DIFFERENTIABLE GREEDY NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimal selection of a subset of items from a given set is a hard problem that requires combinatorial optimization. In this paper, we propose a subset selection algorithm that is trainable with gradient based methods yet achieves near optimal performance via submodular optimization. We focus on the task of identifying a relevant set of sentences for claim verification in the context of the FEVER task. Conventional methods for this task look at sentences on their individual merit and thus do not optimize the informativeness of sentences as a set. We show that our proposed method which builds on the idea of unfolding a greedy algorithm into a computational graph allows both interpretability and gradient based training. The proposed differentiable greedy network (DGN) outperforms discrete optimization algorithms as well as other baseline methods in terms of precision and recall.

1 INTRODUCTION

In this paper, we develop a subset selection algorithm that is differentiable and discrete, which can be trained on supervised data and can model complex dependencies between elements in a straightforward and comprehensible way. This is of particular interest in natural language processing tasks such as fact extraction, fact verification, and question answering where the proposed optimization scheme can be used for evidence retrieval.

Conventional evidence retrieval methods that look at lexical or semantic similarity typically treat sentences or documents independently, potentially missing dependencies between them and therefore select redundant evidence. One way to address this shortcoming is by adding a diversity promoting submodular objective function (Tschiatschek et al., 2014; Lin & Bilmes, 2011; Lovász, 1983; Fujishige, 2005; Krause, 2010). Submodularity is a property of set functions that can be expressed by the notion of diminishing returns that allows near-optimal solutions to be found in polynomial time for NP-hard problems.

A submodular set function is a function that maps sets to scalar values and has the property that the incremental value of the function computed with an additional element to an input set never increases as the input set grows. Submodular functions are defined by this natural diminishing returns property, which makes them well suited for tasks such as claim verification. With respect to a claim, the amount of relevant information in a set of sentences has diminishing returns as the set grows, meaning that the amount of additional information in an additional piece of evidence shrinks as the set of selected evidence grows. Thus, any relevancy-measuring function that is learned from data would potentially benefit from a diminishing returns constraint as it would discount redundancy in favor of diverse but relevant evidence. Claim verification often requires complicated induction from multiple sentences, so promoting diversity among selected sentences is important to capture all facets of the claim. The resulting submodular optimization model can then handle dependencies between sentences and features, and despite making the sentence selection problem more difficult computationally, a near-optimal solution can be found efficiently using a simple forward greedy algorithm.

The main contribution of this paper is a new optimization scheme which integrates continuous gradient-based and discrete submodular frameworks derived by unfolding a greedy optimization algorithm: the Differentiable Greedy Network (DGN). By unfolding a greedy algorithm into a computational graph, we can combine the advantages in interpretability and representation learning. Deep unfolding is a technique that transforms inference algorithms into computational graphs, thereby allowing the original model parameters to be trained discriminatively on labeled data while still exactly corresponding to the original model parameters (Hershey et al., 2014). We show that making a greedy algorithm differentiable and adding trainable parameters leads to promising improvements in recall@k of 10%-18% and precision@k of 5%-27% for a sentence selection task, where $k = 1, 3, 5, 7$ is the number of selected evidence sentences, on the Fact Extraction and Verification (FEVER) dataset (Thorne et al., 2018) and, with fewer parameters, performs very similarly to a conventional deep network. As the DGN is bootstrapping a greedy algorithm, it can be easily extended to work on other information retrieval tasks such as question answering as well as other problems that rely on greedy approaches. While more sophisticated neural architectures can deliver better performance, we focus on showing the power of our new optimization scheme on a simpler model.

In Section 2, we discuss related work in the domains of information retrieval, submodularity, and deep unfolding. In Section 3, we define submodularity and present the proposed Differentiable Greedy Network (DGN). Section 4 contains experiments and results for baseline models and DGN applied to sentence selection for the FEVER dataset as well as an ablation study. We draw conclusions in Section 5. Also, the attached Appendix 6 contains an additional example demonstrating the utility of promoting diversity.

2 Related Work

Evidence retrieval is a key part of claim verification or question answering which aims to provide reasoning and a measure of interpretability (Lei et al., 2016). The FEVER dataset (Thorne et al., 2018) comes with a baseline evidence retrieval system, which returns the top k sentences based on cosine similarity of term-frequency inverse-document-frequency (TF-IDF) features between the claim and candidate sentences. Models (Vaswani et al., 2017; Huang et al., 2018; Seo et al., 2016) that use attention on top of recurrent or convolutional neural networks can be adopted for this problem. However, all these types of models often focus only on learning similarities between claims and candidate evidence sentences, but neglect to model dependencies between the candidate sentences themselves, which is an advantage of a model that uses submodular functions which we focus on.

Central to our work here is the idea of deep unfolding Hershey et al. (2014) a technique that fills the continuum between the two extremes of generative models and deep learning models and combines the advantages of deep learning (discriminative training through backpropogation, ability to leverage large datasets) and generative models (ability to leverage domain knowledge, optimization bounds, fast training on smaller datasets).

Deep unfolding (Hershey et al., 2014) has demonstrated a principled way to derive novel network architectures that are interpretable as inference algorithms by turning iterations of the inference algorithms into layers of a network. Previous work has unfolded Gaussian mixture models for multichannel source separation, the sequential iterative shrinkage and thresholding algorithm (SISTA) for sparse coding, and nonnegative matrix factorization (NMF) (Gregor & LeCun, 2010; Chen et al., 2015; Hershey et al., 2014; Wisdom et al., 2016; 2017; Le Roux et al., 2015). Particularly for sparse models, unfolding has only been applied to inference algorithms that use regularization approaches to promote sparsity.

Tschiatschek et al. (2018) considered subset selection through greedy maximization as well. Since output of their proposed method is differentiable as they can be interpreted as a distribution over sets, the proposed method can be combined with gradient based learning. However, their proposed method is different from us as it relies on stochastic submodular optimization algorithms. Moreover, we focus on sentence selection which comes with its own complexities. Others have proposed related versions of learning through discrete functions, including deep submodular functions (DSF) (Bilmes & Bai, 2017), submodular scoring functions (Sipos et al., 2012), iterative hard thresholding (Xin et al., 2016), and the argmax function (Mensch & Blondel, 2018). The common thread of these related works is relaxation of the discrete functions. Our proposed network is distinct in that it learns the parameters of submodular function and feature encoder through the greedy optimization using a relaxed argmax function at training time. At test time, the model uses the argmax, so when it runs, it is still solving the submodular function maximization problem.

Algorithm 1 FORWARD GREEDY (V, k, f)

```
1: $\hat{A} \leftarrow \emptyset$
2: while $|\hat{A}| < k$ do
3:     $v^* \leftarrow \text{argmax}_{v \in V \setminus \hat{A}} f(v|\hat{A})$
4:     $\hat{A} \leftarrow \hat{A} \cup \{v^*\}$
5: end while
6: return $A$
```

3 DIFFERENTIABLE GREEDY NETWORKS

We start with a brief review of submodularity as it applies to our development of Differentiable Greedy Networks. The innovation of this paper stems from making greedy optimization of a submodular set function differentiable, but at test time, the performance guarantees are determined by the submodularity of the function, the constraints of the optimization problem, and the optimization algorithm used.

3.1 SUBMODULARITY

Submodularity is a property that describes set functions analogous to how convexity describes functions in a continuous space. Rather than exhaustively searching over all combinations of subsets, submodular functions provide a fast and tractable framework to compute a near optimal solution (Lovász, 1983; Fujishige, 2005; Krause, 2010).

Let the set of available objects, known as the ground set, be denoted as V. Submodularity can be expressed via the notion of diminishing returns, i.e., the incremental gain of the objective diminishes as the context grows. If we define the incremental gain of adding element v to A as $f(v|A) = f(A \cup \{v\}) - f(A)$, then a submodular function f is defined as satisfying

$$f(v|A) \geq f(v|B) \quad \forall A \subseteq B \subset V,\ v \notin B. \tag{1}$$

Submodular functions can be both maximized and minimized. Submodular functions that are maximized tend to exhibit concave behavior such as probabilistic coverage, sums of concave composed with monotone modular functions (SCMM), and facility location. Constrained monotone submodular function maximization algorithms include the forward greedy algorithm (Nemhauser et al., 1978; Fisher et al., 1978) and continuous greedy algorithm (Vondrák, 2008). Likewise, constrained non-monotone submodular function maximization algorithms include local search algorithm (Lee et al., 2009), greedy-based algorithms (Gupta et al., 2010), and other algorithms that use some combination of these (Mirzasoleiman et al., 2016). In this paper, we focus on the forward greedy algorithm. Greedy optimization algorithms provide a natural framework for subset selection problems like retrieval, summarization, sparse approximation, feature selection, and dictionary selection. One can derive lower bounds if the objective function in a given subset selection problem is submodular (Nemhauser et al., 1978), and even for some non-submodular cases (Cevher & Krause, 2011; Das & Kempe, 2011; Powers et al., 2016).

For retrieval tasks like evidence extraction, submodular function optimization allows for dependence between potential pieces of evidence. For example, consider a problem where given a claim, the goal is to select sentences that help verify or refute the claim. A common simplifying assumption made in selection models is that the sentences are independent of each other, thereby ensuring the selection has linear time complexity in terms of the number of sentences. While this independence assumption doesn't necessarily adversely affect the relevancy of the selected sentences, it might lead to redundancy in the selected sentences.

The submodular function $f(\cdot)$ that we optimize is an SCMM given as follows:

$$f_\alpha(A) = \sum_{u \in U} \alpha^u \log(1 + \sum_{a \in A} h_a^u), \tag{2}$$

where $\alpha^u \in \mathbb{R}_+$, $\forall u \in U$ are non-negative trainable parameters at index u of the feature dimension that allow the functions to be tuned on data, and $h_a^u \in \mathbb{R}_+$ are the output features from the encoding layer for sentences A and feature indices U. Specifically, they allow the network to identify the relative importance of different features U. For the sentence selection task that we test the DGN on in Section 4, these features correspond to the semantic similarities of a claim and potential evidence.

Given the monotone submodular objective function, the optimization problem is to find the optimal subset A^* that maximizes $f_\alpha(\cdot)$:

$$A^* \in \underset{A \in V, |A| \leq k}{\operatorname{argmax}} \; f_\alpha(A). \tag{3}$$

Even though Problem (3) is NP-hard, we chose the objective function to be submodular, so a near optimal solution can be found in polynomial time (Nemhauser et al., 1978). Specifically, we can use the forward greedy algorithm, detailed in Algorithm 1, to find a subset $\hat{A}$, such that $f_\alpha(\hat{A}) \geq (1 - 1/e) f_\alpha(A^*)$ where $(1 - 1/e) \approx 0.63$. This guarantee ensures that any estimated solution set is no more than a constant factor worse than the true optimal set in terms of the objective function, and in practice the greedy algorithm often performs well above this tight lower bound.

3.2 Differentiable Greedy Networks

Now we present the Differentiable Greedy Network (DGN), an unfolded discrete optimization algorithm, which will allow for discriminative training of parameters while retaining the original guarantees of the forward greedy algorithm shown in Algorithm 1.

The most important change to make the network differentiable is to the argmax function in line 3 of Algorithm 1. During training, the argmax cannot be used directly because its gradients are not well-defined, so we approximate the argmax with softmax and use a temperature parameter τ to scale how close it approximates the argmax (Jang et al., 2017). This is an important parameter during training, as the temperature greatly affects the gradient flow through the network. As is discussed in Section 4, setting the temperature too low results in large gradients with high variance which makes training unstable and limits the gains in precision-recall. The temperature should not be too high either as the output starts to look more uniform and less like the argmax.

Figure 1 depicts the DGN as a computational graph. It is built out from Algorithm 1, where lines 2-4 comprise the greedy layers. The overall structure of the network on the left shows that the input features, X, are run through a linear encoding layer follows by ReLU as an activation function. We choose the ReLU function specifically because the submodularity of the SCMM function requires non-negativity and because of its stability during training (Nair & Hinton, 2010). The resulting non-negative features, H, are then run through successive greedy layers along with the state vector which encodes the progression of sentence selections. The right diagram details a greedy layer. In the left branch, for greedy iteration i, the network enumerates the possible next states, $\{\mathbf{s}^i \cup \{v\} : \forall v \in V \backslash \mathbf{s}^i\}$. The network then evaluates these potential next states in the context of the current state $\mathbf{s}^i$ and finds the sentence that provides the biggest increase in terms of $f_\alpha(\cdot)$, which is then added to the current state to form the new state $\mathbf{s}^{i+1}$.

In some respects, the overall network looks conventional, especially the linear plus ReLU encoding layer. The connections between H and the greedy layers are very reminiscent of a recurrent network where an input or hidden state is fed into layers with shared weights. The greedy layers, though, deviate significantly from conventional network architectures. Functionally, the greedy layers serve a similar purpose as an attention mechanism, but where attention would assign importance or relevance of a sentence to a claim or query, the submodular function in the greedy layer can be designed not only to measure the relevance of a sentence, but also explicitly model dependencies like redundancy between sentences and does so in a straightforward and interpretable manner. Examples of how the DGN models the sentence dependencies can be found in Section 4. The intuition behind how the DGN relates sentences together goes back to the submodularity of the objective function within the network. For the evidence extraction task described in Section 4, the input features are effectively similarity values between the claim and sentences across different feature indices. By applying a compressive concave function for a specific feature index u aggregated across a set of sentences A and summing over the feature indices $\forall u \in U$, a set of sentences that are similar to the claim but at different indices in the feature dimension would be preferred to a set of sentences that are equally similar but at the same indices.

4 Experiments

We apply the DGN to the Fact Extraction and Verification (FEVER) dataset (Thorne et al., 2018). FEVER data consists of claims, classification of the claims as Supported, Refuted, or NotEnough-

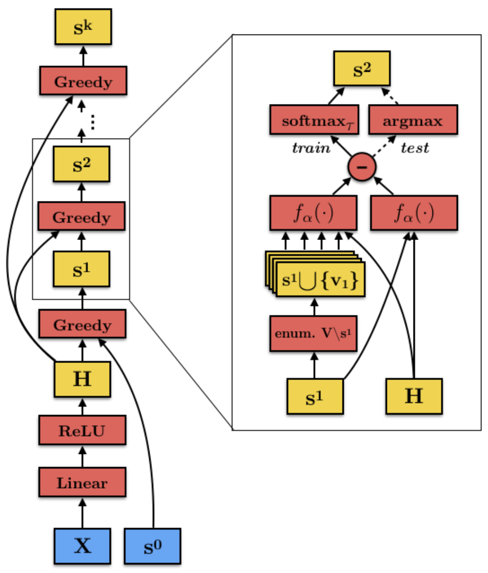

Figure 1: The Differentiable Greedy Network (DGN). The overall structure of the network on the left shows that the input features X are run through a linear encoding layer and a ReLU activation function. The resulting non-negative features H are then run through successive greedy layers along with the state vector which encodes the progression of sentence selections. The right diagram details a greedy layer. In the left branch, for greedy iteration i, the network enumerates the possible next states, $\{s^i \cup \{v\} : \forall v \in V \backslash s^i\}$. The network then evaluates these potential next states in the context of the current state s^i and finds the sentence that provides the biggest increase in terms of $f_\alpha(\cdot)$, which is then added to the current state to form the new state s^{i+1}. During training, the selection is done through a softmax with temperature τ to differentiably approximate the argmax, and at test time we use the argmax to ensure the submodular guarantee of the model.

Info (NEI), and evidence that the classification is based on. The claims and evidence come from Wikipedia. The baseline system for FEVER has two main steps: evidence retrieval and recognizing textual entailment (RTE). The dataset has 109,810 verifiable claims in the training set, 13,332 in the validation set and 6,666 in the test set. The baseline system selects k sentences from a set of retrieved documents that have the highest TF-IDF cosine similarity to a claim.The FEVER scoring function computes the precision, recall, and F1 score for the retrieved sentences with respect to the labeled evidence sentences in verifiable claims. The evidence sentences are then run through the a baseline RTE network along with the claim to classify the claim as Supported, Refuted, or NEI. Notably, the best RTE baseline system drops from 88.00% accuracy to 50.37% when going from oracle evidence retrieval to the baseline retrieval system demonstrating that there is significant room to improve the evidence retrieval module. Also, the attached Appendix 6 contains an additional example demonstrating the utility of promoting diversity.

We narrow our focus to sentence retrieval given oracle documents. Most claims have one or two evidence sentences, but the distribution is fairly heavy-tailed. Based on the distribution, we set the max number of greedy iterations to $k = 7$ in our experiments, which covers 93% of the claims and provides a reasonable balance between coverage, precision, and computational complexity. Claim verification often requires complicated induction from multiple sentences, so ideally, a more complicated model would be able to model the interactions between sentences. The next step is to introduce the concept of redundancy to the model by substituting in a diversity-promoting value function, SCMM, for the baseline value function, cosine similarity.

We use mean-pooled FastText word embeddings of dimension F for each claim and D potential evidence sentences and element-wise multiply the claim feature vector with the sentences (Bojanowski et al., 2016). Thus, the inputs $X \in \mathbb{R}^{\mathbb{F} \times \mathbb{D}}$ signify similarity scores between the claim and sentences for each of the feature dimensions. Connecting back to Equations (2) and (3), $F = |U|$ and $D = |V|$. Compared to TF-IDF features, the FastText word embeddings encode semantic structures of words that are more meaningful than simple lexical matching statistics and motivate the idea of promoting diversity. Consider a claim and three sentences, where the goal is to select the best two sentences as evidence. Using the SCMM in Equation (2), the forward greedy algorithm in Algorithm 1 would

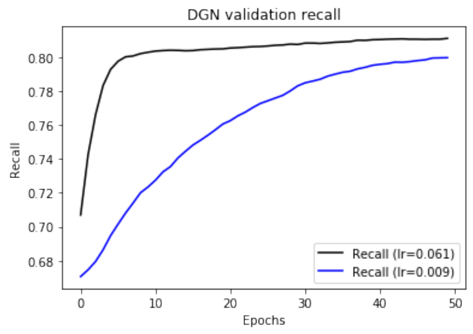

Figure 2: Recall@7 on the validation set. The DGN seems to fit very quickly to the full set in terms of cross entropy, but the recall improves consistently. Decreasing the learning rate results in a smoother and consistently decreasing loss, but lower recall.

first select the sentence with the highest similarity to the sentence. If the other two sentences have are approximately equally similar to the claim, the SCMM will prefer the one that is similar across different feature dimensions than the chosen sentence. Therefore, this submodular framework will encourage semantic diversity among relevant sentences to a claim.

The DGN encoder projects input features X to a matrix $H \in \mathbb{R}_+^{F' \times D}$ by passing the features through two linear layers with ReLU non-linearity. The state vector **s** is of dimension D. We perform random hyperparameter searches over learning rate, encoder output dimension F', and softmax temperature τ. The training loss is accumulated layer-wise cross entropy from output of the greedy layers(eq 4). We choose to compute the loss at each layer, rather than at the end to help better correct decisions made at the individual layers. The loss is computed as follows.

Let $\boldsymbol{s}^k \in R^D$ denotes the state at layer k, which assigns a normalized score to each of the sentences. Labels are given by $L \subset \{1, ..., D\}$. The Cross Entropy loss is given as:

$$Loss_{CE}(L, \boldsymbol{s}^1, .., \boldsymbol{s}^k) = \sum_{j=1}^{min(K,|L|)} \sum_{i=1}^{D} \Big[\mathbb{I}_{L_j = i} \cdot \log \boldsymbol{s}_i^j + \mathbb{I}_{L_j \neq i} \cdot \log(1 - \boldsymbol{s}_i^j) \Big], \tag{4}$$

where $\mathbb{I}_A$ is the indicator function for event A, and K indicates the number of greedy layers(7 in this paper).

We compare the DGN to several baselines, each emphasizing a particular comparison for the DGN: a feedforward greedy algorithm to test the importance of learning better weights for the submodular objective function, a shallow feedforward network to test the impact of modeling dependence between sentences that we call the Encoder model, and a deeper feedforward network model DeepEncoder. Both the Encoder and DeepEncoder models consist of dense linear layers with ReLU activation functions with softmax output layers.They are trained with a binary cross entropy loss function. We use Adam as the network optimizer with default parameters except for learning rate (Kingma & Ba, 2014). The models are implemented in PyTorch (Paszke et al., 2017).

The results in Table 1 show recall, precision and F1 scores for the tested models. The untrained baselines are a recreation of the FEVER system that selects the top k sentences based on cosine similarity of sentence embeddings and a greedy algorithm that maximizes the submodular SCMM function in Equation (2). We also perform an ablation study to measure the effect of the greedy layers by removing them and measuring the performance of Encoder and also test against multi-layer feedforward baseline (DeepEncoder). The Encoder model has 172K trainable parameters. The DeepEncoder has 915K parameters. Each block of rows of Table 1 show recall, precision, and F1 when the models select $k = 7, 5, 3, 1$ sentences as evidence. The DGN outperforms the untrained baselines, improving recall@7 by 10% while also improving precision. When selecting fewer sentences, $k = 5, 3, 1$, the recall improves 14-18% while producing increasing gains in precision 5%, 10%, and 27%, respectively.

While the performance of the Encoder and DeepEncoder as shown in Tables 2 and 1 are very close to that of the DGN, the DGN maintains several advantages. First, the DGN is much more robust to

Table 1: Evidence retrieval results. Each block of rows show recall, precision, and F1 when the models select $k = 7, 5, 3, 1$ sentences as evidence. The DGN significantly outperforms the untrained baselines. When selecting fewer sentences, $k = 5, 3, 1$, the DGN produces increasing gains in recall, precision, and F1. The DGN slightly, but consistently outperforms the Encoder.

Model	Recall@7	Precision@7	F1@7
Top-k	0.704	0.179	0.285
Greedy	0.706	0.179	0.286
Encoder	0.809	0.205	0.327
DGN	**0.811**	**0.206**	**0.328**
Model	Recall@5	Precision@5	F1@5
Top-k	0.598	0.205	0.305
Greedy	0.600	0.205	0.306
Encoder	0.734	0.251	0.374
DGN	**0.738**	**0.253**	**0.377**
Model	Recall@3	Precision@3	F1@3
Top-k	0.438	0.244	0.313
Greedy	0.439	0.245	0.314
Encoder	**0.615**	**0.342**	**0.440**
DGN	**0.615**	**0.342**	**0.440**
Model	Recall@1	Precision@1	F1@1
Top-k	0.194	0.322	0.242
Greedy	0.194	0.322	0.242
Encoder	0.353	0.588	0.441
DGN	**0.356**	**0.593**	**0.445**

Table 2: Evidence retrieval results for the DeepEncoder model and the DGN. Each row shows recall, precision, and F1 when the models select $k = 7$ sentences as evidence. The DGN achieves comparable performance of the DeepEncoder models with a fraction of the parameters in the best performing DeepEncoder.

Model	# Trainable Parameters	Recall@7	Precision@7	F1@7
DeepEncoder	**915K**	0.815	0.207	0.330
DGN	**167K**	0.811	0.206	0.328

class imbalances. In order to get the Encoder and DeepEncoder models to work, the positive samples (evidence sentences) needed to be weighted more heavily in the cross entropy loss. Moreover, the encoder models were very sensitive to this value–set too low the model rejected every sentence, and too high the model selects every sentence. The greedy algorithm regularizes the network to select k sentences, though, and therefore is insensitive to the class imbalance. Second, since the k greedy layers are performing k iterations of submodular function optimization, the DGN explicitly and transparently models dependencies between functions given the right choice of submodular function.

To illustrate how the dependencies encoded by the submodular function help, consider the following claim, which has two distinct facts that need to be verified: "Jarhead, a 2005 American biographical war drama, was directed by the award-winning auteur Sam Mendes." The first half of the claim asserts that Sam Mendes directed "Jarhead," but looking at the labeled evidence the algorithm also needs to verify that he has won an award in the arts (presumably film). The DGN does just this in the first two greedy layers picking up on both facets of the claim with sentences a, $f_\alpha(\{a\}) = 14.48$, and b, $f_\alpha(\{b\}) = 12.86$: "He is best known for directing the drama film American Beauty (1999), which earned him the Academy Award and Golden Globe Award for Best Director, the crime

Table 3: Effects of various optimization modifications: softmax temperature annealing, sentence order shuffling

Greedy optimization modification	Recall@7	Precision@7	F1@7
Temperature annealing	0.756	0.192	0.306
No temperature annealing	0.763	0.194	0.309
Sentence shuffling	0.802	0.203	0.324
No sentence shuffling	0.811	0.206	0.328

film Road to Perdition (2002), and the James Bond films Skyfall (2012) and Spectre (2015)," and "Jarhead is a 2005 American biographical war drama film based on U.S. Marine Anthony Swofford's 2003 memoir of the same name, directed by Sam Mendes, starring Jake Gyllenhaal as Swofford with Jamie Foxx, Peter Sarsgaard and Chris Cooper." On the other hand, the DeepEncoder selects sentence b, verifying that Jarhead was directed by Sam Mendes, but the next highest rated sentence is sentence c: "Samuel Alexander Mendes, (born 1 August 1965) is an English stage and film director." The DGN ranked both sentences b and c fairly highly, $f_\alpha(\{c\}) = 12.26$, but determined that sentence c was more redundant with sentence b than a was, $f_\alpha(\{a, b\}) - f_\alpha(\{a\}) = 8.00$ compared to $f_\alpha(\{a, c\}) - f_\alpha(\{a\}) = 7.25$, so the score for c dropped more than for b.

The DGN also had some interesting failure cases. A potential downside to promoting diversity in the selected sentences is if the DGN selects the wrong sentence when the actual evidence sentences are ones that the submodular function deems redundant. For the claim "Phoenix, Arizona is the capital of the Atlantic Ocean," which is easily refuted, the labelled evidence sentence a is "Phoenix is the capital and most populous city of the U.S. state of Arizona." The DGN rates this sentence highly in the first greedy layer, $f_\alpha(\{a\}) = 13.23$, but not as highly as sentence b, $f_\alpha(\{b\}) = 15.99$: "In addition , Phoenix is the seat of Maricopa County and, at 517.9 square miles (1,341 km^2), it is the largest city in the state, more than twice the size of Tucson and one of the largest cities in the United States." In the second layer, $f_\alpha(\{a, b\}) - f_\alpha(\{b\}) = 7.5$, which again lowers the score enough that the algorithm selects sentence c: "Despite this, its canal system led to a thriving farming community, many of the original crops remaining important parts of the Phoenix economy for decades, such as alfalfa, cotton, citrus, and hay (which was important for the cattle industry)." The incremental gain of sentence a continues to drop enough in subsequent layers to prevent selection as the DGN selects other sentence. Intuitively, this type of error seems especially problematic for claims that need to be refuted, specifically when the fallacy in the claim is seemingly unrelated to the evidence needed to refute it. There is very little connecting Arizona, for which Phoenix is the capital, and the Atlantic Ocean. In this case, we might prefer the algorithm to select more redundant sentences, because the extra information needed to refute it, Arizona, is not closely related to the correspondingly important part of the claim, Atlantic Ocean.

Figure 2 shows and recall on the validation set for the same two models. Both models fit very quickly to the full training and validation sets in terms of cross entropy, but the recall improves consistently. By lowering the learning rate, both the mean layer-wise cross entropy and recall smoothly improve, whereas raising the learning rate achieves a lower cross entropy after several epochs before increasing steadily as well as a higher and monotonically increasing recall. We believe this to be an artifact of the loss due to the greedy layers restriction of selecting a single sentence each. Looking back at Equation 4, at each outer sum we calculate the cross entropy assuming there is only one valid sentence. If there are more than one valid sentences, and the state, $\boldsymbol{s}^i$, assigns large scores to those valid sentences, all except one are penalized by CE loss at that step. The ones penalized will be rewarded when calculating the loss at the later steps. This potentially can increase the loss when the model performance is improving. We observed the summed loss to be an acceptable trade-off as the goal is to maximize recall. Another advantage of the summed loss is that it reduces the issue of vanishing gradients as seen in recurrent networks. It will add direct gradient flow to the earlier layers of the unfolded network in the backward path. In order to test how fixing the sentence order affects the model performance, we shuffled the sentences every epoch, which resulted in slightly worse recall, precision, and F1 as can be seen in Table 3.

Training the DGN was initially challenging. The choice of softmax temperature proved to be one of the most important hyperparameter settings. Our first thought was to make the temperature very low, $\tau = 0.01$, so that it was a close approximation of the argmax. This resulted in both vanishing and exploding gradients, because the loss was computing log of ones and zeros. To solve this, we experimented with two approaches. In the first approach, we set the temperature to be a number in the range (0.5,5) and is kept unchanged during the experiments. In the second approach, we apply temperature annealing, where in the beginning the temperature is set in the range as the first approach, and is annealed by 10% every epoch. Our experiments showed that both approaches produce similar results, which can be seen in Table 3. To make the training simpler, we chose the first approach. The experiments showed that temperatures in (3,6) range produce the best results, which means the softmax is making rather soft selections at each greedy iteration during training. Consistent with previous work, the higher temperature softmax less closely approximates the argmax during training, but does have the positive effect of regularizing the gradient.

5 Conclusion

In this paper, we have shown that unfolding a greedy algorithm into a computational graph, allowing us to retain the interpretability and unsupervised initialization of a conventional greedy sentence selection approach while benefiting from supervised learning techniques. The proposed differentiable greedy network (DGN) outperforms conventional discrete optimization algorithms in terms of both recall and precision. Furthermore, as sentence retrieval is often part of a larger pipeline as in the FEVER shared task, using a differentiable greedy network serves as a step towards an end-end trainable system.

References

Jeffrey A. Bilmes and Wenruo Bai. Deep submodular functions. *CoRR*, abs/1701.08939, 2017.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

V. Cevher and A. Krause. Greedy Dictionary Selection for Sparse Representation. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):979–988, September 2011. ISSN 1932-4553. doi: 10.1109/JSTSP.2011.2161862.

Jianshu Chen, Ji He, Yelong Shen, Lin Xiao, Xiaodong He, Jianfeng Gao, Xinying Song, and Li Deng. End-to-end Learning of LDA by Mirror-Descent Back Propagation over a Deep Architecture. *arXiv:1508.03398 [cs]*, August 2015.

Abhimanyu Das and David Kempe. Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection. *arXiv:1102.3975 [cs, stat]*, February 2011. URL `http://arxiv.org/abs/1102.3975`. arXiv: 1102.3975.

M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. An analysis of approximations for maximizing submodular set functions-ii. In M.L. Balinski and A.J. Hoffman (eds.), *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pp. 73–87. Springer Berlin Heidelberg, 1978.

S. Fujishige. *Submodular Functions and Optimization*. Elsevier, 2005.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proc. ICML*, pp. 399–406, Haifa, Israel, 2010.

A. Gupta, A. Roth, G. Schoenebeck, and K. Talwar. *Constrained Non-monotone Submodular Maximization: Offline and Secretary Algorithms*, pp. 246–257. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-17572-5. doi: 10.1007/978-3-642-17572-5_20. URL `http://dx.doi.org/10.1007/978-3-642-17572-5_20`.

John R. Hershey, Jonathan Le Roux, and Felix Weninger. Deep Unfolding: Model-Based Inspiration of Novel Deep Architectures. *arXiv:1409.2574*, September 2014.

Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *International Conference on Learning Representations*, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. 2017. URL `https://arxiv.org/abs/1611.01144`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

A. Krause. SFO: A toolbox for submodular function optimization. *J. Mach. Learn. Res.*, 11:1141–1144, March 2010.

Jonathan Le Roux, F. J. Weninger, and John R. Hershey. Sparse NMF - half-baked or well done? *MERL Technical Report*, 2015. URL `http://www.merl.com/publications/docs/TR2015-023.pdf`.

J. Lee, M. Sviridenko, and J. Vondrák. *Submodular Maximization over Multiple Matroids via Generalized Exchange Properties*, pp. 244–257. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-03685-9. doi: 10.1007/978-3-642-03685-9_19. URL `http://dx.doi.org/10.1007/978-3-642-03685-9_19`.

Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. Rationalizing neural predictions. *CoRR*, abs/1606.04155, 2016. URL `http://arxiv.org/abs/1606.04155`.

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 - Volume 1*, HLT '11, pp. 510–520. Association for Computational Linguistics, 2011. ISBN 978-1-932432-87-9.

L. Lovász. Submodular functions and convexity. In Achim Bachem, Bernhard Korte, and Martin Grtschel (eds.), *Mathematical Programming The State of the Art*, pp. 235–257. Springer Berlin Heidelberg, 1983.

Arthur Mensch and Mathieu Blondel. Differentiable dynamic programming for structured prediction and attention. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 3459–3468, 2018.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *ICLM16: Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsI. *Mathematical Programming*, 14(1):265–294, December 1978. ISSN 0025-5610, 1436-4646. doi: 10.1007/BF01588971. URL `https://link.springer.com/article/10.1007/BF01588971`.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

T. Powers, J. Bilmes, S. Wisdom, D.W. Krout, and L. Atlas. Constrained robust submodular optimization. In *NIPS Workshop on Optimization for Machine Learning*, December 2016.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. Large-margin learning of submodular summarization models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pp. 224–233, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-19-0. URL `http://dl.acm.org/citation.cfm?id=2380816.2380846`.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*, 2018.

Sebastian Tschiatschek, Rishabh Iyer, Haochen Wei, and Jeff Bilmes. Learning Mixtures of Submodular Functions for Image Collection Summarization. In *Neural Information Processing Systems (NIPS)*, 2014.

Sebastian Tschiatschek, Aytunc Sahin, and Andreas Krause. Differentiable submodular maximization. In *Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2731–2738, 7 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf`.

J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pp. 67–74, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-047-0. doi: 10.1145/1374376.1374389. URL `http://doi.acm.org/10.1145/1374376.1374389`.

Scott Wisdom, Thomas Powers, James Pitton, and Les Atlas. Interpretable Recurrent Neural Networks Using Sequential Sparse Recovery. In *NIPS Workshop on Interpretable Machine Learning for Complex Systems, arXiv:1611.07252*, December 2016. URL `https://arxiv.org/abs/1611.07252`.

Scott Wisdom, Thomas Powers, James Pitton, and Les Atlas. Building Recurrent Networks by Unfolding Iterative Thresholding for Sequential Sparse Recovery. In *Proc. ICASSP*, New Orleans, LA, USA, March 2017.

Bo Xin, Yizhou Wang, Wen Gao, David Wipf, and Baoyuan Wang. Maximal sparsity with deep networks? In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4340–4348. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6346-maximal-sparsity-with-deep-networks.pdf`.

6 Appendix

For completeness, we show here that the SCMM function in equation equation 2 is monotone non-decreasing submodular.

Proposition 1 *Let $m : 2^V \to \mathbb{R}_+$ be a modular function and $g : \mathbb{R} \to \mathbb{R}$ be a concave function. The function $f : 2^V \to \mathbb{R}$ defined as $f = g(m(A))$ is submodular.*

Proof: Given $A \subseteq B \subset V$ and $v \in V \backslash B$, let $m(A) = x$, $m(B) = y$, and $m(\{v\}) = z$. Then $0 \leq x \leq y$, and $0 \leq c$. Since g is concave, $g(x + z) - g(x) \geq g(y + z) - g(y)$, and therefore $g(m(A) + m(\{v\})) - g(m(A)) \geq g(m(B) + m(\{v\})) - g(m(B))$, which satisfies Equation (1). □

Furthermore, submodularity is closed under nonnegative addition so Equation (2) is submodular if $\alpha^u, h^u_a \geq 0$, $\forall u \in U, a \in V$.

For an additional example of how submodularity is useful for diverse evidence extraction, consider the following claim:

"Stripes only featured women."

For context, “Stripes” is a movie starring many male actors, so the claim is classified as Refuted. The DeepEncoder selects a true evidence a sentence,

> “He first gained exposure on Saturday Night Live, a series of performances that earned him his first Emmy Award, and later starred in comedy films–including Meatballs (1979), Caddyshack (1980), Stripes (1981), Tootsie (1982), Ghostbusters (1984), Scrooged (1988), Ghostbusters II (1989), What About Bob? (1991), and Groundhog Day (1993),”

and an incorrect one b that is quite semantically similar,

> “He also received Golden Globe nominations for his roles in Ghostbusters, Rushmore (1998), Hyde Park on Hudson (2012), St. Vincent (2014), and the HBO miniseries Olive Kitteridge (2014), for which he later won his second Primetime Emmy Award,”

both from actor Bill Murray’s Wikipedia page. Initially, the DGN rates those two sentences highly as well, giving them the top two scores of $f_\alpha(\{a\}) = 7.87$ and $f_\alpha(\{b\}) = 7.21$, respectively. Therefore, the DGN selects the same first evidence sentence as the DeepEncoder, but in the second layer, the incremental value of b drops significantly to $f_\alpha(\{a, b\}) - f_\alpha(\{a\}) = 4.74$, because it is highly redundant with a. Instead, the DGN selects c,

> “Stripes is a 1981 American buddy military comedy film directed by Ivan Reitman, starring Bill Murray, Harold Ramis, Warren Oates, P. J. Soles, Sean Young, and John Candy,”

which also provides information refuting the claim in the form of a list of male actor names, but which is semantically different enough from a to only drop from a score of $f_\alpha(\{c\}) = 6.73$ in the first greedy layer to $f_\alpha(\{a, c\}) - f_\alpha(\{a\}) = 5.21$ in the second greedy layer.