CS769: Summarizing Documents Using Submodular Functions

Harsh Diwakar [213070018] Pritish Chakraborty [21Q050011]

1 Introduction

Document summarization is the task of extracting a subset of documents from a collection, such that a performance or utility measure is maximized. One may consider *extractive* summarization, where words and phrases are extracted from the document to create a summary, or *abstractive* summarization, where the summary is built by learning a hidden representation of the text. In this project, we focus on extractive summarization using submodular functions.

We attempt this problem in two ways: one, where we experiment with multiple hyperparameter values using the submodular function defined in the paper, and two, where we try to implement a deep submodular function (DSF) which can be learned from the text. Note that even though the DSF is a learning-based method, it is used to build summaries by extracting words from the corpus of documents.

2 Background

In this section we provide some background related to submodular functions and their property of diminishing gains, greedy algorithms, Nemhauser's result and k-Means Clustering.

2.1 Submodular Functions

A set function $f(X) : 2^V \rightarrow \mathcal{R}$ defined on the ground set V and $X \subseteq V$ is submodular if it satisfies any of the below two equivalent properties:

$$\begin{gathered} \forall A, B \subseteq V, f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \\ \text{(or)} \\ \forall A \subseteq BV \text{ and } e \in V \backslash B, f(A \cup e) - f(A) \geq f(B \cup e) - f(B) \end{gathered} \tag{1}$$

Submodular functions have a natural property of diminishing returns that is the difference in the incremental value of the function that a single element makes when added to an input set decreases as the size of the input set increases. This property of submodular functions makes them suitable for many real world problems.

2.2 Greedy Algorithm

Greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. In case of set functions sometimes we want to maximize or minimize a function subject to

some constraints. Let us take a problem: $\max_{S \in V} f(S)$ s. t. $|S| \leq B$ where, $c(S) = |S|$ is a cost function and B is the budget allowance. A naive greedy algorithm is defined below. Greedy algorithm may

Algorithm 1 Naive Greedy algorithm

$S = \phi$
for $i = 0, 1, \ldots, B$ **do** :
  $e^* = arg \max_{e \in V \setminus S_i} f(S_i \cup \{e\})$
  $S_{i+1} = S_i \cup \{e^*\}$
end for

not always give optimal results however, one setting where the algorithm works quite well is defined next.

2.3 Nemhauser's result

Greedy algorithm gives a $(1 - 1/e)$-approximation (around 63%) for the problem of $\max_{|S| \leq B} f(S)$ when $f : 2^V \rightarrow R$ is a monotone submodular function[5]. In practical scenarios, this lower bound is often beaten by a large margin. This is what makes submodular functions especially attractive, recently so in machine learning.

3 Document Summarization with submodular functions

The paper by Bilmes et al.[4] describes a submodular two-part objective function for efficient document summarization. One objective functions measures the relevance of the summary to the whole document and the other one measures non-redundancy. The quality of summary is modelled as:

$$\mathcal{F}(S) = \mathcal{L}(S) + \lambda \mathcal{R}(S) \tag{2}$$

Where $\mathcal{L}(S)$ measures the coverage of summary set S to the document, $\mathcal{R}(S)$ rewards the diversity in S and λ is a trade-off coefficient. An another view of the above function in terms of machine learning loss is that the coverage function is combined with a regularizer term that encourages diversity. Now we further define the Coverage and Diversity function in detail.

3.1 Coverage Function

Interpretation of $\mathcal{L}$ can be as a set function that measures the similarity of summary set S with the document set or as function representing the coverage of document V by S. Authors have defined this function $\mathcal{L}(S)$ as:

$$\mathcal{L}(S) = \sum_{i \in V} \min\{C_i(S), _i(V)\} \tag{3}$$

where, $C_i : 2^V \rightarrow \mathcal{R}$ is a monotone function and $0 \leq \alpha \leq 1$ is a threshold coefficient. For our objectives we have picked up the same $C_i(S) = \sum_j w_{ij}$ for each sentence i where, $w_{ij} \geq 0$ measures the similarity between sentences i and j. In those terms, the overall coverage function becomes:

$$\mathcal{L}(S) = \sum_{i \in V} \min \left\{ \sum_{j \in S} w_{ij}, \alpha \sum_{k \in V} w_{ik} \right\} \tag{4}$$

3.2 Diversity Function

The submodular Diversity objective function defined by the authors is:

$$\mathcal{R}(S) = \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i \cap S} r_j} \tag{5}$$

where, $P_i, i = 1, \ldots K$ is of the ground set V (and $\cap_i P_i = V$ and also P_is are disjoint) into separate clusters, and $r_i \geq 0$ indicates the reward of adding i into the empty set. Reward of diversity in function $\mathcal{R}$ is such that there is more benefit in selecting a sentence from a cluster not yet having one of its elements already chosen than from a cluster with atleast one of the element already in set. Also thanks to the square root function, as soon as an element is selected from a cluster, other elements from the same cluster start having diminishing gains. For document summarization authors have defined $r_i = \frac{1}{N} \sum_j w_{ij}$, where N is the total number of sentences then r_i is nothing but average similarity of sentence i to the rest of the document. By using the this reward, we have the following reward function:

$$\mathcal{R}(S) = \sum_{k=1}^{K} \sqrt{\sum_{j \in S \cap P_k} \frac{1}{N} \sum_{i \in V} w_{ij}} \tag{6}$$

Also, for generating P_k we clustered the sentences using k-means clustering[3]. As $\mathcal{R}(S)$ is trying to enforce diversity in the summary, it might wish to include certain outlier material that $\mathcal{L}(S)$ might ignore.

4 Deep Submodular Functions

Through the rich history of submodular functions, and the blowup of machine learning in the last decade, there has arisen a need for general-purpose, learnable submodular functions, due to the desirable properties of these functions described earlier. To this end, Bilmes et al.[1] discover a family of submodular functions that are a strict generalization of many submodular functions in present literature. They are also flexible like submodular feature-based functions, while introducing interactions between "layers", much like a deep neural network, that feature functions do not have.

The authors point out that deep neural networks built with popular activation functions (tanh, sigmoid, relu) will behave like a DSF if the trainable weights at each layer are constrained to be non-negative. Moreover, they demonstrate a loss-augmented inference or max-margin training framework for DSFs, as opposed to the classification or regression training regimes. This is because a desirable "summary" or "subset"'s score by our neural network is made to have a margin in the loss function over other candidate subsets.

The loss-augmented inference framework takes the general form -:

$$\min_{w \geq 0} \sum_{S \in \mathcal{S}} \left(\max_{A \in 2^V} [f(A) + l_S(A)] - f(S) \right)_+ + \frac{\lambda}{2} ||w||_2^2 \tag{7}$$

Note that we use a variant of the above loss in our code in which $l_S(A)$ is not present. For arbitrary f and l_S, the above is intractable. For submodular $f(A)$, it depends on the choice of $l_S(A)$. Optimization issues aside, one benefit of LAI is that it allows us to train the DSF on a given set of items and test on a completely different set of items.

One of the authors' experiments is of interest to us, because it works on summarizing collections of images using a "visual" bag of words feature model. In our case, we want to learn summarization on a trainset of documents with human summaries given, and use it to predict summaries of test set documents. In the authors' case, they use a simple, single-layer DSF that takes the form -:

$$f(A) = \hat{\sigma}(\sum_{u \in \mathcal{U}} w_u \sqrt{m_u(A)}) \tag{8}$$

where $\hat{\sigma}$ is the normalized sigmoid function (as it is concave in the non-negative region), w_u is a trainable weight for feature u and $m_u(A)$ is a modular function for feature u. This formulation gives a submodular function as it is concave over weighted sum of concave over modular. We choose a similar DSF, but without the modular function.

5 Experiments and Results

As the original paper by Bilmes et al.[4] used document understanding conference (DUC) (http://duc.nist.org) dataset, we wanted to try this technique on other dataset and so we selected WikiHow[2] dataset. It turned out that the DUC dataset was login-walled, so we could not access it. The WikiHow dataset consists of several documents demarcated by human-provided summaries (may or may not be more than one) and the actual text of the document, so this fit our needs. We used 1000 of the articles for evaluating the performance of this method.

Each article was pre-processed by segmenting the sentences and using the Porter stemmer. We first calculated term frequency- inverse document frequency (TF-IDF) for every sentence in each document and then used cosine similarity to find the w_{ij}.

For Evaluation, we used a popular metric for document summarization known as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) Score. Where ROUGE-1 Recall, Precision and F1-Score is defined as:

$$\begin{aligned}
\text{ROUGE-1, Recall} &= \frac{\text{Number of Word matches}}{\text{Number words in reference}} \\
\text{ROUGE-1, Precision} &= \frac{\text{Number of Word matches}}{\text{Number of words in Summary}} \\
\text{ROUGE-1, F-1 Score} &= 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\end{aligned} \tag{9}$$

Similarly ROUGE-L Recall is defined as the ratio between longest common subsequence between generated summary and reference summary and Number of words in reference. ROUGE-L Precision is defined as the ratio between longest common subsequence between generated summary and reference summary and Number of words in the summary.

5.1 Submodular Summarization on WikiHow

We implemented the submodular functions given in the paper by Bilmes et al. on the Wikihow dataset. The formal detailed equtaion for coverage and diversity function is given in Equation 4 and 6 respectively. First we found the summarization using Only coverage function ($\mathcal{L}(S)$, only diversity function $\mathcal{R}S$ and then both functions combined as:

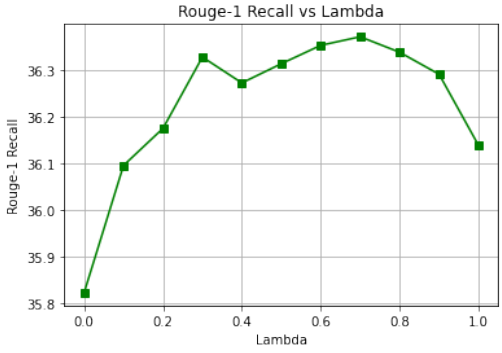
$$\mathcal{F}(S) = \lambda \mathcal{L}(S) + (1 - \lambda)\mathcal{R}(S) \tag{10}$$

That is a trade-off between Coverage and diversity functions. Also for other parameters, we used $K = 0.2 * N$ that is the total number of clusters in finding the diversity function, $a = 6$ that is used to calculate the α in Equation 4 as $\alpha = a/N$. We also experimented with varying the trade-off parameter λ from 0 to 1. Google Colab Notebook [1] for the above experiment is also made available.

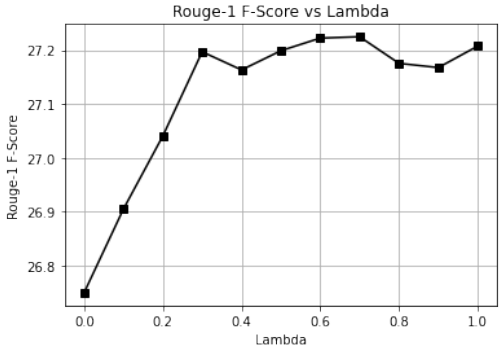
Result for the different Experiment is shown in Table- 5.1. The plots of ROUGE-1 and ROUGE-L Score are displayed in Image 1 and 2 respectively.

Table-5.1: ROUGE Score with different submodular functions

Experiment	ROUGE-1 Recall	ROUGE-1 F1- Score	ROUGE-l Recall	ROUGE-L F-1 Score
$\mathcal{L}(S)$	36.14	27.20	21.83	16.24
$\mathcal{R}(S)$	35.82	26.75	21.72	16.05
$0.15\mathcal{L}(S) + 0.85\mathcal{R}(S)$	34.67	27.67	20.81	16.41

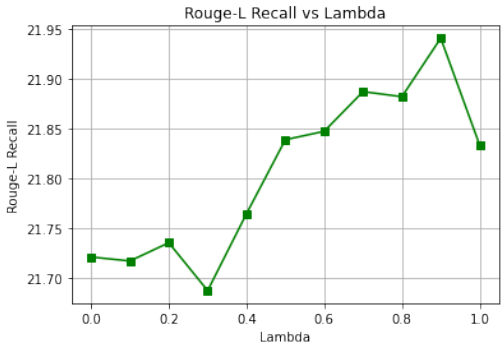


(a)

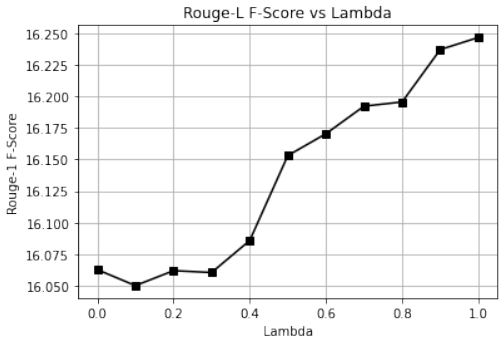


(b)

Figure 1: Plot for ROUGE-1 (a) Recall and (b) F-1 Score vs λ



(a)



(b)

Figure 2: Plot for ROUGE-L (a) Recall and (b) F-1 Score vs λ

[1] The code for Submodular Summarization on WikiHow can be found at https://tinyurl.com/bdcwdw9n.

From the obtained results we find that the ROUGE F-1 Scores was highest with the use of both the function as the generated summary will be more covering and diverse. And as for the optimal value of concerned, For higher F-1 score lambda is giving better results near a value of 1.

5.2 Deep Submodular Function

We defined a simple one-layer DSF as explained in the previous section of this report. The WikiHow dataset has train and test split text files, which we used to collect documents. Initially, we picked the entire set of training documents specified in "train_all.txt" and built a gigantic (sparse) TF-IDF matrix, where the rows were document IDs and the columns were words in the vocabulary.

This turned out to be too big to fit in GPU memory, so we took on a different approach. We pruned the trainset to take 5000 training documents and the test set to take 403 test documents. We then used pre-trained word embeddings from the GloVe embedding repository to obtain 100-dimensional feature vectors for each word. Words were obtained from sentences or text with the help of Spacy's English tokenizers. For a given human summary, each alphanumeric word in the summary has its embedding generated and is stacked into a single tensor, which is fed into the neural model as input. The pytorch library (and its sub-libraries) was used to build the neural architecture.

Our neural model architecture consists of an overall "container" module, a DSF layer module and a positive linear layer module. The positive linear layer is required as per the non-negative weights constraint specified in the paper. During training, we find the output of the model on the human summary, and then, for each word in this summary, we generate 10 replacement candidate summaries by replacing the word with a word from the vocabulary (the "ground set"). The idea is to maximize the margin between the desired human summary and the generated candidate summaries.

Training was done using the standard Adam optimizer with a learning rate set to $1e-3$. We chose the leaky ReLU loss as using standard ReLU we ended up with very negative loss values which were clamped to zero, precluding the model from actually learning anything. The idea was that leaky ReLU would allow leakage of some negative values which would be corrected during training.

In the interest of time, we trained the model for only ten epochs. Even with the Leaky ReLU, we ran into pressing issues of zero-loss during training which we were not able to fix, perhaps due to some bug where the model was not learning properly. We also tried to add a regularizer term *delta* to the hinge loss to enforce a margin, but could not explore it further due to lack of time.

We believe that the model could be improved with batching and more careful implementation of differentiable operations. Had we been more successful, the goal was to use greedy inference on the learned submodular function to produce summaries of a given budget (or length of text). We provide the code [2] along with the report for perusal.

References

[1] Brian W Dolhansky and Jeff A Bilmes. "Deep submodular functions: Definitions and learning". In: *Advances in Neural Information Processing Systems* 29 (2016).

[2] Mahnaz Koupaee and William Yang Wang. "Wikihow: A large scale text summarization dataset". In: *arXiv preprint arXiv:1810.09305* (2018).

[2] The code for DSF can be found at https://tinyurl.com/dsf-colab.

[3] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. "The global k-means clustering algorithm". In: *Pattern recognition* 36.2 (2003), pp. 451–461.

[4] Hui Lin and Jeff Bilmes. "A class of submodular functions for document summarization". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*. 2011, pp. 510–520.

[5] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. "An analysis of approximations for maximizing submodular set functions—I". In: *Mathematical programming* 14.1 (1978), pp. 265–294.