UNIVERSITY OF CALIFORNIA

Santa Barbara

Abstractive Text Summarization Using Hierarchical Reinforcement Learning

A thesis submitted in partial satisfaction of the

requirements for the degree Master of Science

in Computer Science

by

Mahnaz Koupaee

Committee in charge:

Professor William Wang, Chair

Professor Linda Petzold

Professor Xifeng Yan

September 2018

The thesis of Mahnaz Koupaee is approved.

Linda Petzold

Xifeng Yan

William Wang, Committee Chair

June 2018

Abstractive Text Summarization Using Hierarchical Reinforcement Learning

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor professor Wang of the computer science department at UCSB who steered me in the right the direction whenever he thought I needed it.

I would also like to thank the experts who were in my thesis committee: professor Petzold and professor Yan for their helpful comments on my thesis.

Finally, I must express my very profound gratitude to my parents and to dearest husband for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

ABSTRACT

Abstractive Text Summarization Using Hierarchical Reinforcement Learning

by

Mahnaz Koupaee

Sequence-to-sequence models have recently gained the state of the art performance in summarization. However, not too many large-scale high-quality datasets are available and almost all the available ones are mainly news articles with the specific writing style. Moreover, abstractive human-style systems involving a description of the content at a deeper level require data with higher levels of abstraction.

On the other hand, attention-based sequence-to-sequence neural networks optimizing log-likelihoods at word-level or discrete metrics such as ROUGE at sequence-level has achieved promising results on abstractive text summarization but they are far from perfect: the first group of models may fail to handle out of vocabulary words and often produce repetitive words and incorrect facts. The latter methods using reinforcement training while beating the state of the art methods in terms of discrete evaluation metrics, produce non-readable, sometimes irrelevant summaries.

We initially present WikiHow, a dataset of more than 230,000 article and summary pairs extracted and constructed from an online knowledge base written by different human authors. The articles span a wide range of topics and therefore represent high diversity styles. We also evaluate the performance of the existing methods on WikiHow to present its challenges and set some baselines to further improve it.

Moreover, to overcome the problems of existing summarization systems, we propose a novel hierarchical reinforcement learning architecture which makes decisions in two steps: the high-level policy decides on the sub-goal for generating the next chunk of summary and the low-level policy performs primitive actions to fulfill the specified goal. By reinforcing summarization at different levels, our proposed model outperforms the existing approaches in terms of ROUGE and METEOR scores.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. Introduction

Summarization is the process of generating a shorter version of a piece of text while preserving important context information. Text summarization allows readers to efficiently absorb text information by aggregating the most important ideas across a set of sentences.

Traditionally, there are two main types of summarization: extractive summarization [1-4] and abstractive summarization [5-7]. Extractive summarization identifies and ranks the important sentences of the text and sentences with the highest ranks form the summary. While abstractive summarization generates novel sentences like what a human author does.

The extractive methods are more successful in generating grammatically correct sentences since they simply copy parts of the original text. In contrast, abstractive methods attempt to generate novel phrases and sentences which resemble human-written summaries. Abstractive summarization is harder since the generated summaries also need to be syntactically and semantically coherent along with their relevance to the original articles. Therefore, due to the difficulty of the abstractive approaches, most of the existing summarization methods are extractive.

Sequence-to-sequence neural networks have recently obtained significant performance improvement on abstractive summarization [6,7]. However, some issues still exist: there are a lot of repetitions in the final summary; incorrect factual statements might be generated, and rare words or the out of vocabulary words cannot be handled properly. There have been recent attempts such as CopyNet [8] and pointer-generator mechanism [9,10] that learn to copy from the source article. For example, pointer-generator mechanism utilizes a switch between generating a word from the vocabulary or copying the exact word from the article.

Recent work by [9,11] trains the pointer-generator network to only copy words from the out-of-vocabulary words, however the pointer-generator network [10] allows the model to make its decision on generating or pointing regardless of the existence of the words in the vocabulary at each time step.

Most of the state-of-the-art methods [9,10] optimize log-likelihoods over hidden context and decoder input alignments. Maximizing log-likelihood is not always able to produce good results [11,12]. The reason is that while there is only one reference summary for each example, we can still have multiple valid summaries with words arranged in different orders. Another issue is that there is no immediate task-related feedback on the quality of generation probability estimation at each time step.

There have been a few attempts to use reinforced training in sequence generation [12,13] and a very recent attempt [11] applying similar ideas to the summarization task. They directly optimize ROUGE-L score for word generation during training and to avoid repetition they only force the decoder not to generate a repetitive trigram at test time.

However, text summarization consists of different subtasks and instead of using a reward for optimizing the whole model, we can divide the task into its subtasks and use different rewards at different time granularities to optimize decisions of the model. We propose an HRL agent [14], a novel, hierarchical reinforced model that optimizes its decision in two levels by comparing the generated summary with the reference summary in form of the ROUGE [15] score and taking into account a penalty for generating repetitive chunks. Our system mainly consists of a **concept-level policy**, a **surface-level policy,** and an **internal critic**. The concept-level policy operates at a higher level and decides a sub-goal (to point or to generate), while the surface-level policy performs the primitive actions (selecting words) following the sub-goal. The internal critic calculates a uniqueness ratio of the generated chunks (seen as the intrinsic reward) to help train the surface-level policy, relieving the repetition issue. The extrinsic reward (ROUGE score) is employed to train both the concept-level and surface-level policies.

The high-level policy decides on the non-repetitive generation sub-goal, either by using predefined vocabulary or by pointing to a specific part of the original article. Later, the low-level policy chooses primitive actions (selecting words) by following the goal specified by the high-level policy. The intrinsic reward (the ROUGE score of the generated sequence) provided by the internal critic, helps train the low-level agent to better fulfill the sub-goal, while the extrinsic reward (the ROUGE score of the generated sequence along with uniqueness ratio) will guide the high-level goal exploration. Optimizing the ROUGE score enables the model to generate words which may not be in the exact order as the reference summary but are still able to generate a valid summary especially for longer sequences.

Aside from improving summarization techniques and methods, the existence of large-scale datasets is the key to the success of these models. Moreover, the length of the articles and the diversity in their styles can create more complications.

Almost all existing summarization datasets such as DUC [16] Gigaword [17], New York Times [18] and CNN/Daily Mail [9] consist of news articles. The news articles have their own specific styles and therefore the systems trained on only news may not be generalized well. On the other hand, the existing datasets may not be large enough (DUC) to train a sequence-to-sequence model, the summaries may be limited to only headlines (Gigaword), they may be more useful as an extractive summarization dataset (New York Times) and their abstraction level might be limited (CNN/Daily mail).

To overcome the issues of the existing datasets, we also present a new large-scale dataset called WikiHow [19] using the online WikiHow[1] knowledge base. It contains articles about various topics written in different styles making them different from existing news datasets. Each article consists of multiple paragraphs and each paragraph starts with a sentence summarizing it. By merging the paragraphs to form the article and the paragraph outlines to form the summary, the resulting version of the dataset contains more than 200,000 long-sequence pairs. Moreover, each paragraph and its outline can form pairs for short summary generation tasks. We then present two features to show how abstractive our dataset is. Finally, we analyze the performance of some of the existing extractive and abstractive systems on WikiHow as benchmarks for further studies.

The contributions of this work are two-fold:

1. We introduce a large-scale, diverse dataset with various writing styles called WikiHow, convenient for long-sequence text summarization and evaluate the performance of the existing systems to create benchmarks and understand the challenges better
2. We propose a novel hierarchical reinforcement model to improve the summary generation process by maximizing the ROGUE score and uniqueness ratio and our experimental results show that this method improves over the previous state-of-the-art methods in terms of ROUGE and METEOR scores.

[1] http://www.wikihow.com/

In the following sections, we initially introduce our WikiHow dataset. Different aspects of this dataset compared to existing datasets along with its unique features and the construction methodology is described.

Next, we propose our hierarchical reinforced summarization agent to overcome some of the existing issues in summarization. Reinforcement learning and hierarchical reinforcement learning are described as preliminaries and our method is proposed on top of them. Finally, to set benchmarks for our new dataset and show the superiority of our proposed method, multiple baselines are implemented and the results are provided. The thesis will be concluded by some future directives.

II. WikiHow Dataset

We present WikiHow [19], a new large-scale summarization dataset consisting of diverse articles specifying the steps of doing a task form WikiHow knowledge base. The WikiHow features discussed in this chapter can create new challenges to the summarization systems. We hope that the new dataset can attract researchers' attention as a choice to evaluate their systems.

To show the superiority of the newly proposed dataset, we initially need to have a look over the existing datasets and figure out the existing advantages and disadvantages of them. There have been multiple datasets used over the years both by extractive and abstractive systems. We will start this chapter by reviewing the existing datasets which later lead us to introduce the new WikiHow dataset and its unique features.

A. Existing Datasets

There are several datasets used to evaluate the summarization systems. We briefly describe the properties of these datasets as follows.

DUC: The Document Understanding Conference dataset [16] contains 500 news articles and their summaries capped at 75 bytes. The summaries are written by human authors and there exists more than one summary per article which is its major advantage over other existing datasets. Having only one reference summary is highly restrictive for evaluating methods by discrete metrics such as ROUGE [15]. The reason is that while there is one piece of text, it can be summarized into multiple acceptable ways.

The DUC dataset cannot be used for training models with large number of parameters since the number of existing samples are limited (only 500 pairs) and therefore is used along with other datasets [4,6].

Gigaword: Another collection of news articles used for summarization is Gigaword [17]. The original articles in the dataset do not have summaries paired with them. However, some prior work [6,7] used a subset of this dataset and constructed pairs of summaries by using the first line of the article and its headline. The average length of the article and summary pairs are around 15 and 10 tokens, making the dataset suitable only for short text summarization tasks.

New York Times: The New York Times (NYT) dataset [18] is a large collection of articles published between 1996 and 2007. The features of the articles and their summaries pairs are best convenient for extractive summarization systems. While this dataset has been mainly used for extractive systems [20,21] and Paulus.et.al. [11] are the first to evaluate their abstractive system using NYT.

CNN/Daily Mail: This dataset consists of online CNN and Daily Mail news articles and is mainly used in recent summarization papers [4,9,10]. It was originally developed for question/answering systems. The highlights associated with each article are concatenated to form the summary. Two versions of this dataset depending on the preprocessing exist. In the original version of the dataset, the named entities are anonymized. Authors in [4] have used the entity anonymization to create the anonymized version of the dataset and used it for summarization. The second version of the dataset replaces the anonymized entities with their actual values and create the non-anonymized version. See et al. [10] uses this version of the dataset to evaluate the performance of their proposed method.

NEWSROOM: This corpus [22] is the most recent large-scale dataset introduced for text summarization. It consists of diverse summaries combining abstractive and extractive strategies yet it is another news dataset and the average length of summaries are limited to 26.7.

B. WikiHow Dataset

The existing summarization datasets, consist of news articles. These articles are written by journalists and follow the journalistic style. The journalists usually follow the Inverted Pyramid style [23] (depicted in Figure 1) to prioritize and structure a text by starting with mentioning the most important, interesting or attention-grabbing elements of a story in the opening paragraphs and later adding details and any background information.

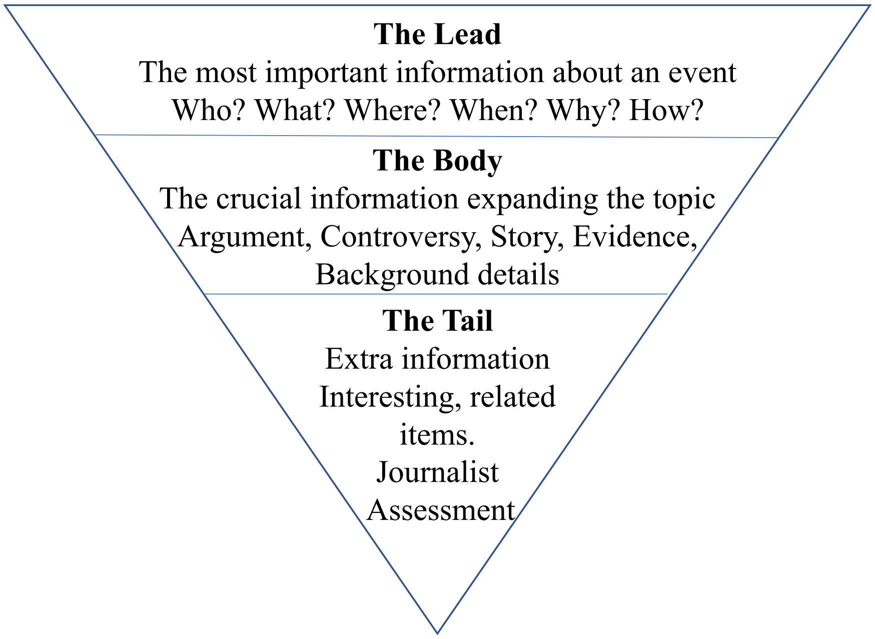

The Lead
The most important information about an event
Who? What? Where? When? Why? How?
The Body
The crucial information expanding the topic
Argument, Controversy, Story, Evidence,
Background details
The Tail
Extra information
Interesting, related
items.
Journalist
Assessment

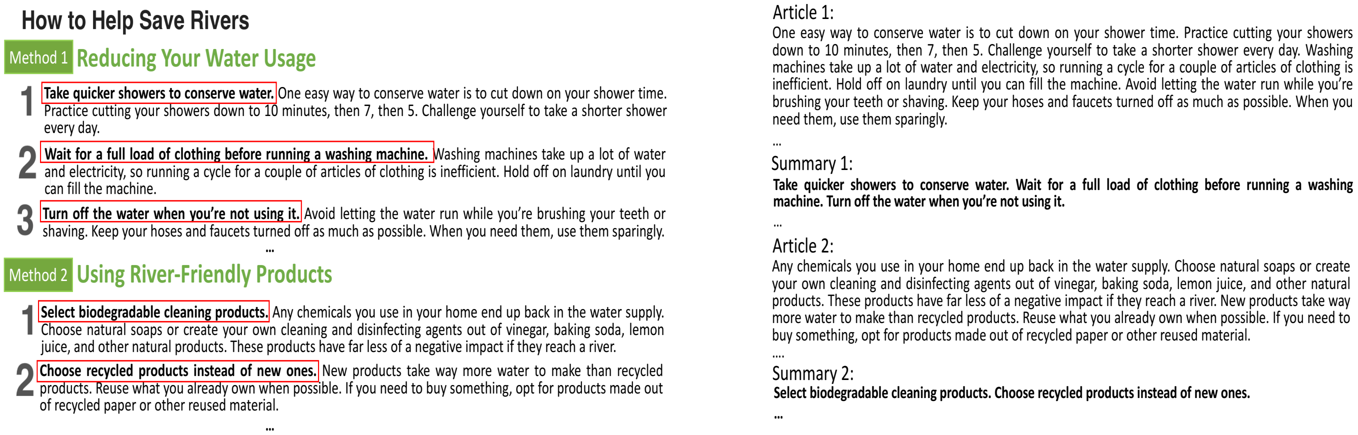
Figure 1. Inverted Pyramid writing style. The first few sentences of news articles contain the important information making Lead-3 baselines outperforming most of the systems.

This writing style might be the cause why lead-3 baselines (where the first three sentences are selected to form the summary) usually score higher compared to the existing summarization systems.

We introduce a new dataset called WikiHow, obtained from WikiHow data dump. This dataset contains articles written by ordinary people, not journalists, describing the steps of doing a task throughout the text. Therefore, the Inverted Pyramid does not apply to it as all parts of the text can be of similar importance.

1. WikiHow Knowledge Base

The WikiHow knowledge base contains online articles describing a procedural task about various topics (from arts and entertainment to computers and electronics) with multiple methods or steps and new articles are added to it regularly. Each article consists of a title starting with ``How to" and a short description of the article. There are two types of articles: the first type of articles describes single-method tasks in different steps, while the second type of articles represents multiple steps of different methods for a task. Each step description starts with a bold line summarizing that step and is followed by a more detailed explanation. A truncated example of a WikiHow article and how the data pairs are constructed is shown in Figure 2.



How to Help Save Rivers

Method 1 Reducing Your Water Usage

1 **Take quicker showers to conserve water.** One easy way to conserve water is to cut down on your shower time. Practice cutting your showers down to 10 minutes, then 7, then 5. Challenge yourself to take a shorter shower every day.

2 **Wait for a full load of clothing before running a washing machine.** Washing machines take up a lot of water and electricity, so running a cycle for a couple of articles of clothing is inefficient. Hold off on laundry until you can fill the machine.

3 **Turn off the water when you're not using it.** Avoid letting the water run while you're brushing your teeth or shaving. Keep your hoses and faucets turned off as much as possible. When you need them, use them sparingly.

...

Method 2 Using River-Friendly Products

1 **Select biodegradable cleaning products.** Any chemicals you use in your home end up back in the water supply. Choose natural soaps or create your own cleaning and disinfecting agents out of vinegar, baking soda, lemon juice, and other natural products. These products have far less of a negative impact if they reach a river.

2 **Choose recycled products instead of new ones.** New products take way more water to make than recycled products. Reuse what you already own when possible. If you need to buy something, opt for products made out of recycled paper or other reused material.

...

Article 1:

One easy way to conserve water is to cut down on your shower time. Practice cutting your showers down to 10 minutes, then 7, then 5. Challenge yourself to take a shorter shower every day. Washing machines take up a lot of water and electricity, so running a cycle for a couple of articles of clothing is inefficient. Hold off on laundry until you can fill the machine. Avoid letting the water run while you're brushing your teeth or shaving. Keep your hoses and faucets turned off as much as possible. When you need them, use them sparingly.

...

Summary 1:

Take quicker showers to conserve water. Wait for a full load of clothing before running a washing machine. Turn off the water when you're not using it.

...

Article 2:

Any chemicals you use in your home end up back in the water supply. Choose natural soaps or create your own cleaning and disinfecting agents out of vinegar, baking soda, lemon juice, and other natural products. These products have far less of a negative impact if they reach a river. New products take way more water to make than recycled products. Reuse what you already own when possible. If you need to buy something, opt for products made out of recycled paper or other reused material.

....

Summary 2:

Select biodegradable cleaning products. Choose recycled products instead of new ones.

...

Figure 2. An example of our new dataset: WikiHow summary dataset, which includes +200K summaries. The bold lines summarizing the paragraph (shown in red boxes) are extracted and form the summary. The detailed descriptions of each step (except the bold lines) will form the article. Note that the articles and the summaries are truncated and the presented texts are not in their actual lengths.

2. Data Extraction and Dataset Construction

We made use of the python Scrapy[2] library to write a crawler to get the data from the WikiHow website. The articles classified into 20 different categories cover a wide range of topics. Our crawler was able to obtain 142,783 unique articles (some containing more than one method) at the time of crawling (new articles are added regularly).

To prepare the data for the summarization task, each method (if any) described in the article is considered as a separate article. To generate the reference summaries, bold lines representing the summary of the steps are extracted and concatenated. The remaining parts of the steps (the detailed descriptions) are also concatenated to form the source article. After this step, 230,843 articles and reference summaries are generated.

[2] https://scrapy.org

There are some articles with only the bold lines i.e. there is no more explanation for the steps, so they cannot be used for the summarization task. To filter out these articles, we used a size threshold so that pairs with summaries longer than the article size will be removed. The final dataset is made of 204,004 articles and their summaries. The statistics of the dataset are shown in Table 1.

Dataset Size	230,843
Average Article Length	579.8
Average Summary Length	62.1
Vocabulary Size	556,461

Table 1. The WikiHow datasets statistics.

C. WikiHow Properties

The large scale of the WikiHow dataset by having more than 230,000 pairs, and its average article and summary lengths makes it a better choice compared to DUC and Gigaword corpus.

We also define two metrics to represent the abstraction level of WikiHow by comparing it with CNN/Daily mail known as one of the most abstractive and common datasets in recent summarization papers [4,9,10,11].

1. Level of Abstraction

Abstraction of the dataset is measured by calculating the unique n-grams in the reference summary which are not in the article. The comparison is shown in Figure 3. Except for common unigrams, bi-grams, and trigrams between the articles, and the summaries, no other common n-grams exist in the WikiHow pairs. The higher level of abstraction creates new challenges for the summarization systems as they must be more creative in generating more novel summaries.

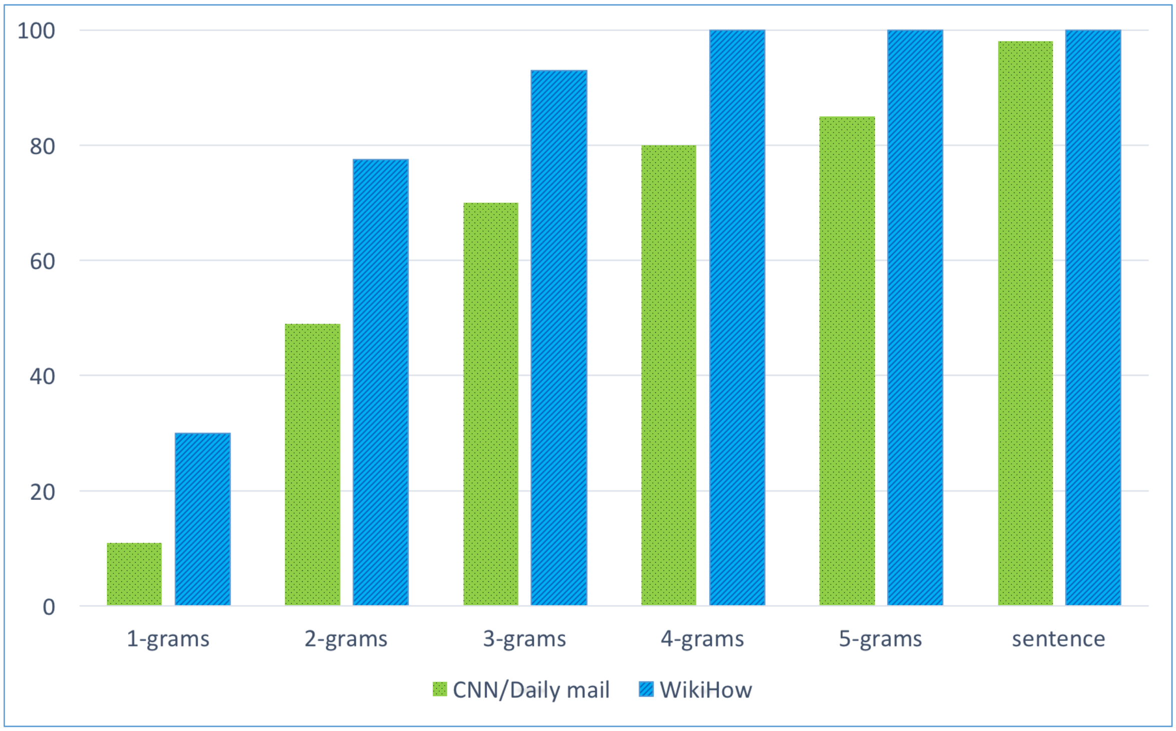


Figure 3. The Uniqueness of n-grams in CNN/Daily mail and WikiHow datasets.

2. Compression Ratio

We define the compression ratio to characterize the summarization. We first calculate the average length of sentences for both the articles and the summaries. The compression ratio is then defined as the ratio between the average length of sentences and the average length of summaries. The higher the compression ratio, the more difficult the summarization task, as it needs to capture higher levels of abstraction and semantics. Table 2 shows the results for WikiHow and CNN/Daily Mail. The higher compression ratio of WikiHow shows the need for higher levels of abstraction.

Dataset	WikiHow	CNN/Daily Mail
Article Sentence Length	100.68	118.73
Summary Sentence Length	42.27	82.63
Compression Ratio	2.38	1.44

Table 2. Compression ratio of WikiHow and CNN/Daily mail datasets. The represented article and summary lengths are the mean over all sentences.

III. Abstractive Text Summarization

In general, text summarization can be categorized into extractive summarization [1-4,9] and abstractive summarization [5-7]. While extractive summarization might provide summaries that have fewer grammar errors, it is primarily limited to the reusing of human written sentences with information retrieval, clustering, or re-ranking approaches [24].

In this work, we focus on improving abstractive methods, because of their ability to generate novel sentences. Knowing what the existing summarization systems are able to offer and what issues they are facing can help future design decisions made much more wisely. In the following sections, we will look at the existing generation systems and inspect the pros and cons of them. Next, the preliminaries of the proposed method are described and finally, the hierarchical reinforced summarization agent is described in detail.

A. Related Work

1. Sequence-to-sequence abstractive summarization

The overall architecture of sequence-to-sequence models with attention mechanism is shown in Figure 4.

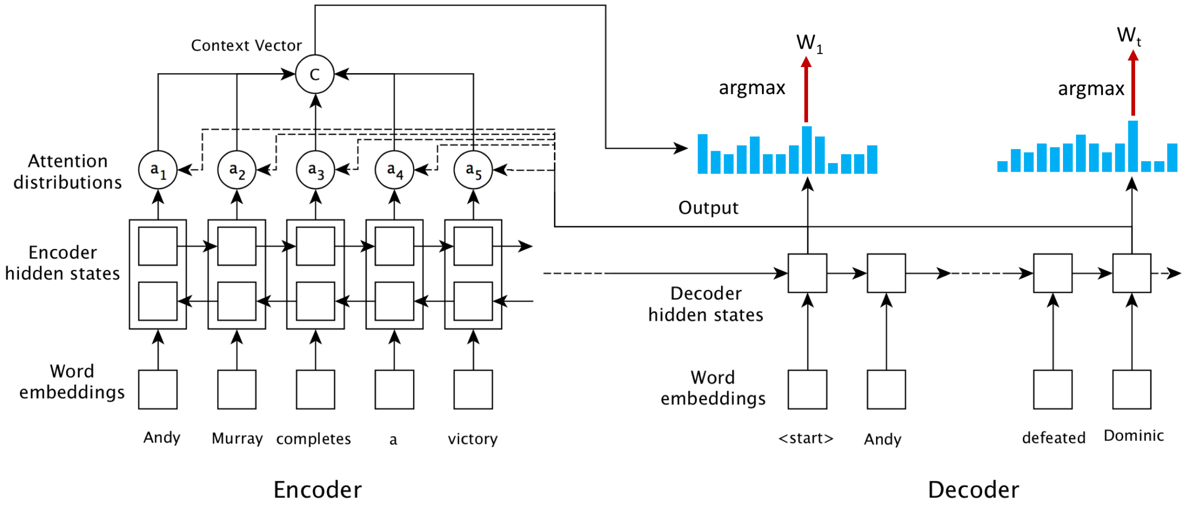


Figure 4: Sequence-to-sequence model with attention mechanism

A sequence-to-sequence model consists of an encoder and a decoder. The encoder is responsible for reading the input tokens. The input tokens represented by their vectors known as word embeddings are read and fed into a bi-directional LSTM to generate the encoder hidden states. The decoder receives the final encoder state and feeds it along with its input to a uni-directional LSTM to generate the decoder hidden state for the given time step. To help decoder generate the correct outputs, the distribution over the input tokens known as attention is calculated. Next, the context vector, the weighted average of attention weights is calculated and the decoder uses this weight to generate the final distribution over all the words in the vocabulary. Finally, the word with maximum probability is selected as the output of that time step.

Our work is clearly aligned with recent studies on sequence-to-sequence models [25] for abstractive summarization. A neural attention-based abstractive summarization model [6] for headline generation inspired by [26] on short texts (one or two sentences) is one of the earliest attempts to use neural networks to beat the performance of the existing traditional summarization models. The work done by Chopra et al. [7] is the extension of [6] which computes the conditional probabilities of next words using a recurrent decoder improving the results on short text summary generation. An attentional encoder-decoder recurrent neural networks for abstractive summarization is introduced by [9]. They also introduce a new dataset consisting of multi-sentence articles and summaries of on-line CNN and Daily Mail articles originally developed for question-answering systems [27] Their results also provided the benchmarks for later studies. A neural sequence model using hierarchical RNNs for extractive text summarization [4] which is also evaluated on CNN/Daily Mail dataset also showed promising results.

2. Pointer-generator mechanism

A major issue with sequence-to-sequence models is that there is no guarantee about the grammar and semantic coherence of the decoder output, and in reality, the generated sequence may be repetitive and not readable. Some recent attempts to address this issue include CopyNet [8], which learns to copy words and phrases from the input sequence. Pointer-generator mechanism is another recent attempt, which operates as a switch between abstractive and extractive summarization features.

The pointer network [28] uses attention as a pointer to select a member of the input sequence for the output. [9] also use this mechanism to produce rare out of vocabulary words and named entities. The same mechanism for OOVs on summarization and machine translation tasks is used by [29] which improves the performance on multiple datasets.

However, See et al. [10] train the model not only to be activated for OOVs but to freely select between pointing to word in input sequence or generating a novel word from vocabulary as shown in Figure 5. At each time step, the decoder has to make a decision whether it is better to point to a token in the input sequence or generate a novel word from the predefined vocabulary.

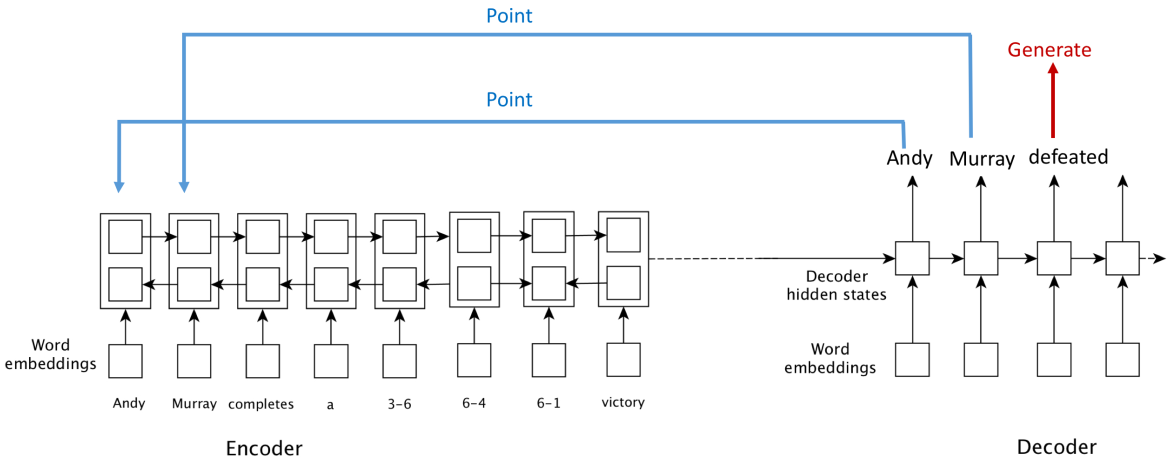


Figure 5: The pointer generator mechanism

Their results showed improvements over the existing pointer-generator models but the model tends to point more than to generate.

3. Reinforcement learning for text generation

Cross-entropy optimization using ground truth words causes exposure bias. This problem happens because during training the model has access to the actual sequence words, however at test time there is no such supervision and the model has to make use of its own predictions. To overcome the exposure bias, and optimizing at sequence level rather than word level, [12] and [17] trained sequence generation models using discrete metrics such as ROUGE and BLEU in a reinforced fashion. The most recent summarization work of Paulus et al. [11] also applies REINFORCE to abstractive summarization, using similar ideas.

These methods use the same architecture but with two main differences shown in Figure 6.

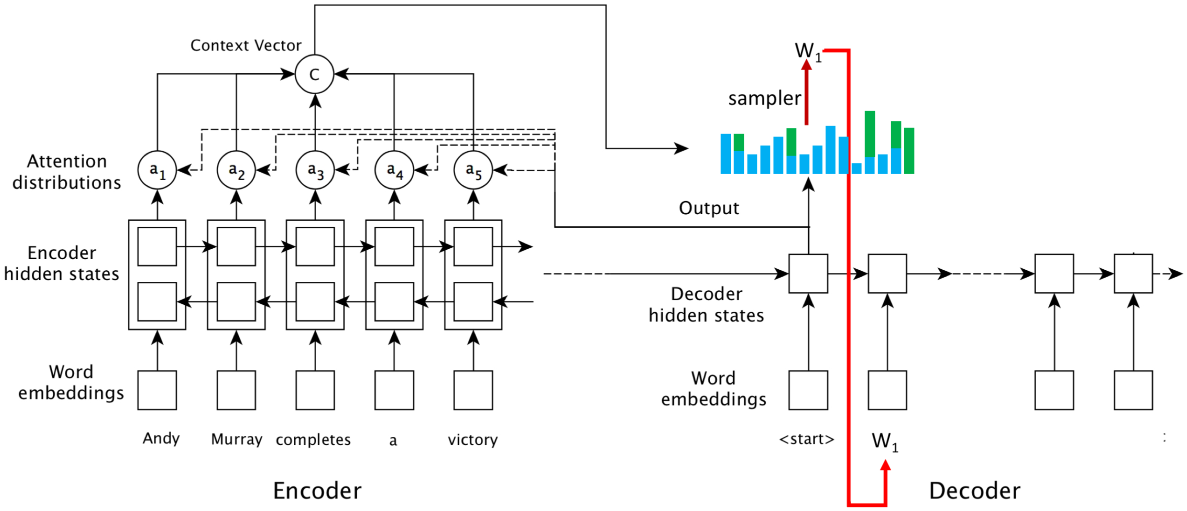


Figure 6: Sequence-to-sequence model with reinforce training

As shown in Figure 6, at each decoder time step, instead of selecting the word with maximum probability, a sampler selects a word from the vocabulary distribution. It is worth mentioning that the distribution is over both the words in the vocabulary (shown in blue) and over the words in the input article (shown in green). Moreover, to reduce the effect of exposure bias, the output of the previous time step is fed as the input of the next time step.

4. Hierarchical reinforcement learning

Hierarchical reinforcement learning (HRL) has shown significant improvements on Atari games [30,31]. There are two different policies in the typical HRL settings: a high-level policy that operates at the higher resolution to set sub-goals, and a low-level policy that performs primitive actions at the lower resolution to achieve the sub-goal from the high-level policy.

An internal critic is usually employed to control whether the low-level policy has achieved the sub-goal. Recently, some attempts have been done to apply HRL for practical applications.

[32] proposes a composite task-completion dialogue agent to handle the complex dynamics in dialogue systems, and [33] invents an HRL framework for video captioning which generates the semantic descriptions for videos phrase by phrase.

Our work is similar to that of [10] since we do not restrict the pointer-generator function solely to named-entities and OOVs and we let the model make its own decisions. But it is different from their work since it only maximizes the log-likelihood and uses a coverage mechanism to take into account the parts of the text which are already covered and therefore reduces repetition in the final generated summary. We use the hierarchical reinforcement framework with discrete metrics and calculate the reward in two levels making it different from the work by [11] which only use reinforcement learning.

B. Reinforcement Learning

Reinforcement Learning (RL) is a general-purpose framework for decision making, as depicted in Figure 7.

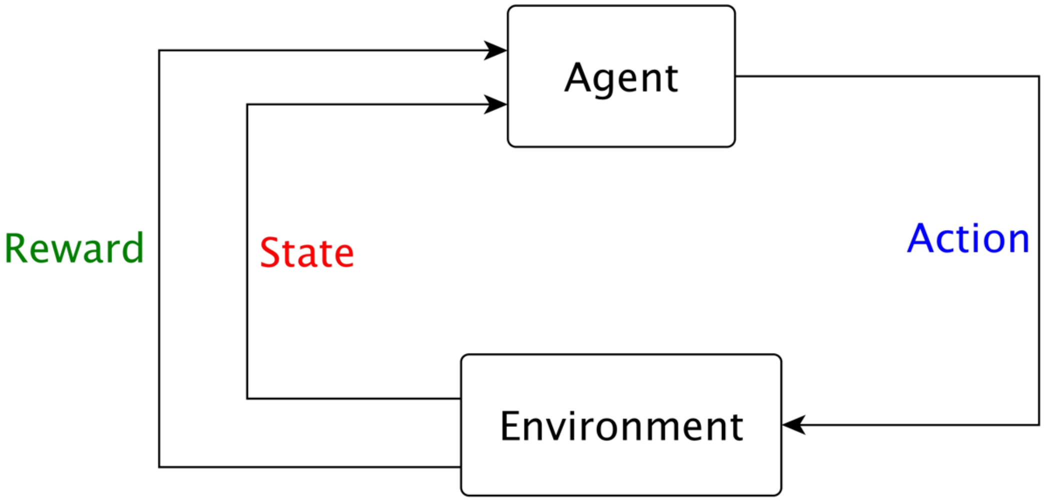


Figure 7: Reinforcement learning framework

On each step of interaction, the agent receives as input, some indication of the current state, s, of the environment; the agent then chooses an action, a, to generate an output. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar reinforcement signal, r. The agent's behavior should choose actions that tend to increase the long-run sum of values of the reinforcement signal. It can learn to do this over time by systematic trial and error, guided by a wide variety of algorithms.

Reinforcement learning can be understood using the concepts of agents, environments, states, actions, and rewards, all of which we'll explain below.

- Agent: An **agent** takes actions;
- Action (A): A is the set of all possible moves the agent can make.

- Environment: The world through which the agent moves. The environment takes the agent's current state and action as input, and returns as output the agent's reward and next state.
- State (S): A **state** is a concrete and immediate situation in which the agent finds itself; i.e. a specific place and moment. It can be the current situation returned by the environment or any future situation.
- Reward (R): A **reward** is the feedback by which we measure the success or failure of an agent's actions. From any given state, an agent sends output in the form of actions to the environment, and the environment returns the agent's new state (which resulted from acting on the previous state) as well as rewards if there are any. Rewards can be immediate or delayed. They effectively evaluate the agent's action.
- Policy (π): The **policy** is the strategy that the agent employs to determine the next action based on the current state. It maps states to actions, the actions that promise the highest reward.
- Trajectory: A sequence of states and actions that influence those states.

C. Hierarchical Reinforcement Learning

Hierarchical reinforcement learning uses the same ideas as reinforcement learning in a hierarchical fashion. The idea of hierarchical reinforcement learning is to break one big task into multiple subtasks. Therefore, to achieve the main goal, multiple sub-goals need to be fulfilled. The overall architecture of such a framework is shown in Figure 8.

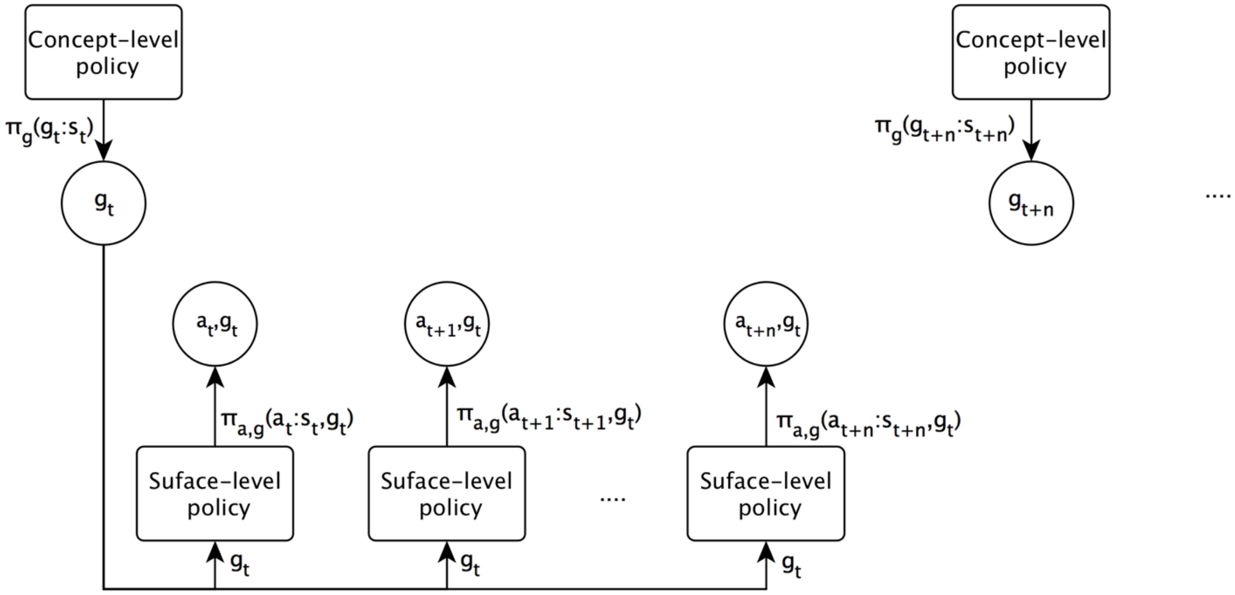


Figure 8: Hierarchical reinforcement learning

The concept-level policy or the manager is responsible for generating the high-level sub-goals. It receives the state of the agent and produces a probability over all the existing sub-goals. Once the sub-goal is selected, the surface-level policy or the worker performs primitive actions to fulfill the specified goal given the state of the agent. The surface-level policy takes actions until either the sub-goal is achieved or the maximum number of steps is reached. Next, the concept-level policy generates the next sub goal. This process continues achieving all sub-goals to finally fulfill the main goal.

D. Proposed Approach

We propose a method that combines deep reinforcement learning and hierarchical policies to learn a composite summarization agent. The overall figure is shown in Figure 9.

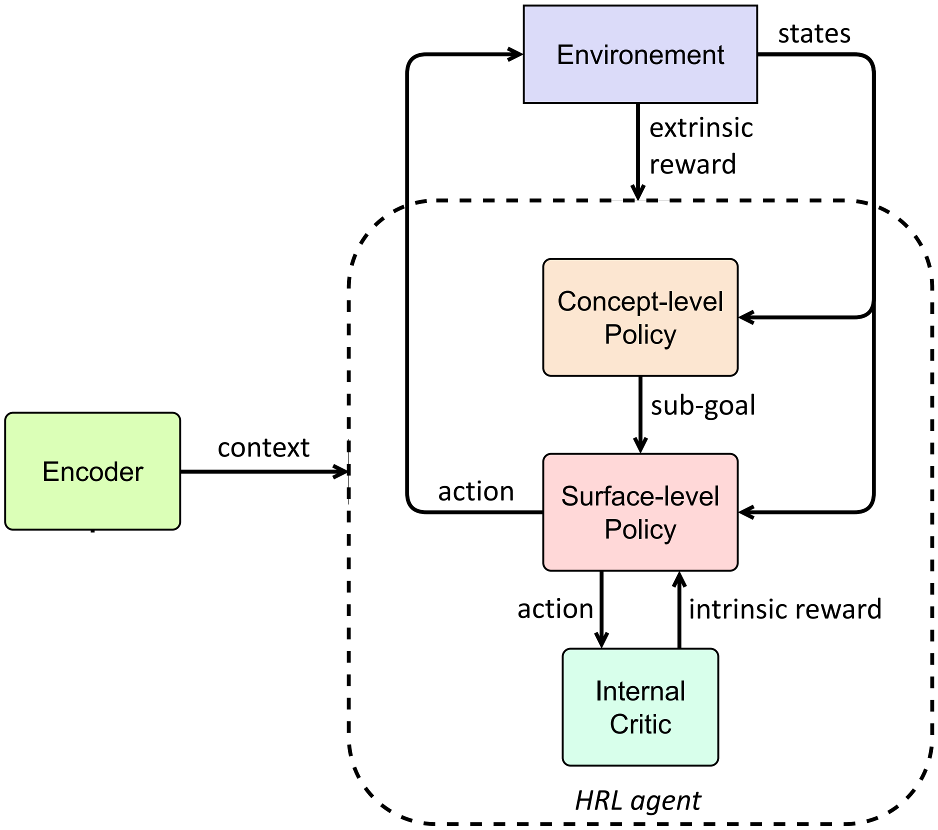


Figure 9: Overview of the hierarchical reinforced summarization agent [14].

Initially, the encoder receives the tokens (w_1, w_2, …, w_n) of the source article and passes them through a Bi-LSTM layer to produce attention distribution and the context vector. The hierarchical reinforcement learning (HRL) agent plays the role of the decoder. It receives the context from the encoder and the concept-level policy generates the goal distribution π_g over

$g \in G$ (set of all possible sub-goals) for the next time step using the context and the states from the environment.

The concept-level policy asks the surface-level policy to produce the next chunk by either pointing to a part of source article or using the vocabulary. The surface-level policy receives the goal and generates words to accomplish its task. The internal critic receives the generated chunk and calculates the intrinsic reward to guide the surface-level policy performance. The extrinsic reward is also calculated and helps the concept-level policy to optimize its goal generation. The process continues until either the EOS token is generated or the decoder reaches its maximum length.

1. Sequence-to-sequence attention model

The baseline sequence to sequence attention model shown in Figure 10 is similar to that of [9].

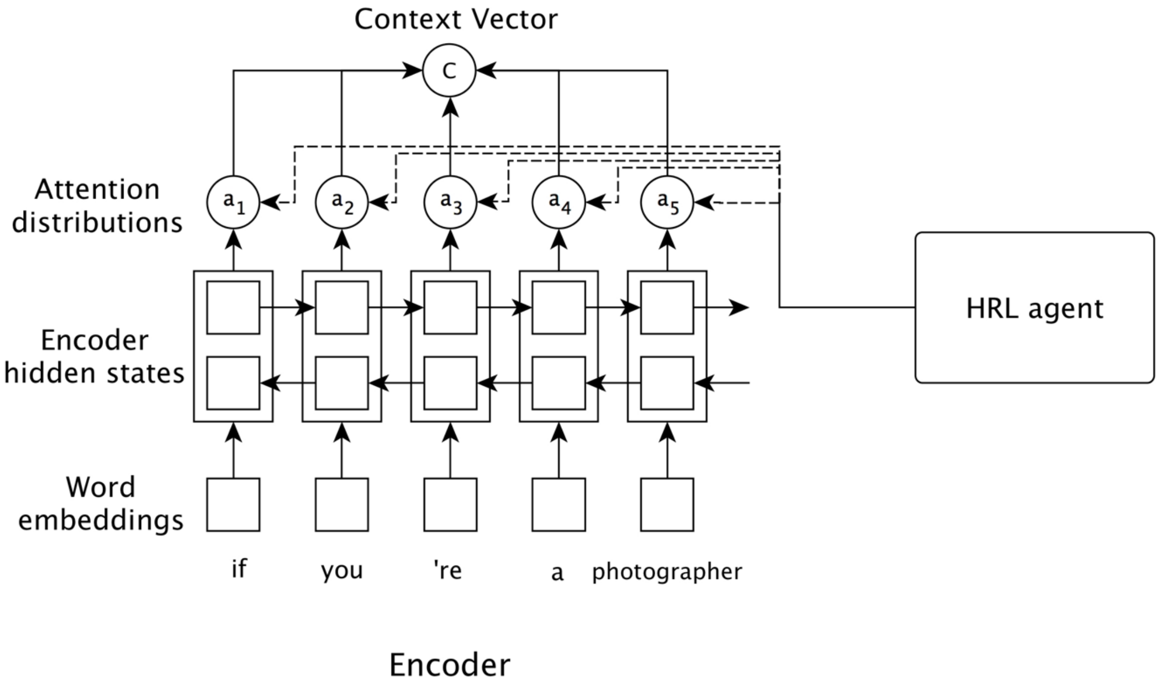


Figure 10: Overview of the attention model

The encoder which is a single-layer bidirectional LSTM generates the hidden states h_i for each article's token and the HRL agent generates state s_t at each decoder time step. The attention distribution which is a mechanism to help decoder produce summary words by knowing where to look at is a distribution over the source article's words and is calculated as follows:

$$a_t = \text{SoftMax} (w_h * h_i + w_s * s_t + b_a)$$

where w_h and w_s and the b_a are learned at training time. A new vector called context vector c_t is calculated for each decoder time step as well. The context vector is the sum of the encoder hidden states h_i of input tokens multiplied by the attention a_t at each time step.

$$c_t = \sum_i a_{it} h_i$$

2. Concept-level policy network

The concept-level policy is responsible for deciding on high-level goals for the surface-level agent to perform. This network concatenates the context vector c_t, decoder state s_t and decoder input d_t at each time step to form the current state. The output of the network is the probability distribution over high-level goals and is calculated as follows:

$$P_{goal} = \text{SoftMax}\ (w_g * (s_t + c_t + d_t) + b_g)$$

In our summarization framework, the concept-level policy represents the pointer-generator mechanism and the goals are either ***generate*** using predefined vocabulary or ***point*** using the attention distribution over source article words.

3. Surface-level policy network

When the high-level goal is generated, the surface-level policy performs primitive actions to fulfill the specified goal. This network receives the goal and along with its own state, outputs the probability distribution over all possible actions from the action space after a series of computation. The vocabulary distribution is initially calculated:

$$P_v = \text{SoftMax}\ (w'_v * (w_v * (s_t + c_t) + b_1) + b_2)$$

Where s_t and c_t are the decoder hidden state and the context vector respectively.

Later, the final distribution over actions is computed based on both the goal specified earlier and the vocabulary distribution as follows:

$$P_{action} = P_{goal\_0} * P_{vocab} + P_{goal\_1} * \sum_{i:w_i=w} a_{it}$$

The action space at each time step is the total number of words in vocabulary and the words in the original document.

4. Learning objective

At each time step, the surface-level summarization policy π(a) selects an action $a_t \in V$ (vocabulary) conditioned on the goal g_t specified by the concept-level policy. The scalar intrinsic reward R_i is calculated. The process continues until either the EOS token is generated or the maximum length is reached. The surface-level policy minimizes the following loss function:

$$L(\theta_s) = -E_{a \sim \pi(a|s,\, g,\, \theta)} [R_i(a)]$$

where *a* is the primitive action to fulfill the concept-level specified sub-goal for the current time step, *s* is the state and θ_s are the parameters of the surface-level network.

Since the defined loss function is based on discrete value reward and therefore non-differentiable, REINFORCE [34] algorithm is used to calculate its gradients. Therefore, the gradients of the above loss function are calculated as follows:

$$\nabla\theta_s \mathrm{L}_s = -\sum E_{s\sim\pi(a)}\,[\nabla\theta_s \log\pi(a)R_i]$$

$$\approx\; -\frac{1}{T}\sum_{t}^{T}\sum_{a\,\in A}[\nabla\theta_s \log\pi(a_t)\,R_{i,t}]$$

The above estimation of the gradient has high variance. There are different ways to reduce this variance and speed up the convergence. One common way to reduce the variance of the gradient calculated by REINFORCE algorithm is subtracting a baseline value b^s from the reward R_i. Therefore, the gradient becomes:

$$\nabla\theta_s \mathrm{L}_s = -\frac{1}{T}\sum_{t}^{T}\sum_{a\,\in A}[\nabla\theta_s \log\pi(a_t)\,R_{i,t} -\; b_t^s]$$

To obtain the value *b*, ($b = w_b * s_t + b_{base}$) is used where s_t is the decoder state and values w_b and the *bias* (θ_{base}) are learned by minimizing the squared mean error as follows:

$$L_{baseline} = (R_i(a)\text{-}b^s)^2$$

The concept-level policy is trained by the extrinsic reward to better guide the surface-level policy to achieve its goals. The loss function for the concept-level agent to minimize is the negative expected extrinsic reward under the distribution defined by the generation policy:

$$L(\theta_c\}) = -E_{g\,\sim\,\pi(g\,|\,s,\,\theta)}\,[R_e(g)]$$

where g is the selected goal for the current time step, s is the state and θ_c are the parameters of the concept-level network.

The gradient to optimize the parameters of the concept-level summarization policy is:

$$\nabla\theta_c \mathrm{L}_c = -\sum E_{g\sim\pi(g|s,\theta_c)}\left[\nabla\theta_c \log\pi(g) R_e\right]$$

$$\approx -\frac{1}{T}\sum_{t}^{T}\sum_{g\in G}\left[\nabla\theta_c \log\pi(g_t)\, R_{e,t}\right]$$

The final gradients for the concept-level policy after adding a baseline estimator to reduce the variance will yield:

$$\nabla\theta_c \mathrm{L}_c = -\frac{1}{T}\sum_{t}^{T}\sum_{g\in G}\left[\nabla\theta_c \log\pi(g_t)\,(R_{e,t} - b_t^{s'})\right]$$

The baseline is a one-layer neural network using the decoder hidden state s_t and is optimized by the squared mean error similar to the equation used for surface-level policy baseline.

5. Learning algorithm

The details of the training procedure are shown in Algorithm 1.

Algorithm 1 Training Procedure

```
Initialize HRL, $N_{XE}, N_{REINFORCE}$
for epochs 1 to $\text{N}_{\text{cross-entropy}}$ do
    Train HRL agent using cross-entropy
End for
for epochs $N_{XE}$ to $N_{REINFORCE}$ do
    Select a mini-batch of training pairs
    while token != <EOS> or decoder MAX length not reached do
        Select a sub-goal g form $\pi\ (g \mid s)$
        Select an action a from $\pi\ (a \mid s; g)$
        Calculate intrinsic-reward $R_i$
        Calculate extrinsic-reward $R_e$
         Update the surface-level policy using equation (8)
         Update the concept-level policy using equation (12)
    End while
End for
```

The REINFORCE algorithm starts with a random policy and tries to converge to an optimal policy. However, in the case of text generation since the action space is very large, starting from a random policy almost never converges towards an optimal policy. To overcome this problem, we make use of imitation learning and start with a pre-trained model. We let the model to gradually explore and make use of its own predictions. We use the cross-entropy loss (XE) to train the model initially. After N_{XE} training epochs, the REINFORCE algorithm is applied for $N_{REINFORCE}$ epochs, allowing the HRL model to explore among possible goals and actions, and thus learn towards better policies.

Once the cross-entropy optimization is done, the HRL agent will start making use of its own explorations and REINFORCE is used to optimize the concept-level and surface-level summarization policies. For each iteration, a random mini-batch of training pairs is selected. For each decoding step and for each pair, while the EOS token is not generated, a sub-goal is selected by sampling from the goal distribution. Next, based on that goal, a token sampled from the actions distribution is generated and is fed as the input to the next decoding step. To update the parameters of the surface-level policy, the concept-level policy is kept fixed, the intrinsic reward is calculated and the optimization step is performed. The intrinsic reward is defined as the ratio of the total number of unique trigrams in the generated sequence by the total number of trigrams of the reference summary:

$$\mathrm{R(i)} = \frac{\mathrm{SummaryUniqueTrigrams}}{\mathrm{ReferenceUniqueTrigrams}}$$

Later, the surface-level policy is fixed and the concept-level policy will be updated by calculating the extrinsic reward which is the ROUGE-2 score (better performance compared to ROUGE-1 and ROUGE-L) of the generated sequence:

$$R(e) = \text{ROUGE-2}(\text{summary})$$

The reason for using trigrams is that our datasets rarely have the same trigram more than once.

IV. Experimental Evaluation

Over the past few sections, we pointed out the issues and challenges that the existing summarization systems face. To overcome some of the mentioned problems, we introduced a new dataset called WikiHow [19] and a new summarization method, HRL Agent [14], using hierarchical reinforcement learning. We implemented multiple extractive and abstractive baselines to first, set new benchmarks for WikiHow dataset and second to show that our proposed HRL agent is able to outperform the existing state-of-the-art abstractive systems.

In the following parts, we briefly describe the datasets, baselines and experimental settings used for evaluation. Later, the comparison between the proposed method with other baselines is performed both quantitatively and qualitatively.

A. Datasets

To evaluate the performance of the proposed summarization method and illustrate the features of the new dataset, we used both the CNN/Daily Mail and WikiHow datasets. The CNN/Daily Mail dataset is one of the most common datasets used for evaluating summarization systems. The results on WikiHow dataset are the benchmarks set for the first time.

1. CNN/Daily Mail

This dataset which is mainly used in recent summarization papers [4,9,10] consists of online news articles and was originally developed for question/answering systems. We made use of the scripts available by [10][3] to preprocess the CNN/Daily Mail dataset and obtain the

[3] https://github.com/abisee/pointer-generator

same non-anonymized version of the dataset with the actual named entities with 287,113 training pairs, 13,368 and 11,490 validation and test pairs (about 3.5% of total) respectively.

2. WikiHow

Most of the new summarization papers made use of CNN/Daily Mail dataset which was originally used for question/answering systems. We also make use of our new proposed dataset called WikiHow, obtained from WikiHow data dump for our experiments. The final dataset is made of 180,127 articles and their summaries. The validation and test sets have 6,000 pairs (about 3.5% of total) and the rest is in the training set. This new dataset contains longer articles and summaries and its summaries are more abstractive compared to the CNN/Daily Mail dataset.

B. Evaluated Systems

Multiple extractive and abstractive systems are implemented and used for evaluation. The purpose of some of the implemented baselines is to merely set benchmarks for the WikiHow dataset. Below we will briefly introduce the implemented summarization systems used for evaluations.

1. TextRank Extractive system

An extractive summarization system [35,36] using a graph-based ranking method to select sentences from the article and form the summary.

2. Sequence-to-sequence Model with Attention

A baseline system applied by Chopra et al. [7] and Nallapati et al. [9] to abstractive summarization task to generate summaries using the predefined vocabulary. This baseline is not able to handle Out of Vocabulary words (OOVs).

3. Pointer-generator abstractive system

A pointer-generator mechanism (See et al. [10]) allowing the model to freely switch between copying a word from the input sequence or generating a word from the predefined vocabulary.

4. Pointer-generator with coverage abstractive system

The pointer-generator baseline with added coverage loss (See et al. [10]) to reduce the repetition in the final generated summary.

5. Lead-3 baseline

A baseline selecting the first three sentences of the article to form the summary. This baseline cannot be directly used for the WikiHow dataset as the first 3 sentences of each article only describe a small portion of the whole article. We created the Lead-3 baseline by extracting the first sentence of each paragraph and concatenated them to create the summary.

6. Reinforcement Learning Baseline (RL)

To show the improvements made by hierarchical reinforcement learning, we also implemented the reinforcement learning baseline. This baseline directly optimizes ROUGE score in one level.

7. Hierarchical Reinforcement Learning Agent (HRL)

The final implemented system is the proposed HRL agent. This method optimizes the discrete ROUGE metric in one level and the uniqueness ratio in another level, trying to generate better summaries in a hierarchical fashion.

C. Experimental Settings

The encoder is limited to 400 tokens and articles with more than 400 tokens are truncated and the decoder is limited to produce 100 token summaries (Larger thresholds must be selected for WikiHow dataset but we skip this for simplicity). All hidden layers are of size 256. The word embeddings are 128 dimensional and are learned in the training process. A 50k vocabulary size is used. Increasing the size of vocabulary does not make significant improvements in performance but rather increase the training time.

The summarization model is trained using cross-entropy with a batch size of 16, Adagrad optimizer with learning rate 0.01 and gradient clipping with a maximum gradient norm of 2 and all the parameters are tuned on the validation set. When there is no significant improvement of the model on the validation set, the cross-entropy optimization is disabled and REINFORCE uses the intrinsic and extrinsic rewards to optimize the HRL summarization agent with the learning rate of 0.001. Similar to the previous phase, the training is done for some epochs until there is no significant improvement on the validation set. During the test time, the summaries are generated using beam search decoding with beam size 4. The summary of the settings used during evaluations is represented in Table 3.

Encoder length	400	**Batch size**	16
Decoder length	100	**Optimizer**	Adam
Hidden dimension	256	**learning rate**	0.01
Embeddings dimension	128	**Beam size**	4

Table 3: The experimental settings

D. Results

To illustrate the performance of the proposed method, numerical evaluation and qualitative analysis of the results is performed. First, the performance of the implemented baselines is evaluated in terms of two known discrete metrics ROUGE and METEOR. ROUGE score is based on counting the number of exact common n-grans between the reference summary and the automatically generated summary by summarization system. Due to the dependence on the exact matches, this metric is not able to count for synonyms and paraphrases, thus reducing the flexibility of evaluations. The METEOR metric first tries to find alignments between the words in reference and generated summaries and then counts the number of aligned words. This metric has also the ability to consider synonyms, paraphrases, and word stems.

The qualitative results try to look at the underlying causes of such differences in the performance of different systems along with the overall quality of the generated summaries by presenting some sample outputs.

1. Quantitative Results

The results of our HRL summarization model on our new dataset, the WikiHow and the CNN/Daily Mail (non-anonymized version) are shown in Table 4.

Model	CNN/Daily Mail				WikiHow			
	ROUGE			METEOR	ROUGE			METEOR
	1	2	L	exact	1	2	L	exact
TextRank [35]	35.23	13.90	31.48	18.03	27.56	7.41	25.51	**12.92**
Seq-to-seq+attention [7]	31.33	11.81	28.53	12.03	22.04	6.27	20.87	10.06
Pointer-generator [10]	36.44	15.66	33.42	15.35	27.30	9.10	25.65	9.70
Pointer-generator+coverage [10]	39.53	17.28	36.38	17.32	28.53	9.23	26.54	10.56
RL baseline	40.03	17.42	36.47	17.85	29.25	10.05	26.92	10.94
HRL agent [14]	**40.40**	**17.82**	**36.75**	**18.60**	**30.20**	**10.58**	**28.15**	12.21
Lead-3 baseline	40.34	17.70	36.57	20.48	26.00	7.24	24.25	12.85

Table 4: The ROUGE-F-1 and METEOR scores of different methods on non-anonymized version of CNN/Daily Mail dataset and WikiHow dataset. The ROUGE scores are given by the 95% confidence interval of at most ±0:25 in the official ROUGE script.

We used the Pyrouge package[4] to report the F1 score for ROUGE-1, ROUGE-2 and ROUGE-L and the METEOR[5] to evaluate our method. We also implemented the reinforcement learning (RL) baseline and report the lead-3 baselines (first three sentences of the article as a summary) and pointer-generator network [10] to compare with our results. As it can be seen from the results, our HRL agent obtains higher ROUGE and METEOR scores than the existing abstractive baselines. The coverage mechanism of [10] performs better on the CNN/daily mail dataset in comparison to the WikiHow dataset. The METEOR score boost of the RL and HRL is more compared to the ROUGE score boost. The better performance of the HRL compared to the baseline RL shows the improvement this framework can result. We also reported the lead-3 for WikiHow. As stated earlier, each of summary sentences in this

[4] https://pypi.python.org/pypi/pyrouge/0.1.3

[5] www.cs.cmu.edu/~alavie/METEOR

dataset gives the gist of one or a few paragraphs of the original article, so we concatenated the first sentences of each paragraph to form the Lead-n baseline.

As it can be seen, the summarization systems perform a lot better on CNN/Daily mail compared to the WikiHow dataset with lead-3 outperforming other baselines due to the news inverted pyramid writing style described earlier. On the other hand, the poor performance of lead-3 on WikiHow shows the different writing styles in its articles. Moreover, all baselines perform about 10 ROUGE scores better on the CNN/Daily mail compared to the WikiHow. This difference suggests new features and aspects inherent in the new dataset which can be used to further improve the summarization systems.

2. Qualitative Results

Abstractive summarization methods do not generate the reference summary as is but generate summaries which convey the general idea of each text by using different words combination and order. This shows that metrics based on exact matches have their own shortcomings and are not able to fully evaluate the results of an abstractive summarization model. Our HRL model is also able to get a few related sentences and generate a completely novel sentence preserving the whole idea of the summarized sentences.

Comparing the results from two datasets, we realized that the models tend to copy more than to generate for CNN/Daily mail dataset. One difference of these two datasets is the length of the sentences. CNN/Daily Mail consists of longer sentences in comparison to the WikiHow dataset. We suppose that the model is able to summarize short sentences into a short summary but while encountering longer sentences it tends to copy one whole sentence. Moreover, as stated earlier, WikiHow dataset contains more abstractive summaries compared to the

CNN/daily mail, the model is exposed to more abstractive supervision during training and therefore generates more abstractive summaries at decoding time. There are also some cases in which the generated summary produces incorrect facts even though the sentence is grammatically correct. This usually happens in the existence of negations. More inspection needs to be done to find the cause and prevent such a problem.

Finally, from the results on the two datasets, we can observe that the WikiHow dataset has a good potential for further improvements. To further improve the results on this dataset knowing its specific features such as being procedural (each step is related to the previous ones and all steps together define a whole task), is an open area.

Table 5 represents a sample article from the CNN/Daily Mail dataset and the reference summary associated with it. We also show the outputs of three implemented baselines comparing the behavior of different summarization systems.

The output of the pointer-generator system plus coverage mechanism produces a readable correct output however, it is simply the copy of sentences from the original article making it an extractive system despite being intended to be abstractive.

The second baseline, the reinforcement learning summarization system, is repeating itself and generates the same fact in multiple sentences. Finally, the output of the HRL agent (the proposed method) is able to generate an abstractive summary (abstractive chunks shown in bold). The summary is different from the reference summary, however, can still be accepted as the summary of the article. This also shows the inability of discrete metrics such as ROUGE to evaluate the performance of abstractive summarization systems as they only look for exact matches.

Example	Article	More than 22 months ago, the body of Odin Lloyd was found in an industrial park in North Attleboro, Massachusetts. Lloyd had been shot seven times near a pile of construction waste. The semipro football player for the Boston Bandits was 27. Days after he was found dead, then-NFL star Aaron Hernandez was arrested and charged with murder. Hernandez was charged in the double homicide in May, and he pleaded not guilty.
	Reference summary	A jury has found ex-New England Patriots star guilty of murder. Aaron Hernandez also charged with murder in 2012 double homicide. 'Golden boy' had just inked 40$ million contract when troubles began.
	Pointer-generator with coverage	More than 22 months ago, the body of Odin Lloyd was found in an industrial park in North Attleboro, Massachusetts. Lloyd had been shot seven times near a pile of construction waste.
	RL summarization	the body of Odin Lloyd was found in an industrial park in North Attleboro, Massachusetts. He was found in an industrial park in North Attleboro, Massachusetts.
	HRL summarization	**Odin Lloyd was found dead** in an industrial park in North Attleboro, Massachusetts. he was found dead, then-NFL star Aaron Hernandez was arrested and charged with murder. **Hernandez** has pleaded not guilty.

Table 5: Sample output from multiple summarization systems

V. Conclusion

Summarization as the process of generating a shorter version of a piece of text while preserving the important context information is one of the most challenging NLP tasks. Abstractive text summarization due to its ability to generate novel sentences similar to what a human author does on one hand, and its difficulty to due to the need for preserving syntactic and semantic coherence, on the other hand, has been more of interest in recent research.

Systems using sequence-to-sequence models to generate summaries have gained the state-of-the-art performance however, they are not perfect. The generated summaries by such systems often produce repetitive sentences or incorrect factual statements. Moreover, the abstractive methods sometimes fail to be abstractive and generate rather extractive summaries by simple copying the chunks of the original article.

The key to success for training models with a large number of parameters is the existence of large-scale diverse datasets. Most of the existing datasets, follow the journalistic style known as pyramid style; Not all of them are large enough to be used to train a neural network with a lot of number of parameters and some of them best suit the extractive systems.

To overcome the issues both in the existing datasets and the abstractive systems, we proposed a dataset and a new summarization method using reinforcement learning.

We present WikiHow, a new large-scale summarization dataset consisting of diverse articles form WikiHow knowledge base. The WikiHow features discussed in the paper can create new challenges to the summarization systems. We hope that the new dataset can attract researchers' attention as a choice to evaluate their systems.

We also propose a hierarchical reinforcement learning model for abstractive text summarization by dividing the task into a set of sub-tasks and optimizing the sub-tasks. We validate that our model is able to outperform the existing summarization methods by achieving higher ROUGE and METEOR scores on CNN/Daily Mail and the new WikiHow dataset.

As for future work, we plan to improve the abstraction and semantic coherence of generated summaries as well as the results on procedural WikiHow dataset by considering the temporal connection of summary sentences.

References

1. Kathleen McKeown and Dragomir R Radev. 1995. Generating summaries of multiple news articles. In Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, pages 74–82. ACM.

2. G¨unes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of Artificial Intelligence Research, 22:457–479.

3. William Yang Wang, Yashar Medhad, Dragomir Radev, and Amanda Stent. 2016. A low-rank approximation approach to learning joint embeddings of news stories and images for timeline summarization. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA. ACL.

4. Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. AAAI.

5. Kavita Ganesan, Cheng Xiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In Proceedings of the 23rd international conference on computational linguistics, pages 340–348. Association for Computational Linguistics.

6. Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685.

7. Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In HLT-NAACL, pages 93–98.

8. Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393.

9. Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, C¸ a glar Gulc¸ehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. CoNLL 2016, page 280.

10. Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer generator networks. ACL.

11. Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304.

12. Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. CoRR, abs/1511.06732.

13. Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. An actorcritic algorithm for sequence prediction. CoRR, abs/1607.07086.

14. Mahnaz Koupaee, Xin Wang, William Wang, Abstractive Text Summarization Using Hierarchical Reinforcement Learning, in submission.

15. Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out: Proceedings of the ACL-04 workshop, volume 8. Barcelona, Spain.

16. Donna Harman and Paul Over. 2004. The effects of human variation in duc summarization evaluation. Text Summarization Branches Out.

17. Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction, pages 95–100. Association for Computational Linguistics.

18. Evan Sandhaus. 2008. The new york times annotated corpus. Linguistic Data Consortium, Philadelphia, 6(12):e26752.

19. Mahnaz Koupaee, William Wang, WikiHow: A Large Scale Text Summarization Dataset, in submission.

20. Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multidocument summarization. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 712– 721.

21. Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. arXiv preprint arXiv:1603.08887.

22. Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. arXiv preprint arXiv:1804.11283.

23. Horst Po¨ttker. 2003. News and its communicative quality: The inverted pyramid when and why did it appear? Journalism Studies, 4(4):501–511.

24. Dragomir R Radev, Hongyan Jing, Małgorzata Sty´s, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. Information Processing & Management, 40(6):919–938.

25. Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pp. 3104-3112.

26. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

27. Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In Advances in Neural Information Processing Systems, pages 1693–1701.

28. Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In Advances in Neural Information Processing Systems, pages 2692–2700.

29. Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. arXiv preprint arXiv:1603.08148.

30. Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In Advances in Neural Information Processing Systems, pages 3675–3683.

31. Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. arXiv preprint arXiv:1703.01161.

32. Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2231–2240.

33. Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang. 2017. Video captioning via hierarchical reinforcement learning. arXiv preprint arXiv:1711.11135.

34. Ronald J. William. 1992. Simple statistical gradient following algorithms for connectionist reinforcement learning. 8(3-4):229–256.

35. Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In Proceedings of the 2004 conference on empirical methods in natural language processing.

36. Federico Barrios, Federico L´opez, Luis Argerich, and Rosa Wachenchauzer. 2016. Variations of the similarity function of textrank for automated summarization. arXiv preprint arXiv:1602.03606.