Applications of Submodular Functions in Speech and NLP

Jeff Bilmes

Department of Electrical Engineering
University of Washington, Seattle
`http://ssli.ee.washington.edu/~bilmes`

June 27, 2011

Summary

- Submodular functions are a powerful class of functions on discrete sets.
- We show that they naturally represent the problem of document summarization, and show good results.
- We also apply them to data set selection.

Acknowledgments and Key References

- This is joint work with my student, *Hui Lin*.
- Hui Lin and Jeff A. Bilmes. Optimal Selection of Limited Vocabulary Speech Corpora. In *Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Florence, Italy, August 2011.
- Hui Lin and Jeff Bilmes. A Class of Submodular Functions for Document Summarization. In *North American chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL/HLT-2011)*, Portland, OR, June 2011.

Outline

Outline

Rank function of a matrix

- Given an $n \times m$ matrix $\mathbf{X} = (x_1, x_2, \ldots, x_m)$ consisting of length n column vectors $\{x_i\}_i$
- The rank of the matrix $\mathbf{X}$ is $r \leq \min(m, n)$.
- Let E be the set of column indices, $E = \{1, 2, \ldots, m\}$, called the ground set.
- For any $A \subseteq E$, let $r(A)$ be the rank of the vectors indexed by A.
- $r(A)$ is the dimensionality of the vector space spanned by the vectors $x_{a_1}, x_{a_2}, \ldots, x_{a_\ell}$ where $|A| = \ell$ and $A = \{a_1, a_2, \ldots, a_\ell\}$.
- Intuitively, $r(A)$ is the size of the largest set of independent vectors contained within the set of vectors indexed by A.
- Thus, $r(E)$ is the rank of the matrix $\mathbf{X}$.

Example: independence in linear space

Consider the following 3×8 matrix.

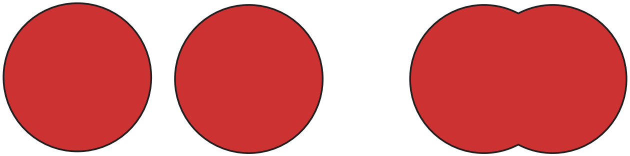
$$\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 0 & 0 & 1 & 1 & 2 & 1 & 3 & 1 \\ 2 & 0 & 1 & 1 & 0 & 2 & 0 & 2 & 4 \\ 3 & 1 & 1 & 1 & 0 & 0 & 3 & 1 & 5 \end{array} = \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ | & | & | & | & | & | & | & | \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ | & | & | & | & | & | & | & | \end{array}$$

- Consider the column index set $E = \{1, 2, \ldots, 8\}$. Then any subset $A \subseteq E$ corresponds to a set of column vectors.
- We can define the rank function on subsets of E - the rank function of $A \subseteq E$ is the number of linearly independent column vectors within A.
- E.g., $\text{rank}(\{4, 5, 6\}) = \text{rank}(\{4, 7, 8\}) = \text{rank}(\{1, 2, 3\}) = 3$ while $\text{rank}\{1, 2\} = \text{rank}(\{1, 4, 6\}) = 2$.

Rank function

- Let $A, B \subseteq E$ be two subsets of column indices.
- When we look at the rank of the two sets unioned together $A \cup B$, the rank will be no more than the sum of the two individual ranks.
- That is

$$r(A) \quad + \quad r(B) \qquad \geq \qquad r(A \cup B)$$



- If some of the dimensions spanned by A overlap some of the dimensions spanned by B, then the inequality will be strict.
- Any function where the above inequality is true for all $A, B \subseteq E$ is called subadditive.

Rank functions

- Given A, let span(A) be the vector subspace spanned by vectors indexed by A.
- Sets A and B will have some common span (possibly empty) and each have some non-common residual span (also possibly empty).
- $r(A) = r(C) + r(A_r)$ where C is a set that spans the subspace common to that spanned by A and B, and A_r is a "residual" set that spans what is spanned by A but not spanned by B.
- Similarly, $r(B) = r(C) + r(B_r)$.
- Then $r(A) + r(B)$ counts the dimensions spanned by C twice, i.e.,

$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r). \tag{1}$$

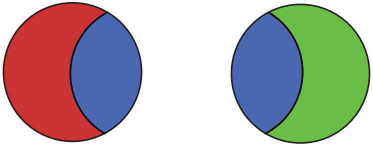
- But $r(A \cup B)$ counts the dimensions spanned by C only once.

$$r(A \cup B) = r(A_r) + r(C) + r(B_r) \tag{2}$$

Rank functions

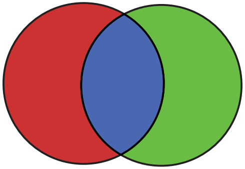
- Then $r(A) + r(B)$ counts the dimensions spanned by C twice, i.e.,

$$r(A) + r(B) = r(A_r) + 2r(C) + r(B_r)$$



- But $r(A \cup B)$ counts the dimensions spanned by C only once.

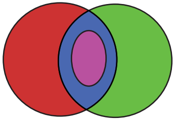
$$r(A \cup B) = r(A_r) + r(C) + r(B_r)$$



Rank function

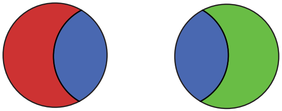
- On the other hand, $r(A \cap B) \leq r(C)$. This follows since the vectors indexed by $A \cap B$ span no more than the dimensions commonly spanned by vectors indexed by A and vectors indexed by B.

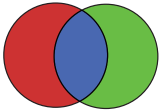
$$r(C) \geq r(A \cap B)$$

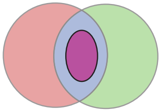


- Therefore, with (matrix) rank functions, the lower bound can be tightened as follows.

$$\begin{array}{cccc} r(A) + r(B) & & r(A \cup B) & \\ = r(A_r) + 2r(C) + r(B_r) & \geq & = r(A_r) + r(C) + r(B_r) & + \quad r(A \cap B) \end{array}$$







Generalized Abstract Independence

- This inequality apparently captures an essential property of "independence" within rank, and in fact defines abstract independence.
- Any function $f : 2^E \to \mathbb{R}$ that satisfies the above is called submodular

Definition

A function $f : 2^E \to \mathbb{R}$ is called submodular if, for all $A, B \subseteq E$, the following inequality holds:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \tag{3}$$

Diminishing Returns

An alternate and equivalent definition of submodularity is:

Definition

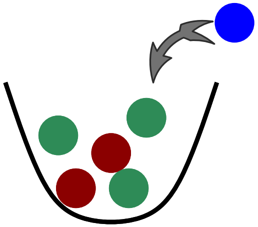
A function $f : 2^E \to \mathbb{R}$ is called submodular if for any $A \subseteq B \subset E$, and $v \in E \setminus B$, we have that:

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B) \quad (4)$$

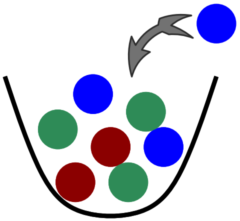
This means that the incremental "value", "gain", or "cost" of v decreases (diminishes) as the context in which v is considered grows from A to B.

Example Submodular: Number of Colors of Balls in Urns

- Consider an urn containing colored balls. Given a set S of balls, $f(S)$ counts the number of distinct colors.
- Submodularity: Incremental Value of Object Diminishes in a Larger Context.



Initial value: 2 (colors in urn).
New value with added blue ball: 3



Initial value: 3 (colors in urn).
New value with added blue ball: 3

- Thus, f is submodular.

Submodular Functions

- Many examples of submodular functions in economics, game theory, combinatorial optimization, electrical networks, and operations research.
- Examples include:
    - The entropy function $f(A) = H(X_A)$ for set of random variables X_E.

Submodular Functions

- Many examples of submodular functions in economics, game theory, combinatorial optimization, electrical networks, and operations research.
- Examples include:
  - The entropy function $f(A) = H(X_A)$ for set of random variables X_E.
  - Social network influence.

Submodular Functions

- Many examples of submodular functions in economics, game theory, combinatorial optimization, electrical networks, and operations research.
- Examples include:
  - The entropy function $f(A) = H(X_A)$ for set of random variables X_E.
  - Social network influence.
  - Sensor placement (coverage functions).

Submodular Functions

- Many examples of submodular functions in economics, game theory, combinatorial optimization, electrical networks, and operations research.
- Examples include:
    - The entropy function $f(A) = H(X_A)$ for set of random variables X_E.
    - Social network influence.
    - Sensor placement (coverage functions).
    - Many feature selection objectives (e.g., certain instances of mutual information).

Submodular Functions

- Many examples of submodular functions in economics, game theory, combinatorial optimization, electrical networks, and operations research.
- Examples include:
  - The entropy function $f(A) = H(X_A)$ for set of random variables X_E.
  - Social network influence.
  - Sensor placement (coverage functions).
  - Many feature selection objectives (e.g., certain instances of mutual information).
  - Graph cuts and hypergraph cuts.

Submodular Functions

- Many examples of submodular functions in economics, game theory, combinatorial optimization, electrical networks, and operations research.
- Examples include:
  - The entropy function $f(A) = H(X_A)$ for set of random variables X_E.
  - Social network influence.
  - Sensor placement (coverage functions).
  - Many feature selection objectives (e.g., certain instances of mutual information).
  - Graph cuts and hypergraph cuts.
  - Matrix rank functions (more generally, matroid rank functions).

Submodular Functions

- Many examples of submodular functions in economics, game theory, combinatorial optimization, electrical networks, and operations research.
- Examples include:
  - The entropy function $f(A) = H(X_A)$ for set of random variables X_E.
  - Social network influence.
  - Sensor placement (coverage functions).
  - Many feature selection objectives (e.g., certain instances of mutual information).
  - Graph cuts and hypergraph cuts.
  - Matrix rank functions (more generally, matroid rank functions).
  - Economies of scale functions.

Submodular Functions

- Many examples of submodular functions in economics, game theory, combinatorial optimization, electrical networks, and operations research.
- Examples include:
    - The entropy function $f(A) = H(X_A)$ for set of random variables X_E.
    - Social network influence.
    - Sensor placement (coverage functions).
    - Many feature selection objectives (e.g., certain instances of mutual information).
    - Graph cuts and hypergraph cuts.
    - Matrix rank functions (more generally, matroid rank functions).
    - Economies of scale functions.
    - Value of information functions.

Minimization of Submodular Functions

- Computing $A^* \in \operatorname{argmin}_{A \subseteq E} f(A)$ is intractable, in general.
- For submodular functions this can be done in polynomial time.
- Current worst case strongly polynomial algorithms are about $O(n^6)$, so not practical.
- There exists algorithms that, in practice, often have run-time of about $O(n^{3.3})$ (min-norm algorithm).
- Many special cases (e.g., graph-representable, cuts, etc.) run much faster.

Maximization of Non-Decreasing Submodular Functions

- The problem is in general NP-hard (reduction from max-cut).
- An important result by Nemhauser et. al. (1978) states that for normalized ($f(\emptyset) = 0$) monotone submodular functions can be maximized using a simple greedy algorithm.
- Starting with $S_0 = \emptyset$, we repeat

$$S_{i+1} = S_i \cup \left\{ \operatorname*{argmax}_{v \in V \setminus S_i} f(S_i \cup \{v\}) \right\} \tag{5}$$

- This algorithm has guarantee $f(S_i) \geq (1 - 1/e) \max_{|S| \leq i} f(S)$.
- Feige (1998) showed that this can't be improved. Unless $P = NP$, no polynomial time algorithm can do better than $(1 - 1/e + \epsilon)$ for any $\epsilon > 0$.

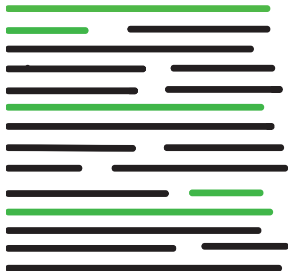
Outline

Extractive Document Summarization

- The figure below represents the sentences of a document

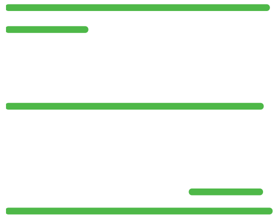
Extractive Document Summarization

- We extract sentences (green) as a summary of the full document



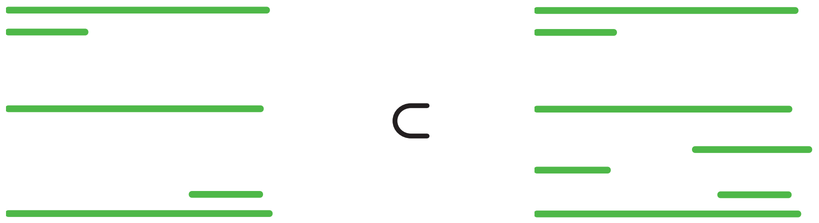
Extractive Document Summarization

- We extract sentences (green) as a summary of the full document



Extractive Document Summarization

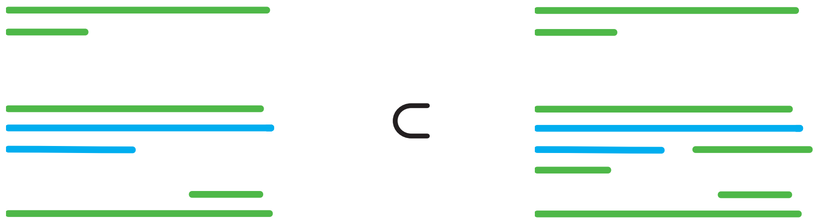
- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.

Extractive Document Summarization

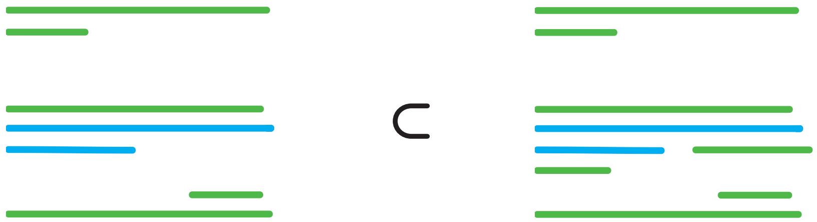
- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.
- Consider adding a new (blue) sentence to each of the two summaries.

Extractive Document Summarization

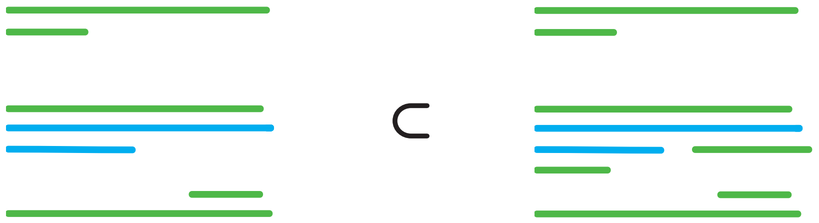
- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.
- Consider adding a new (blue) sentence to each of the two summaries.
- The marginal (incremental) benefit of adding the new (blue) sentence to the smaller (left) summary is no more than the marginal benefit of adding the new sentence to the larger (right) summary.

Extractive Document Summarization

- We extract sentences (green) as a summary of the full document



- The summary on the left is a subset of the summary on the right.
- Consider adding a new (blue) sentence to each of the two summaries.
- The marginal (incremental) benefit of adding the new (blue) sentence to the smaller (left) summary is no more than the marginal benefit of adding the new sentence to the larger (right) summary.
- **diminishing returns** $\leftrightarrow$ **submodularity**

Problem setup

- The ground set V corresponds to all the sentences in a document.
- Extractive document summarization: select a small subset $S \subseteq V$ that accurately represents the entirety (ground set V).

Problem setup

- The ground set V corresponds to all the sentences in a document.
- Extractive document summarization: select a small subset $S \subseteq V$ that accurately represents the entirety (ground set V).
- The summary is usually required to be length-limited.
  - c_i: cost (e.g., the number of words in sentence i),
  - b: the budget (e.g., the largest length allowed),
  - knapsack constraint: $\sum_{i \in S} c_i \leq b$.

Problem setup

- The ground set V corresponds to all the sentences in a document.
- Extractive document summarization: select a small subset $S \subseteq V$ that accurately represents the entirety (ground set V).
- The summary is usually required to be length-limited.
  - c_i: cost (e.g., the number of words in sentence i),
  - b: the budget (e.g., the largest length allowed),
  - knapsack constraint: $\sum_{i \in S} c_i \leq b$.
- A set function $f : 2^V \to \mathbb{R}$ measures the quality of the summary S,
- Thus, the summarization problem is formalized as:

Problem (Document Summarization Optimization Problem)

$$S^* \in \operatorname*{argmax}_{S \subseteq V} f(S) \text{ subject to: } \sum_{i \in S} c_i \leq b. \tag{6}$$

A Practical Algorithm for Large-Scale Summarization

When f is both **monotone** and **submodular**:

- A greedy algorithm with partial enumeration (Sviridenko, 2004), theoretical guarantee of near-optimal solution, but not practical for large data sets.

A Practical Algorithm for Large-Scale Summarization

When f is both **monotone** and **submodular**:

- A greedy algorithm with partial enumeration (Sviridenko, 2004), theoretical guarantee of near-optimal solution, but not practical for large data sets.
- A greedy algorithm (Lin and Bilmes, 2010): near-optimal with theoretical guarantee, and practical/scalable!
    - We choose next element with largest ratio of gain over **scaled** cost:

$$k \leftarrow \underset{i \in U}{\operatorname{argmax}} \frac{f(G \cup \{i\}) - f(G)}{(c_i)^r}. \tag{7}$$

A Practical Algorithm for Large-Scale Summarization

When f is both **monotone** and **submodular**:

- A greedy algorithm with partial enumeration (Sviridenko, 2004), theoretical guarantee of near-optimal solution, but not practical for large data sets.
- A greedy algorithm (Lin and Bilmes, 2010): near-optimal with theoretical guarantee, and practical/scalable!
  - We choose next element with largest ratio of gain over **scaled** cost:

$$k \leftarrow \operatorname*{argmax}_{i \in U} \frac{f(G \cup \{i\}) - f(G)}{(c_i)^r}. \tag{7}$$

  - Scalability: the argmax above can be solved by $O(\log n)$ calls of f, thanks to submodularity
  - Integer linear programming (ILP) takes 17 hours vs. greedy which takes < 1 second!!

Objective Function Optimization: Performance in Practice

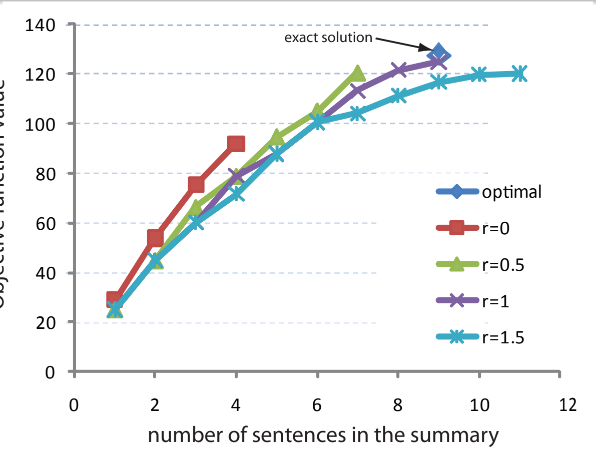


Figure: The plots show the achieved objective function value as the number of selected sentences grows. The plots stop when in each case adding more sentences violates the budget.

The General Form of Our Submodular Functions

- Two properties of a good summary: relevance and non-redundancy.
- Common approaches (e.g., MMR): encourage relevance and (negatively) penalize redundancy.
- The redundancy penalty is usually what violates monotonicity.
- Our approach: we positively **reward diversity** instead of negatively penalizing redundancy:

The General Form of Our Submodular Functions

- Two properties of a good summary: relevance and non-redundancy.
- Common approaches (e.g., MMR): encourage relevance and (negatively) penalize redundancy.
- The redundancy penalty is usually what violates monotonicity.
- Our approach: we positively **reward diversity** instead of negatively penalizing redundancy:

Definition (The general form of our submodular functions)

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

- $\mathcal{L}(S)$ measures the coverage (or fidelity) of summary set S to the document.
- $\mathcal{R}(S)$ rewards diversity in S.
- $\lambda \geq 0$ is a trade-off coefficient.

Coverage function

Coverage Function

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

- $\mathcal{C}_i : 2^V \to \mathbb{R}$ is monotone submodular, and measures how well i is covered by S: $\Rightarrow \mathcal{L}(S)$ is monotone submodular.
- $0 \leq \alpha \leq 1$ is a threshold coefficient — sufficient coverage fraction.

Coverage function

Coverage Function

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

- $\mathcal{C}_i : 2^V \to \mathbb{R}$ is monotone submodular, and measures how well i is covered by S: $\Rightarrow \mathcal{L}(S)$ is monotone submodular.
- $0 \leq \alpha \leq 1$ is a threshold coefficient — sufficient coverage fraction.
- if $\min\{\mathcal{C}_i(S), \alpha\mathcal{C}_i(V)\} = \alpha\mathcal{C}_i(V)$, then sentence i is well covered by summary S (saturated).

Coverage function

Coverage Function

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

- $\mathcal{C}_i : 2^V \to \mathbb{R}$ is monotone submodular, and measures how well i is covered by S: $\Rightarrow \mathcal{L}(S)$ is monotone submodular.
- $0 \leq \alpha \leq 1$ is a threshold coefficient — sufficient coverage fraction.
- if $\min\{\mathcal{C}_i(S), \alpha\mathcal{C}_i(V)\} = \alpha\mathcal{C}_i(V)$, then sentence i is well covered by summary S (saturated).
- After saturation, further increases in $\mathcal{C}_i(S)$ won't increase the objective function values (return diminishes).
- Therefore, new sentence added to S should focus on sentences that are not yet saturated, in order to increasing the objective function value.

Coverage function

Coverage Function

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

- $\mathcal{C}_i$ measures how well i is covered by S.
- One simple possible $\mathcal{C}_i$ (that we use in our experiments and works well) is:

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

  where $w_{i,j} \geq 0$ measures the similarity between i and j.
- With this C_i, $\mathcal{L}(S)$ is monotone submodular, as required.

Diversity reward function

Diversity Reward Function

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

- $P_i, i = 1, \cdots K$ is a partition of the ground set V
- $r_j \geq 0$: **singleton reward** of j, which represents the importance of j to the summary.
- square root over the sum of rewards of sentences belong to the same partition (diminishing returns).
- $\mathcal{R}(S)$ is monotone submodular as well.

Diversity Reward Function

- **singleton reward** of j: the importance of being j (to the summary).
  - Query-independent (generic) case:
$$r_j = \frac{1}{N} \sum_{i \in V} w_{i,j}.$$
  - Query-dependent case, given a query Q,
$$r_j = \beta \frac{1}{N} \sum_{i \in V} w_{i,j} + (1 - \beta) r_{j,Q}$$
    where $r_{j,Q}$ measures the relevance between j and query Q.

Diversity Reward Function

- **singleton reward** of j: the importance of being j (to the summary).
    - Query-independent (generic) case:
    $$r_j = \frac{1}{N} \sum_{i \in V} w_{i,j}.$$
    - Query-dependent case, given a query Q,
    $$r_j = \beta \frac{1}{N} \sum_{i \in V} w_{i,j} + (1 - \beta) r_{j,Q}$$
    where $r_{j,Q}$ measures the relevance between j and query Q.

Multi-resolution Diversity Reward

$$\mathcal{R}(S) = \sum_{i=1}^{K_1} \sqrt{\sum_{j \in P_i^{(1)} \cap S} r_j} + \sum_{i=1}^{K_2} \sqrt{\sum_{j \in P_i^{(2)} \cap S} r_j} + \cdots$$

Generic Summarization

- DUC-04: generic summarization

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

DUC-04	R	F
$\mathcal{L}_1(S)$	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 and Okumura (2009)	38.50	-
Wang et al. (2009)	39.07	-
Lin and Bilmes (2010)	-	38.39
Best system in DUC-04 (peer 65)	38.28	37.94

Generic Summarization

- DUC-04: generic summarization

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

DUC-04	R	F
$\mathcal{L}_1(S)$	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 and Okumura (2009)	38.50	-
Wang et al. (2009)	39.07	-
Lin and Bilmes (2010)	-	38.39
Best system in DUC-04 (peer 65)	38.28	37.94

- Note: this is the best ROUGE-1 result ever reported on DUC-04.

Query-focused Summarization

- DUC-05,06,07: query-focused summarization
- For each document cluster, a title and a narrative (query) describing a user's information need are provided.
- Nelder-Mead (derivative-free) for parameter training.

DUC-05 results

Table: ROUGE-2 recall (R) and F-measure (F) results (%)

	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 and Marcu (2006)	6.98	-
Wei et al. (2010)	8.02	-
Best system in DUC-05 (peer 15)	7.44	7.43

- DUC-06 was used as training set for the objective function with single diversity reward.
- DUC-06 and 07 were used as training sets for the objective function with multi-resolution diversity reward

DUC-05 results

Table: ROUGE-2 recall (R) and F-measure (F) results (%)

	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 and Marcu (2006)	6.98	-
Wei et al. (2010)	8.02	-
Best system in DUC-05 (peer 15)	7.44	7.43

- DUC-06 was used as training set for the objective function with single diversity reward.
- DUC-06 and 07 were used as training sets for the objective function with multi-resolution diversity reward
- Note: this is the best ROUGE-2 result ever reported on DUC-05.

DUC-06 results

Table: ROUGE-2 recall (R) and F-measure (F) results (%)

	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 and Hakkani-tür (2010)	9.10	-
Shen and Li (2010)	9.30	-
Best system in DUC-06 (peer 24)	9.51	9.51

- DUC-05 was used as training set for the objective function with single diversity reward.
- DUC-05 and 07 were used as training sets for the objective function with multi-resolution diversity reward

DUC-06 results

Table: ROUGE-2 recall (R) and F-measure (F) results (%)

	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 and Hakkani-tür (2010)	9.10	-
Shen and Li (2010)	9.30	-
Best system in DUC-06 (peer 24)	9.51	9.51

- DUC-05 was used as training set for the objective function with single diversity reward.
- DUC-05 and 07 were used as training sets for the objective function with multi-resolution diversity reward
- Note: this is the best ROUGE-2 result ever reported on DUC-06.

DUC-07 results

Table: ROUGE-2 recall (R) and F-measure (F) results (%)

	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 and Vanderwende (2009)	11.80	-
Celikyilmaz and Hakkani-tür (2010)	11.40	-
Best system in DUC-07 (peer 15), using web search	**12.45**	12.29

- DUC-05 was used as training set for the objective function with single diversity reward.
- DUC-05 and 06 were used as training sets for the objective function with multi-resolution diversity reward.

DUC-07 results

Table: ROUGE-2 recall (R) and F-measure (F) results (%)

	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 and Vanderwende (2009)	11.80	-
Celikyilmaz and Hakkani-tür (2010)	11.40	-
Best system in DUC-07 (peer 15), using web search	**12.45**	12.29

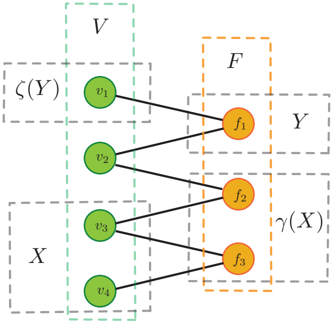
- DUC-05 was used as training set for the objective function with single diversity reward.
- DUC-05 and 06 were used as training sets for the objective function with multi-resolution diversity reward.
- Note: this is the best ROUGE-2 F-measure result ever reported on DUC-07, and best ROUGE-2 R without web search expansion.

Outline

Corpus Selection: motivation

- Machine learning: complexity is often linear in number of samples but polynomial in the number of types of objects.
- Canonical example: speech recognition: adding more training samples is relatively easy, except when the vocabulary expands (e.g., $O(N^3)$ or $O(N^4)$).
- This inhibits rapid turnaround time for novel and expensive surface methods (e.g., acoustic modeling in speech recognition).
- Goal: find a way to select a subset of the data while limiting the number of types.

Corpus Selection: description



- Bipartite graph (V, F, E) where V is the set of utterances (sentences) and F is the set of words.
- $w(\gamma(X))$ is submodular, where $w : 2^V \to \mathbb{R}$ is a modular function;
- $w(\zeta(Y))$ is supermodular.
- **King et al. 2005**: maximizing $w(\zeta(Y))$ with cardinality constraint, using greedy algorithm. This can do unboundedly poorly.

Corpus Selection: when supermodular greedy fails

- $F = \{x, y, z\}$ with $f(\{x\}) = 1$, $f(\{y\}) = f(\{z\}) = 0$, $f(\{x, y\}) = f(\{x, z\}) = 1$, and $f(\{y, z\}) = p > 1$ and $b = 2$.
- Greedily maximizing f leads to a solution $\{x, y\}$ having objective function value 1
- The true optimal objective function value is p.
- Since p is arbitrary, the approximation factor for this example is unbounded.

Our approach

- We find a set of words $X \subseteq V$ that maximizes the following expression
$$w(X) - \lambda \Gamma(X) \tag{8}$$
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.
- Larger λ prefers smaller corpora.
- this is identical to minimizing
$$L(\lambda, X) \triangleq w(V \setminus X) + \lambda \Gamma(X). \tag{9}$$
which is a submodular function minimization problem.
- But both $|V|$ and $|F|$ are big, and our control parameter is λ, and we want to find solution for all values $\lambda \in \mathbb{R}$
- How can this be made practical w/o optimality loss?

Principle Partition

- Goal: for all $\lambda \in \mathbb{R}_+$, minimize

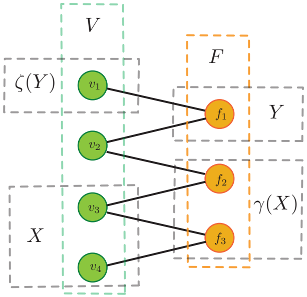
$$L(\lambda, X) \triangleq w(V \setminus X) + \lambda \Gamma(X). \tag{9}$$

- Let $\mathcal{X}(\lambda) = \text{argmin}_{X \subseteq V} L(\lambda, X)$ be the set of minima for value λ.
- Let $\mathcal{X}^-(\lambda) \in \text{argmin}_{X \subseteq V} L(\lambda, X)$ be the smallest minima for value λ.
- Let $\mathcal{X}^+(\lambda) \in \text{argmin}_{X \subseteq V} L(\lambda, X)$ be the largest minima for value λ.

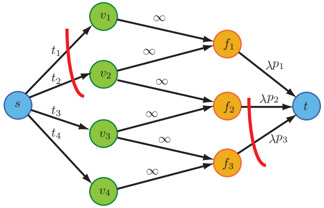
Theorem (Theorem 7.15 in Fujishige-2005)

If Γ is non-decreasing submodular. Then there exists a sequence of real values $\lambda_1 < \lambda_2 < \cdots < \lambda_p$ (the critical points) where $p \leq |V|$ such that the distinct $\mathcal{X}(\lambda), \lambda \in \mathbb{R}_+$ are given by: $\mathcal{X}(\lambda_i), i = 1, 2, \cdots, p$; For any $\lambda_i < \lambda < \lambda_{i+1}$ we have $\mathcal{X}(\lambda) = \{X^-(\lambda_i)\} = \{X^+(\lambda_{i+1})\} \triangleq \{X_i\}$.

Fast Parametric Max flow implementation



(a) Bipartite graph



(b) $s - t$ graph

Figure: In subfigure (a), $V = \{v_1, v_2, v_3, v_4\}$ and $F = \{f_1, f_2, f_3\}$. 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. 9.

A variety of sub-corpora

- Depending on Γ different sub-corpora result.
- $\Gamma_1(X) = |\gamma(X)|$. This represents the collective vocabulary size of utterances in set X.
- $\Gamma_2(X) = m(\gamma(X)) = \sum_{i \in \gamma(X)} p_i$, where p_i indicates the unimportance of word i. A larger p_i states that word i 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 $p_i = 1, \forall i$, then $\Gamma_2 = \Gamma_1$.

Results - try to get words with many phones

Corpus D,$\Gamma_2(X) = \sum_{i \in \gamma(X)} p_i = \sum_{i \in \gamma(X)} \frac{C}{q_i}$, where C is a constant, and q_i is the number of phonemes in the pronunciation of word i.

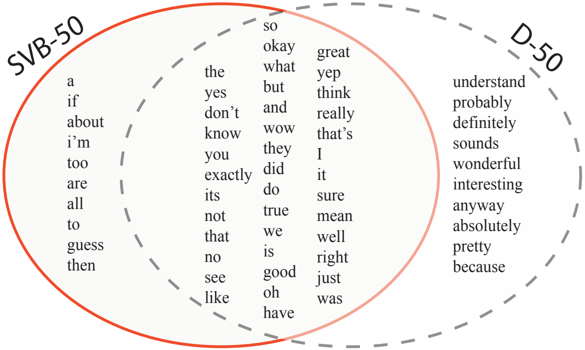


Figure: Venn diagram showing the vocabulary difference between SVitchboard-50 and D-50.

Outline

Summary

- Submodular functions are a powerful class of functions on discrete sets.
- We show that they naturally represent the problem of document summarization, and show good results.
- We also apply them to data set selection.