Mixture	**9.92**	**9.93**

Since 2004, DUC summarization evaluations have concentrated on query-focused summarization. We tested our approach for summarization on DUC-05, DUC-06 and DUC-07 data. In particular, we used DUC-06,07 as the training set for the DUC-05 task, DUC-05,07 as the training set for the DUC-06 task, and DUC-05,06 as the training set for the DUC-07 task.

Table 4: ROUGE-2 recall (R) and F-measure (F) results (%) on DUC-07. DUC-05 and DUC-06 were used as training sets. Note that the peer 15 system in DUC-07 used commercial web search engine to expand the queries.

DUC-07	R	F
Toutanova et al. [53]	11.89	11.89
Haghighi and Vanderwende [16]	11.80	-
Celikyilmaz and Hakkani-tür [4]	11.40	-
Lin and Bilmes [28]	12.38	12.33
Best system in DUC-07 (peer 15)	12.45	12.29
Submodular Shell Mixture	**12.51**	**12.40**

We used diversity components, clustered facility location components, and fidelity components to form the submodular shell mixture for query-focused summarization. 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 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 total. Compared to other results reported in the literature (Table 2, Table 3 and Table 4), as far as we know, our approach achieves the best results reported so far on DUC-05, DUC-06, and DUC-07.

8 Conclusions

In this paper, we propose the notion of learning submodular shells, which abstract a set of submodular functions that can be instantiated into submodular functions given the input of a structured prediction task. Given a set of training instances, we use subgradient descent to learn the mixture coefficients over a set of submodular shells that may be instantiated into a weighted sum of submodular functions. We show that the submodular shell mixture is very expressive, and the risk of learning can be bounded when only approximate inference in possible. When applied to the task of document summarization, our approach achieves the best results reported so far on standardized benchmark takes for query-focused extractive summarization tasks. This work was partially supported by a Google research award, and by NSF award IIS-1162606.

References

[1] M.F. Balcan and N.J.A. Harvey. Learning submodular functions. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 793–802. ACM, 2011.

[2] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *Proc. of 12th IPCO*, pages 182–196. Citeseer, 2007.

[3] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. of SIGIR*, 1998.

[4] A. Celikyilmaz and D. Hakkani-tür. A hybrid hierarchical model for multi-document summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 815–824, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[5] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. In *NIPS*, pages 359–366, 2001.

[6] Y.J. Chu and T.H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14(1396-1400):270, 1965.

[7] P.L.B.M. Collins and B.T.D. McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in neural information processing systems*, volume 17, page 113, 2004.

[8] H. Daumé III and D. Marcu. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, page 312, 2006.

[9] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 1967.

[10] J. Edmonds. *Combinatorial Structures and their Applications*, chapter Submodular functions, matroids and certain polyhedra, pages 69–87. Gordon and Breach, 1970.

[11] U. Feige, V. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. In *Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.

[12] T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *Proceedings of the 25th international conference on Machine learning*, pages 304–311. ACM, 2008.

[13] ML Fisher, GL Nemhauser, and LA Wolsey. An analysis of approximations for maximizing submodular set functions—II. *Polyhedral combinatorics*, pages 73–87, 1978.

[14] L. Getoor and B. Taskar. *Introduction to statistical relational learning*. The MIT Press, 2007.

[15] M.X. Goemans, N.J.A. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 535–544. Society for Industrial and Applied Mathematics, 2009.

[16] A. Haghighi and L. Vanderwende. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[17] Rishabh Iyer and Jeff Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. In *uai*, Catalina Island, USA, July 2012. AUAI.

[18] D. Kempe, J. Kleinberg, and E. Tardos. 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)*, 2003.

[19] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proc. of Uncertainty in AI*, 2005.

[20] A. Krause, H.B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *Journal of Machine Learning Research*, 9: 2761–2801, 2008.

[21] H.W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[22] A. Kulesza. Approximate learning for structured prediction problems. *UPenn WPE-II Report*, 2009.

[23] A. Kulesza and B. Taskar. Learning determinantal point processes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

[24] A. Kulesza, F. Pereira, et al. Structured learning with approximate inference. *Advances in neural information processing systems*, 20:785–792, 2007.

[25] J. Lee, M. Sviridenko, and J. Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Approximation, Ran-*

domization, and Combinatorial Optimization. Algorithms and Techniques, pages 244–257, 2009.

[26] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 2004.

[27] H. Lin and J. Bilmes. 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)*, Los Angeles, CA, June 2010.

[28] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, Portland, OR, June 2011.

[29] L. Lovász. Submodular functions and convexity. *Mathematical programming-The state of the art,(eds. A. Bachem, M. Grotschel and B. Korte) Springer*, pages 235–257, 1983.

[30] A.F.T. Martins, N.A. Smith, and E.P. Xing. 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, 2009.

[31] R. McDonald. A study of global inference algorithms in multi-document summarization. *Lecture Notes in Computer Science*, 4425:557, 2007.

[32] K. Murota. Recent developments in discrete convex analysis. *Research Trends in Combinatorial Optimization*, pages 219–260, 2009.

[33] M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Proc. Conf. Uncertainty in Artifical Intelligence, Edinburgh, Scotland, July. Morgan Kaufmann Publishers*, 2005.

[34] Mukund Narasimhan and Jeff Bilmes. PAC-learning bounded tree-width graphical models. In *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI-2004)*. Morgan Kaufmann Publishers, July 2004.

[35] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988.

[36] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, 14(1):265–294, 1978.

[37] A. Nenkova and R. Passonneau. Evaluating content selection in summarization: The pyramid method. In *Proceedings of HLT-NAACL*, volume 2004, pages 145–152, 2004.

[38] J. Oxley. *Matroid theory*, volume 21. Oxford Univ Pr, 2011.

[39] R.J. Passonneau, A. Nenkova, K. McKeown, and S. Sigelman. Applying the pyramid method in duc 2005. In *Proceedings of the Document Understanding Conference (DUC 05), Vancouver, BC, Canada*, 2005.

[40] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[41] K. Raman, P. Shivaswamy, and T. Joachims. Learning to diversify from implicit feedback. In *WSDM Workshop on Diversity in Document Retrieval*, 2012.

[42] N. Ratliff, J.A. Bagnell, and M. Zinkevich. Subgradient methods for maximum margin structured learning. In *ICML Workshop on Learning in Structured Output Spaces*, 2006.

[43] N.D. Ratliff, J.A. Bagnell, and M.A. Zinkevich. (online) subgradient methods for structured prediction. In *AISTATS*, volume 2007. Citeseer, 2006.

[44] P. Ravikumar, A. Agarwal, and M.J. Wainwright. Message-passing for graph-structured linear programs: proximal projections, convergence and rounding schemes. In *Proceedings of the 25th international conference on Machine learning*, pages 800–807. ACM New York, NY, USA, 2008.

[45] C. Shen and T. Li. Multi-document summarization via the minimum dominating set. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 984–992, Beijing, China, August 2010. Coling 2010 Organizing Committee.

[46] R. Sipos, P. Shivaswamy, and T. Joachims. Large-margin learning of submodular summarization methods. In *Proceedings of the 13th conference of the European chapter of the Association for Computational Linguistics*, 2012.

[47] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *NIPS*, 2010.

[48] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[49] H. Takamura and M. Okumura. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781–789. Association for Computational Linguistics, 2009.

[50] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. *Advances in neural information processing systems*, 16:25–32, 2004.

[51] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM, 2005.

[52] B. Taskar, S. Lacoste-Julien, and M.I. Jordan. Structured prediction, dual extragradient and bregman projections. *The Journal of Machine Learning Research*, 7:1627–1653, 2006.

[53] K. Toutanova, C. Brockett, M. Gamon, J. Jagarlamudi, H. Suzuki, and L. Vanderwende. The PYTHY summarization system: Microsoft research at DUC 2007. In *the proceedings of Document Understanding Conference*, 2007.

[54] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453, 2006.

[55] D. Wang, S. Zhu, T. Li, and Y. Gong. Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 297–300, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[56] D.J.A. Welsh. *Matroid theory*, volume 5. Academic Press London, New York, San Francisco, 1976.

[57] Yisong Yue and Carlos Guestrin. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems 24*, pages 2483–2491, 2011.

A γ-approximate gradient and ρ-approximate inference

Ratliff et al. [43] 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 1.

Lemma 1 (43)**.** *Assume $\hat{\ell}(\boldsymbol{w})$ is upper-bounded by 1. Let $\hat{\boldsymbol{w}}_t$ be the solution produced at each iteration of Algorithm 1 with a γ-approximate subgradient, and $\boldsymbol{w}^*$ be the solution generated by Algorithm 1 with an exact subgradient. For $\lambda = \frac{\sqrt{1+\log T}}{\|\boldsymbol{w}^*\|\sqrt{T}}$, $\eta_t = \frac{1}{\lambda t}$, and $S(T) = \|\boldsymbol{w}^*\|\sqrt{T(1+\log T)}$,*

$$\frac{1}{T}\sum_{t=1}^{T}\ell_t(h(\boldsymbol{x}^{(t)};\hat{\boldsymbol{w}}_t)) \le \frac{1}{1-\gamma}\left(\frac{1}{T}\sum_{t=1}^{T}\hat{\ell}_t(\boldsymbol{w}^*) + \frac{S(T)}{T}\right). \tag{9}$$

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 this, let $\hat{\boldsymbol{y}}_t$ be the solution found by ρ-approximate LAI, i.e.

$$\boldsymbol{w}^\top \boldsymbol{f}_t(\hat{\boldsymbol{y}}_t) + \ell_t(\hat{\boldsymbol{y}}_t) \ge \rho\left(\boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}_t^*) + \ell_t(\boldsymbol{y}_t^*)\right) \tag{10}$$

where

$$\boldsymbol{y}^* \in \underset{\boldsymbol{y}\in\mathcal{Y}_t}{\operatorname{argmax}}\, \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}_t) + \ell_t(\boldsymbol{y}_t), \tag{11}$$

and let

$$c_t(\boldsymbol{w}) \triangleq \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^*) + \ell_t(\boldsymbol{y}^*) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \tag{12}$$

$$\hat{c}_t(\boldsymbol{w}) \triangleq \boldsymbol{w}^\top \boldsymbol{f}_t(\hat{\boldsymbol{y}}_t) + \ell_t(\hat{\boldsymbol{y}}_t) - \boldsymbol{w}^\top \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \tag{13}$$

We compute a subgradient of $\hat{c}_t(\boldsymbol{w})$ at $\boldsymbol{w}$:

$$\hat{\boldsymbol{g}}_t = \nabla_{\boldsymbol{w}}\hat{c}_t(\boldsymbol{w}) = \boldsymbol{f}_t(\hat{\boldsymbol{y}}_t) - \boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \lambda\boldsymbol{w}, \tag{14}$$

and the goal is to check whether $\hat{\boldsymbol{g}}_t$ is a $\phi(\rho)$-approximate subgradient of $c_t(\boldsymbol{w})$ at $\boldsymbol{w}$, where $\phi : [0,1] \to [0,1]$.

We have for any $\boldsymbol{w}'$,

$$\begin{aligned}
\hat{\boldsymbol{g}}_t^\top(\boldsymbol{w}' - \boldsymbol{w}) &\le \hat{c}_t(\boldsymbol{w}') - \hat{c}_t(\boldsymbol{w}) && (15)\\
&\le c_t(\boldsymbol{w}') - \rho c_t(\boldsymbol{w}) + (1-\rho)\boldsymbol{w}^\top f_t(\boldsymbol{y}^{(t)}) - (1-\rho)\frac{\lambda}{2}\|\boldsymbol{w}\|^2 && (16)\\
&= c_t(\boldsymbol{w}') - c_t(\boldsymbol{w}) + (1-\rho)c_t(\boldsymbol{w}) + (1-\rho)\left(\boldsymbol{w}^\top f_t(\boldsymbol{y}^{(t)}) - \frac{\lambda}{2}\|\boldsymbol{w}\|^2\right) && (17)\\
& && (18)
\end{aligned}$$

where the first inequality due to the fact that $\hat{\boldsymbol{g}}$ is a subgradient of $\hat{c}_t(w)$ at $\boldsymbol{w}$, and the second inequality is because of

$$\begin{aligned}
\hat{c}_t(\boldsymbol{w}) &\le c_t(\boldsymbol{w}) && (19)\\
& && (20)
\end{aligned}$$

by the definition of y_t^*, and

$$\hat{c}_t(\boldsymbol{w}) \ge \rho c_t(\boldsymbol{w}) - (1-\rho)\boldsymbol{w}^\top f_t(\boldsymbol{y}^{(t)}) + (1-\rho)\frac{\lambda}{2}\|\boldsymbol{w}\|^2 \tag{21}$$

by the definition of $\hat{y}_t$.

Therefore, we see $\hat{\boldsymbol{g}}_t$ is not necessarily a $(1-\rho)$-subgradient of c_t at $\boldsymbol{w}$ due to $C_t(\boldsymbol{w})$, where

$$C_t(\boldsymbol{w}) \triangleq (1-\rho)\left(\boldsymbol{w}^\top f_t(\boldsymbol{y}^{(t)}) - \frac{\lambda}{2}\|\boldsymbol{w}\|^2\right). \tag{22}$$

B Proof of Theorem 1

Proof. Let

$$L(\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{w}}) = \hat{\boldsymbol{w}}^\top \boldsymbol{f}(\boldsymbol{x}, h(\boldsymbol{x};\hat{\boldsymbol{w}})) + \ell_{\boldsymbol{y}}(h(\boldsymbol{x};\hat{\boldsymbol{w}})) - \hat{\boldsymbol{w}}^\top \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}). \tag{23}$$

Following [30], we adapt a result in [5] to get for any δ, with probability at least $1-\delta$

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[L(\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{w}})] \le \frac{1}{T}\sum_{t=1}^{T}\hat{\ell}_t(\boldsymbol{w}_t) + B\sqrt{\frac{2}{T}\log\frac{1}{\delta}}. \tag{24}$$

note that $L(\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{w}}) \ge \ell_{\boldsymbol{y}}(h(\boldsymbol{x};\hat{\boldsymbol{w}}))$ 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

$$c_t(\boldsymbol{w}^*) \ge c_t(\boldsymbol{w}) + \nabla c_{\boldsymbol{w}}^\top c_t(\boldsymbol{w}^* - \boldsymbol{w}) + \frac{\lambda}{4}\|\boldsymbol{w}^* - \boldsymbol{w}\|^2. \tag{25}$$

when an approximate gradient $\hat{\boldsymbol{g}}_t$ is generated with ρ-approximate LAI, we have

$$c_t(\boldsymbol{w}^*) \ge c_t(\boldsymbol{w}_t) + \hat{\boldsymbol{g}}_t^\top(\boldsymbol{w}^* - \boldsymbol{w}_t) + \frac{\lambda}{4}\|\boldsymbol{w}^* - \boldsymbol{w}_t\|^2 - (1-\rho)c_t(\boldsymbol{w}_t) - C_t(\boldsymbol{w}_t). \tag{26}$$

Rearrange to get

$$\rho c_t(\boldsymbol{w}_t) - c_t(\boldsymbol{w}^*) - C_t(\boldsymbol{w}_t) \le \hat{\boldsymbol{g}}_t^\top(\boldsymbol{w}_t - \boldsymbol{w}^*) - \frac{\lambda}{4}\|\boldsymbol{w}^* - \boldsymbol{w}_t\|^2 \tag{27}$$

We upper-bound $\hat{\boldsymbol{g}}_t^\top(\boldsymbol{w}^* - \boldsymbol{w}_t)$. Using the update for $\boldsymbol{w}_{t+1}$, we get

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2 \le \|\boldsymbol{w}_t - \eta_{t+1}\hat{\boldsymbol{g}}_t - \boldsymbol{w}^*\|^2, \tag{28}$$

and hence

$$\hat{\boldsymbol{g}}^\top(\boldsymbol{w}_t - \boldsymbol{w}^*) \le \frac{\|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2}{2\eta_{t+1}} + \frac{\eta_{t+1}}{2}G^2. \tag{29}$$

Using above and the fact that $\eta_t = \frac{2}{\lambda t}$, summing Eq. 27 from $t = 1, \cdots, T$, we have

$$\begin{aligned}\sum_{t=1}^{T}(\rho c_t(\boldsymbol{w}_t) - c_t(\boldsymbol{w}^*) - C_t(\boldsymbol{w}_t))) &\le \sum_{t=1}^{T}\|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2\left(\frac{1}{2\eta_{t+1}} - \frac{1}{2\eta_t} - \frac{\lambda}{4}\right) + \frac{G^2}{2}\sum_{t=1}^{T}\eta_{t+1} && (30)\\ &= 0 + G^2\sum_{t=1}^{T}\frac{1}{\lambda(t+1)} && (31)\\ &\le \frac{G^2}{\lambda}(1+\log T), && (32)\end{aligned}$$

which gives us

$$\sum_{t=1}^{T}\rho c_t(\boldsymbol{w}_t) \le \sum_{t=1}^{T}(c_t(\boldsymbol{w}^*) + C_t(\boldsymbol{w}_t)) + \frac{G^2}{\lambda}(1+\log T). \tag{33}$$

Using the facts that $c_t(\boldsymbol{w}) = \hat{\ell}_t(\boldsymbol{w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$, we have

$$\begin{aligned}&\rho\frac{1}{T}\sum_{t=1}^{T}\hat{\ell}_t(\boldsymbol{w}_t) \le && (34)\\ &\frac{1}{T}\sum_{t=1}^{T}\hat{\ell}_t(\boldsymbol{w}^*) + \frac{\lambda}{2}\left(\|\boldsymbol{w}^*\|^2 - \frac{1}{T}\sum_{t=1}^{T}\|\boldsymbol{w}_t\|^2\right) + \frac{1-\rho}{T}\sum_{t=1}^{T}\boldsymbol{w}_t^\top\boldsymbol{f}_t(\boldsymbol{y}^{(t)}) + \frac{G^2}{\lambda T}(1+\log T) && (35)\\ &\le \frac{1}{T}\sum_{t=1}^{T}\hat{\ell}_t(\boldsymbol{w}^*) + \frac{M^2\lambda}{2} + \frac{G^2}{\lambda T}(1+\log T) + M(1-\rho) && (36)\end{aligned}$$

Combining above with Eq. 24 and choosing $\lambda = \frac{G}{M}\sqrt{\frac{2(1+\log T)}{T}}$, we have the theorem. □

C Proof of Theorem 2

Proof. The Pyramid method is one of the manual evaluation metrics that has been used in recent TAC summarization track.[4] 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 have been expressed in the summary. Scores are then computed as a ratio of the sum of the weighted SCUs found in machine summary over the sum of the weights of the SCUs in an ideal summary. The sum of weights of the SCUs found in a summary is monotone submodular, and therefore scores used in Pyramid method are monotone submodular. In particular, we have the following theorem.

Denote $\Gamma(A)$ as the set of SCUs contained in summary A, and w_i as the weight for SCU i. Following the modified score defined in [39], we have

$$f_{\text{pyramid}}(A) = \frac{\sum_{i \in \Gamma(A)} w_i}{\sum_{i \in \Omega_m} w_i} \tag{37}$$

where

$$\Omega_m \triangleq \underset{B \in V, |B|=m}{\operatorname{argmax}} \sum_{i \in B} w_i,$$

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, f_{pyramid} is therefore monotone submodular. □

D Submodular functions in [46]

The representation of the submodular score function in [46] turns out to be a special case of a submodular shell 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 the function proposed in [27], say f_{penalty}, is linear in $\sigma_{i,j}$ and the coverage function in [46], say 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

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

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

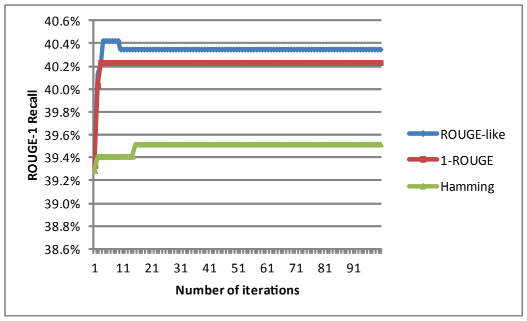


Figure 1: Convergence of ROUGE-1 recall for different loss functions.

we have

$$\begin{aligned} f_{\text{penalty}}(S) &= \sum_{i \in V \setminus S} \sum_{j \in S} \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}, i, j) - \lambda \sum_{i,j \in S : i \neq j} \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}, i, j) \\ &= \boldsymbol{w}^\top \left(\sum_{i \in V \setminus S} \sum_{j \in S} \boldsymbol{\phi}(\boldsymbol{x}, i, j) - \lambda \sum_{i,j \in S : i \neq j} \boldsymbol{\phi}(\boldsymbol{x}, i, j) \right) \\ &= \sum_k w_k \left(\sum_{i \in V \setminus S} \sum_{j \in S} \phi_k(\boldsymbol{x}, i, j) - \lambda \sum_{i,j \in S : i \neq j} \phi_k(\boldsymbol{x}, i, j) \right) \\ &= \sum_k w_k f^{\text{penalty}}_{\lambda, (V, \phi_k)}(S), \end{aligned}$$

which is exactly a submodular shell mixture with components being a submodular shell:

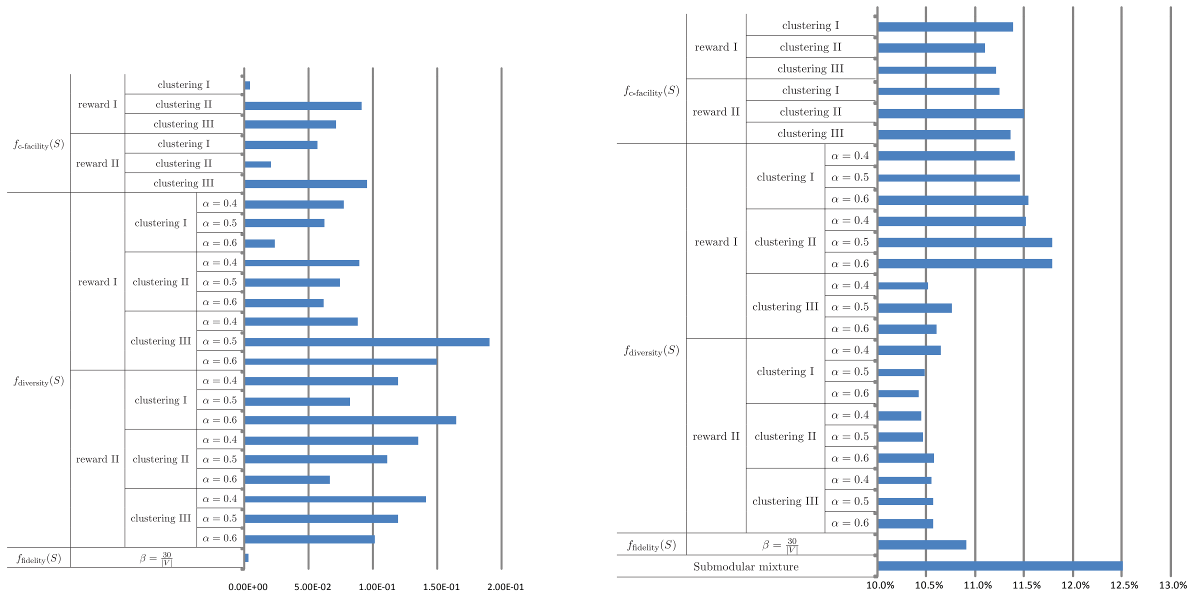
$$f^{\text{penalty}}_{\lambda, (V, \phi_k)}(S) = \sum_{i \in V \setminus S} \sum_{j \in S} \phi_k(\boldsymbol{x}, i, j) - \lambda \sum_{i,j \in S : i \neq j} \phi_k(\boldsymbol{x}, i, j). \tag{38}$$

E Comparison of loss functions for Summarization

We also compared the performance of using different loss functions in the subgradient algorithm for summarization. Two other loss functions are considered: the hamming loss, and the one-minus-ROUGE loss (non-submodular objective, as used in [46]).

Note that 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, we can use the greedy algorithm to optimize it near-optimally, given the human references. The outputs of the greedy algorithm over ROUGE-N score functions are then used as pseudo labels. The pseudo labels were then used in computing the hamming loss. For the ROUGE-like loss function we proposed in Section 5.3, no extractive labels are required, and thus pseudo labels we generated were only used for hamming loss in our experiments.

The ROUGE-1 recall score improvement as the number of iterations increases in the subgradient algorithm is shown in Figure 1. 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.



(a) Component weight (b) Component ROUGE-2 Recall score

Figure 2: Visualization of component weights and ROUGE-2 recall scores. Each single component and the submodular shell mixture were trained on DUC-05 and DUC-06 and tested on DUC-07.

F Single Component Performance

Figure 2 visualizes the component weights and single component performance. In particular, the left figure shows the weights of the components in the submodular shell mixture, while the right figure shows the ROUGE-2 recall score of each component when used alone as a score function. As we can see, the performance of the mixture outperforms that of each individual component, indicating the effectiveness of using mixtures. Moreover, note that the best performing component does not necessary have the highest weight in the mixture, suggesting that a naive approach of assigning higher weights to components with better individual performance might not work as well as our approach of submodular shell mixture learning.