Abstractive Summarization: A Survey of the State of the Art

Hui Lin, Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083–0688
{hui,vince}@.hlt.utdallas.edu

Abstract

The focus of automatic text summarization research has exhibited a gradual shift from extractive methods to abstractive methods in recent years, owing in part to advances in neural methods. Originally developed for machine translation, neural methods provide a viable framework for obtaining an abstract representation of the meaning of an input text and generating informative, fluent, and human-like summaries. This paper surveys existing approaches to abstractive summarization, focusing on the recently developed neural approaches.

1 Introduction

Nowadays online users suffer from an information overload: the vast amount of fast-growing textual information on the Web makes it challenging for a user to read all the material she is potentially interested in. Automatic text summarization, which is arguably one of the important high-level natural language applications, seeks to alleviate this information overload problem by automatically creating a concise summary of one or more text documents. Broadly, there are two kinds of text summarization tasks. Extractive summarization aims to create a summary by selecting a subset of the sentences in the input text that maximizes the coverage of important content while minimizing redundancy. In contrast, abstractive summarization aims to create an abstract representation of the input text and use natural language generation techniques to generate a summary. In comparison to extractive summaries, abstractive summaries are more challenging to produce, but are arguably a better approximation of human summaries as they may contain expressions that do not exist in the original text (Cohn and Lapata 2008). Table 1 shows an input document and the corresponding human-generated abstractive summary.

The focus of text summarization research has exhibited a gradual shift from extractive techniques to abstractive techniques in recent years, owing in part to significant advances in the development of neural methods. Originally developed for machine translation, neural methods have arguably revolutionized the way abstractive summarization research is conducted, creating new, exciting opportunities for summarization and generation researchers.

Source: the sri lanka government on wednesday announced the closure of government schools with immediate effect as a military campaign against tamil separatists escalated in the north of the country.
Summary: sri lanka **closes** schools as **war escalates**.

Table 1: An example on abstractive summarization taken verbatim from Zhou et al. (2017). The boldfaced words in the summary do not appear in the input document.

While several surveys on text summarization have been published over the years (Radev et al. 2002; Spärck Jones 2007; Lloret and Palomar 2012; Nenkova and McKeown 2012; Saggion and Poibeau 2012; Allahyari et al. 2017), their foci are extractive rather than abstractive summarization. Our goal in this paper is to provide the AI audience with a timely survey on abstractive summarization.

2 Evaluation Methods

Two types of evaluation methods are typically used to evaluate machine-produced summaries: manual evaluation and automatic evaluation.

In manual evaluation, human judges are asked to choose the best summary among several candidates by manually scoring each one along multiple dimensions of quality such as accuracy, clarity, and completeness (Greenbacker 2011). However, as manual evaluation is time-consuming and is particularly inefficient for large-scale evaluations, there have been a lot of attempts to develop automatic evaluation methods. For this reason, several automatic evaluation metrics have been developed. The widely-used metrics include (1) BLEU (Papineni et al. 2002), which was originally developed to evaluate machine translation systems; (2) METEOR (Denkowski and Lavie 2014), which addresses BLEU's weakness when applied to low-resource languages and has a better correlation with human judgment at the sentence/segment level than BLEU; (3) Pyramid (Nenkova et al. 2007), a well-known method for evaluating content selection in summarization; and (4) ROUGE (Lin 2004), a recall-based evaluation metric for summarization. Being one of the most popular metrics, ROUGE has several commonly used variants, such as ROUGE-N, which computes the n-gram recall between a candidate summary and a reference summary; ROUGE-SU, which uses skip-bigrams and unigrams to measure recall;

and ROUGE-L (Longest Common Subsequence), which requires in-sequence but not consecutive matches that reflect sentence-level word order n-grams.

Large-scale summarization evaluation efforts sponsored by the U.S. government, such as SUMMAC (1996–1998) (Mani et al. 2002), DUC (2000–2007) (Over et al. 2007), and TAC (2008–present), have played an important role in designing evaluation standards over the years. However, designing appropriate metrics for automatically evaluating summaries is challenging, and the definition of what constitutes a good summary remains largely an open question.

3 Datasets

Next, we describe several datasets that have been extensively used to evaluate automatic summarization systems.

DUC The summarization evaluations conducted as part of the NIST-sponsored Document Understanding Conference (DUC) (2000–2007) series (Over et al. 2007) and DUC's successor, the Text Analysis Conferences (TACs) (2008–now), have provided a set of annotated datasets for training and evaluating text summarization systems, focusing on the evaluation of generic and focused summaries of English newspaper and newswire articles. The DUC corpora (2000–2007) are popularly used to evaluate the vast majority of existing abstractive summarizers. However, as these corpora are relatively small, they do not provide enough data typically needed to train neural models.

Annotated English Gigaword Annotated English Gigaword is another popular corpus for abstractive summarization research. In comparison to the DUC corpora, Annotated English Gigaword is considerably larger. It contains nearly ten million documents (over four billion words) of the original English Gigaword Fifth Edition from various domestic and international news services over the last two decades, providing plentiful data needed to train neural models. To build a summarization dataset from Annotated English Gigaword, Rush et al. (2015) automatically create a source-summary pair from each article by using the first sentence of the article as the source and its headline as the summary.

CNN/Daily Mail For each story in the CNN/Daily Mail corpus, Nallapati et al. (2016) came up with a human summary. This corpus has 286,817 training pairs, 13,368 validation pairs and 11,487 test pairs, and has been widely used in many abstractive summarization tasks. Not only does this corpus provide plentiful training data, but it has two interesting aspects. First, in comparison to the documents in Gigaword and DUC, those in CNN/Daily Mail are much longer (781 tokens on average), thus yielding a comparatively more challenging summarization task. Second, unlike Gigaword, which has often been criticized for only having headlines as summaries, CNN/Daily Mail contains multi-sentence summaries (3.75 sentences or 56 tokens on average), and can therefore stimulate research on multi-sentence summary generation from long documents.

TAC 2010 The TAC 2010 summarization track pioneers the *guided summarization* task, where the goal is to create a 100-word summary of a set of 10 newswire articles for a given topic within a predefined category given a list of aspects relevant to each category. For example, Table 2 shows the aspects that a summary should address for the Attacks category. The documents released as part of the guided summarization task has been widely used for abstractive summarization (Genest and Lapalme 2012).

WHAT:	what happened
WHEN:	date, time, other temporal markers
WHERE:	physical location
PERPETRATORS:	persons/groups initiating the attack
WHY:	reasons for the attack
WHO_AFFECTED:	affected individuals
DAMAGES:	damages caused by the attack
COUNTERMEASURES:	rescue efforts, prevention efforts

Table 2: Aspects of TAC 2010's guided summarization task for the Attacks category.

AMI The AMI Meeting Corpus consists of 100 hours of meeting recordings and includes 139 multi-party meetings along with their corresponding extractive and abstractive summaries (Carletta et al. 2006). This corpus has sparked a lot of recent research on meeting summarization.

Other corpora Other notable corpora for summarization research that have been introduced over the years include TIPSTER (Mani et al. 2002), the Chinese corpus LCSTS (Hu et al. 2015), the IELTS summary corpus (Fang et al. 2016), a news corpus (Hu et al. 2015), a chat corpus (Zhou and Hovy 2005; Uthus and Aha 2013) and an email corpus (Ulrich et al. 2008).

4 Classical Approaches

In this section, we introduce classical approaches to abstractive summarization, using the term "classical" to broadly refer to any approach that is *not* neural-based.

Early Work

Early approaches to abstractive summarization include: (1) sentence compression (Cohn and Lapata 2009), which aims to create a grammatical summary of a given sentence; (2) sentence fusion (Barzilay and McKeown 2005; Filippova and Strube 2008), which involves using bottom-up local multi-sequence alignment to identify phrases conveying similar information and statistical generation to combine common phrases into a sentence; and (3) sentence revision (Tanaka et al. 2009), which generates sentences not found in the input and synthesizes information across sentences.

Fully Abstractive Summarization

The aforementioned approaches offer little improvement over extractive methods, however. This motivates the development of a fully abstractive approach, which typically contains three subtasks performed in a pipeline fashion: information extraction, content selection, and surface realization.

Information extraction Information extraction aims to extract important information from the input text. Many abstractive summarizers focus on extracting phrasal-level information such as noun phrases (NPs) and verb phrases (VPs) together with their contextual information (Genest and Lapalme 2012; Bing et al. 2015). Mehdad et al. (2014) employ *query-based extraction*, which aims to extract important contents using automatically generated queries and filter contents that have a low probability of being included in a summary. Genest and Lapalme (2012) extract Information Items (INITs), which they define as the smallest element of coherent information in a sentence. Concretely, an INIT is defined as a dated and located subject-verb-object triple.

Some domain-specific summarizers make use of knowledge of the category, topic, or domain of the input to guide the kind of information to be extracted (Wang and Cardie 2013). Recall from the previous section that in guided summarization, the aspects for a category (e.g., Attacks) are given. As a result, extraction rules can be designed based on abstraction schemas specific to a certain category to extract the desired information. For example, a killing schema requires that the killer, the verb that triggers the killing event, and the victim be extracted. In some cases, however, the input document covers multiple topics, which make manual pre-tagging of the document difficult. For example, in meeting transcript summarization, several topics may be mentioned during the meeting (Oya et al. 2014), in which case topic segmentation can be applied to identify the topics.

Content selection Content selection aims to select a subset of the candidate phrases extracted from the information extraction step for inclusion in the final summary, typically subject to length constraints. For instance, Genest and Lapalme's (2012) guided summarizer heuristically selects the candidate phrases most frequently mentioned for an aspect.

While heuristic methods can be used for content selection, many researchers have resorted to Integer Linear Programming (ILP) (Murray et al. 2010; Woodsend and Lapata 2011; Bing et al. 2015). As a constrained optimization framework, ILP can be used to optimize an objective function subject to a set of linear constraints. When applied to content selection, the objective function is a weighted sum of a set of binary variables. Each variable represents a candidate phrase and has the value 1 if and only if ILP decides to select it for inclusion in the final summary. The weight associated with each variable indicates the importance of the corresponding candidate phrase. Bing et al. (2015), for instance, estimate the salience of each candidate phrase based on its position and its grammatical role in the input document and use the salience score as its weight. The linear constraints encode length constraints. For instance, one constraint limits the number of words in each sentence in the summary.

The key advantage of employing ILP for content selection is that the decision of which phrases to include in the summary is made *jointly* and not *independently* of other phrases. This contrasts with non-optimization approaches, where such decisions are typically made in a heuristic, sequential, and therefore potentially suboptimal manner.

Surface realization Surface realization aims to combine the candidates selected in content selection using grammatical/syntactic rules to generate a summary. An existing natural language generator such as SimpleNLG (Gatt and Reiter 2009) can be adapted to generate the actual sentences.

Graph-based Methods

In graph-based methods, graphs are used to implement the aforementioned three abstractive summarization subtasks. Graphs are chosen because of their expressiveness: they facilitate the extraction of not only the concepts in an input document but also the potentially complex and abstract relations between them (Greenbacker 2011).

For example, *event semantic link networks* (ESLNs) have been used for joint information extraction and content selection (Li et al. 2016). Given an input text, an ESLN can be constructed to provide an abstract representation of the text. Specifically, each node corresponds to an event mentioned in the input text, where an event is composed of an event trigger/action and its arguments. An edge between two nodes encodes the semantic relation between the corresponding events. After network construction, ILP can be applied to this network to perform information extraction and content selection (i.e., selecting a subset of nodes for generating the summary), using constraints similar to Bing et al.'s (2015) (e.g., the length constraints) as well as constraints defined on the semantic relations (e.g., the nodes should be chosen such that the resulting graph remains connected).

As another example, *entailment graphs* can be used for content selection via detecting redundant sentences as follows (Mehdad et al. 2014). If two sentences have the same meaning (bidirectional entailment), one of them will be removed. If one of them is more informative than the other (unidirectional entailment), the less informative one will be removed. If both of them have some parts that do not overlap with the other, none of them will be removed.

The *word graph* method, which encodes the sentences into a graph, is usually used in generation (Filippova 2010). Briefly, a word graph is a weighted directed graph with words as nodes and is built by incrementally adding sentences to it. In this graph, words that share some sort of similarities are mapped onto the same existing node, and the summary is generated by selecting the best path in the graph. For instance, after constructing a word graph, Mehdad et al. (2014) propose a scoring function for ranking viable paths based on coverage, fluency, and path weight.

Template-based Methods

Template-based methods are motivated by the observation that human summaries of a given type (e.g., meeting summaries for accomplishing a certain task) have common sentence structures, which can be learned from the human summaries in the training set and encoded as *templates*. Given an input document, a summary can be generated by filling the slots in the best fitted templates learned for this type of documents. Template-based methods typically consist of three steps: (1) learning the templates from the human summaries; (2) extracting important phrases from the input document; and (3) generating a summary based on the filled templates.

For instance, Oya et al. (2014) propose a robust template-based method for meeting summarization. In step 1 (template learning), a template is first generated from a sentence of each human summary in the training set by replacing each NP in the sentence with a blank slot that is labeled with the hypernym of the NP's head using WordNet. Then, these templates are clustered based on their root verbs. Finally, the templates in each cluster are further generalized using the word graph method described in the previous subsection. In step 2 (keyphrase extraction), the important phrases for each topic segment of the input document are extracted and labeled with their hypernyms. Finally, in step 3 (generation), the templates having the highest similarity with each topic segment of the meeting are selected. As both the selected phrases and the most similar template have hypernym labels, candidate summary sentences can be generated by filling each template with matching labels. Since a potentially large number of sentences can be generated for each topic segment, a sentence ranker is trained to rank the generated sentences in each segment. The highest ranked sentence for each topic segment will be selected for inclusion in the summary. The selected sentences are sorted by the chronological order of the topic segments in the input document.

5 Neural Approaches

In classical methods to abstractive summarization, information extraction, content selection, and surface realization are all challenging subtasks. In contrast, neural methods offer an *end-to-end* approach to abstractive summarization, learning how to abstract from the source document and generate the corresponding summary in one network. Being the first to apply neural machine translation to abstractive summarization, Rush et al.'s (2015) work sparks a novel way of building abstractive summarizers. Since then, neural methods have become the core technology underlying the vast majority of abstractive summarizers. While it is fairly easy to keep track of the information that is being extracted and selected in classical models, there is comparatively less control over what is learned and how information is encoded in neural models. Below we introduce the key ideas that emerged from neural abstractive summarization research.

The Encoder-Decoder Framework

The vast majority of existing neural abstractive summarization models are sequence to sequence (seq2seq) models, which employ the encoder-decoder architecture (Sutskever et al. 2014). This architecture is composed of an encoder and a decoder. An encoder encodes source sentences as a list of fixed-length vector representations, each of which captures a word and its surrounding context. A decoder then outputs a summary based on the encoded vectors. The architecture is jointly trained on document-summary pairs to maximize the probability of a correct summary for each input document.

Encoding

Encoding has similar aims as information extraction in classical approaches: they both focus on capturing information relevant to summary generation. Encoding involves two key steps: (1) data preprocessing and (2) encoder selection, as described below.

Preprocessing To preprocess input sentences, many models use a word-based representation, but for some languages (e.g., Chinese), a character-based representation may be a better alternative as it can avoid errors introduced by word segmentation (Chang et al. 2018). Some use word vectors pre-trained on large corpora via word2vec (Mikolov et al. 2013) or GloVe (Pennington et al. 2014), while others learn the word embeddings during training (See et al. 2017).

Encoding *long documents* without losing important information is challenging. One way to address this problem is to compress a (long) input document into a more compact, informative representation by leveraging extractive methods to select representative sentences (Chen and Bansal 2018; Hsu et al. 2018; Lebanoff et al. 2018).

There have also been recent attempts to improve abstractive summarization by exploiting the *background knowledge* extracted from knowledge bases. For example, Amplayo et al. (2018) extract additional knowledge about the entities in the input document (e.g., the matches won by a football team) and subsequently use the resulting external knowledge to guide the decoder to generate better summaries.

Encoder selection Aiming to learn a better abstract representation of the input text and control the information flow from the encoder to the decoder, some researchers have focused on selecting or designing their encoders.

Rush et al. (2015) construct their encoder-decoder architecture with a convolutional neural network (CNN) as the encoder and a feed-forward neural network as the decoder. However, CNNs are typically replaced by recurrent neural networks (RNNs) in recent methods in part because CNNs lack the ability to process long sequences (Chopra et al. 2016; Nallapati et al. 2016). To address this problem, long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber 1997) are frequently applied instead (Nema et al. 2017; Pasunuru et al. 2017; Paulus et al. 2017; See et al. 2017; Tan et al. 2017). In some cases, GRUs (Cho et al. 2014) have been shown to be a better alternative to LSTMs (Chen et al. 2016; Kim et al. 2016; Li et al. 2017; Zhou et al. 2017), as they have fewer parameters and are faster to train while achieving comparable results (Chung et al. 2014; Greff et al. 2017).

Recent research on encoding has focused on designing complicated networks for leveraging existing information in *long* documents (Çelikyilmaz et al. 2018; Cohan et al. 2018). Nevertheless, how to encode long sequences remains an open question in seq2seq models.

Decoding

A decoder is commonly implemented using an RNN. At each timestep, the RNN takes as input two vectors, a representation of the previously generated words and a representation of the input sequence obtained via the encoding step, and produces a vector matching the size of the vocabulary, which is subsequently turned into a distribution over the vocabulary using a softmax layer. Given this distribution, either the

most probable word is generated as the output or, more commonly, the k-best paths up to this timestep are identified via a beam search, where k is the beam size (Rush et al. 2015; Chopra et al. 2016; Nallapati et al. 2016; Paulus et al. 2017; See et al. 2017).

Improvements to the Encoder-Decoder Framework

Numerous attempts have been made to improve the encoder-decoder framework for abstractive summarization.

Attention Some words/phrases are more important than others in a document. These important words/phrases are more likely to appear in a summary than their less important counterparts. To identify important words/phrases, one can employ *attention*.

The key idea behind attention is to feed the decoder with an extra input vector (known as the *context* vector) that encodes the important phrases (Bahdanau et al. 2014). At a high level, attention can first be used to compute a weight for each element in each timestep indicating its importance, and the resulting weight distribution over the elements can then be used to compute the context vector. Intuitively, the context vector amplifies the useful information from the input processed so far (i.e., information associated with high weights in the attention distribution) and de-emphasizes the unimportant information (i.e., information associated with low weights in the attention distribution).

Depending on whether we employ global (sentence-level) or local (word-level) attention, the resulting neural model has the ability to retrieve important information at different levels of a document (Luong et al. 2015; Nallapati et al. 2016; Tan et al. 2017; Çelikyilmaz et al. 2018; Cohan et al. 2018). Attention can also be applied to networks that do not employ the encoder-decoder framework (Vaswani et al. 2017).

Distraction/Coverage While attention enables us to identify and focus on important phrases, it is not without its problem. Researchers have observed that the same region/content could be overly focused, thereby leading to redundancy in the summary. *Distraction* can be used to avoid focusing on the same content (Nema et al. 2017). The idea is to employ a constraint that reduces the probability of the repeated content or the weight associated with that content.

Chen et al. (2016) show that distraction can be applied to the context vector, the attention weight vectors, and decoding, although the application of distraction is not limited to these three places. For example, in the training step, they implement distraction by subtracting the history context vector from the current context vector, effectively distracting the network from content that has been attended to previously.

Some researchers refer to distraction as *coverage* (See et al. 2017). Coverage is a concept originated in statistical machine translation (Koehn et al. 2007) and is subsequently used for neural machine translation by Tu et al. (2016). See et al. (2017) define a coverage loss which, when compared to the original loss, has an additional penalty term for repetition (i.e., more repetition implies less coverage).

Pointer networks/Copy mechanism Frequently occurring words are likely to be identified as important words by an attention mechanism. In contrast, it is commonly known that neural sequence models lack the ability to generate rare words and out-of-vocabulary (OOV) words, even if the generated context makes the prediction unambiguous.

To alleviate this problem, Vinyals et al. (2015) propose a pointer network, which copies an element from the input directly to the output. More generally, pointers can be seen as an extension of attention that allows us to focus on those rare or OOV words that are important.

If we use a pointer network to point to a *region* of the input rather than just a rare/OOV word, it is known as *copying*. In the context of abstractive summarization, the copy mechanism allows us to copy a segment of the input directly to the output (Gu et al. 2016; Paulus et al. 2017; See et al. 2017; Çelikyilmaz et al. 2018; Cohan et al. 2018). Specifically, the decoder is equipped with a "switch" that determines whether a generator or a pointer should be used at each timestep. If the switch is turned off, the decoder will generate a pointer to a word-position in the input sentence and copy the corresponding word to the summary (Nallapati et al. 2016).

Although the pointer/copy mechanism has proven useful for generating readable summaries, it leads to an obvious problem: the summaries may resemble those generated by extractive approaches, particularly when the decoder overuses the pointer. Therefore, in order to generate a summary that is more abstractive than extractive, one should control the extent to which the pointer is applied.

Other linguistic information can be used in conjunction with the copy mechanism to yield better summaries. For instance, when using the copy mechanism, Song et al. (2018) leverage syntactic structure, copying a word from the input to the summary if "it contains salient semantic content, or it serves a critical syntactic role in the source sentence". The syntactic label of each word, such as its part-of-speech tag and its depth in the associated dependency parse tree, is encoded by the encoder network.

Reinforcement learning The encoder-decoder framework has two weaknesses. First, while the network is trained to maximize the probability of generating a correct summary, the generated summary is evaluated by automatic metrics such as ROUGE. In other words, minimizing the maximum-likelihood (ML) loss is not necessarily equivalent to optimizing the desired evaluation metric. Second, while the decoder is trained on *gold* summaries, it decodes the next word by using the *generated* summary from the last timestep during test time. In other words, decoding performance can be adversely affected by this *exposure bias* (Ranzato et al. 2015), which stems from the fact that "the network has knowledge of the ground truth sequence up to the next token during training but does not have such supervision when testing" (Paulus et al. 2017).

To address these problems, researchers have recently leveraged Reinforcement Learning (RL) (Paulus et al. 2017; Pasunuru and Bansal 2018). RL enables us to train an agent to interact with a certain environment so as to maximize a reward. With the goal of finding an optimal policy (i.e., the best action to take for each state), RL can be used to solve optimization problems that are not differentiable. For

sequence generation, rather than minimizing the ML loss, we can maximize a reward based on the desired evaluation metric or even employ a hybrid training objective that not only minimizes the ML loss but also optimizes the desired metric (Paulus et al. 2017). This allows RL to make global (sentence-level) decisions rather than local (word-level) decisions during the generation process (Çelikyilmaz et al. 2018; Pasunuru and Bansal 2018): in each timestep, the model can be trained to select a word that maximizes (global) evaluation metrics such as ROUGE rather than making a decision based purely on the words that have been generated thus far. Note that RL also addresses the aforementioned exposure bias problem, as gold summaries do not need to be used at each step in the training process.

6 The State of the Art

Table 3 shows the ROUGE scores of state-of-the-art summarizers on the most frequent used evaluation corpus in recent years, the CNN/Daily Mail dataset. Several observations deserve mention. First, in comparison to an encoder-decoder RNN whose pointer components are trained to activate only for OOV words and named entities (row 1), a pointer-generator network that can freely learn when to use the pointer (row 2), especially when used in conjunction with coverage (row 3), yields better results. Second, comparing a model that employs RL (row 5) with one that uses a ML objective (row 4), RL helps to generate better summaries at the sentence level, achieving the highest ROUGE-L score. Third, there have been attempts to combine ML and RL: depending on how they are combined, the performance of the resulting system may increase (row 7) or decrease (row 6). Finally, the extractive summarizers (rows 8–10) slightly outperform almost all of the non-RL-based abstractive summarizers (rows 1–4). One reason is that extractive summarizers make sentence-level decisions, which enable the creation of summaries that are more readable than abstractive summaries. However, making sentence-level decisions also makes it difficult for them to retain all of the important contents if these contents are scattered throughout the input text. This could explain why extractive summarizers underperform the RL-based abstractive summarizers: RL can address an abstractive summarizer's weakness in making sentence-level decisions.

7 Concluding Remarks

Despite recent advances in neural abstractive summarization, state-of-the-art summarization results are still far from satisfactory. Producing an informative, fluent, and readable summary remains a difficult task. Below we discuss several avenues of research that we believe are worth pursuing.

Text simplification Encoding long sentences remains a challenge for neural approaches to abstractive summarization. One could leverage the techniques from text simplification to convert a complex sentence into simpler ones, which could be encoded more easily by neural models.

Phrase-based models Considering the fact that most of the classical summarizers are phrase-based, we believe that a phrase-based representation can capture the semantics of text more accurately for neural models. One can design a hybrid phrase-word representation that combines the advantages of word- and phrase-based representations (Chang et al. 2018). With a hybrid representation, while the vocabulary size can be larger, the decoder can generate a phrase in one step. Alternatively, we can enable a pointer to have a extra state for making phrase-level decisions. For example, a pointer can select more words at each step if two or more consecutive words have high probabilities.

Model	R_1	R_2	R_L
words-lvt2k-temp-att (Nallapati et al. 2016)	35.5	13.3	32.7
pointer-generator (See et al. 2017)	36.4	15.7	33.4
pointer-generator+coverage (See et al. 2017)	39.5	17.3	36.4
ML (Paulus et al. 2017)	38.3	14.8	35.5
RL (Paulus et al. 2017)	41.2	15.8	**39.1**
ML+RL (Paulus et al. 2017)	39.9	15.8	36.9
DCA ML+SEM+RL (Çelikyilmaz et al. 2018)	**41.7**	**19.5**	37.9
SummaRuNNer (Nallapati et al. 2017)	39.6	16.2	35.3
lead-3 (See et al. 2017)	40.3	17.7	36.8
REFRESH (Narayan et al. 2018)	40.0	18.2	36.6

Table 3: Empirical results of different summarizers on the CNN/Daily Mail dataset expressed in terms of ROUGE-1 (R_1), ROUGE-2 (R_2), and ROUGE-L (R_L).

Multi-document abstractive summarization Virtually all recent work on neural abstractive summarization has focused on summarizing a single document. Few systems are designed for multi-document abstractive summarization (Lebanoff et al. 2018; Liao et al. 2018). Note that we can treat neural multi-document abstractive summarization as a longer version of single-document summarization after preprocessing the input documents using classical multi-document summarization methods. Specifically, we can first remove redundant information from the set of input documents by (1) clustering the sentences in the input documents, (2) identifying the representative sentence in each of the top clusters, and (3) forming a single document using the resulting sentences and reordering them as needed. Then, we can apply the single-document abstractive summarization techniques discussed earlier to produce an abstractive summary of our artificially synthesized document.

Evaluation on different text types The vast majority of the neural models for abstractive summarization are evaluated on corpora composed of news articles because (1) news articles are well-organized and have predicable structures and (2) news corpora, which significantly outnumber other kinds of corpora, provide more data needed to train data-driven models. We believe it is worthwhile to investigate the applicability of neural models to corpora other than news, such as those composed of meetings and conversations.

Beyond seq2seq models While seq2seq models are extensively used for neural abstractive summarization, it is hard to interpret their results. For instance, it is hard to understand what exactly is being learned. This in turn can make it hard to generalize the resulting model to other datasets. In the long run, interpretable models for abstractive summarization should be investigated.

Extrinsic evaluation Whether commonly-used evaluation metrics such as ROUGE are sufficient for evaluating abstractive summaries is debatable, as the same content can be realized using many different words/phrases. Hence, in addition to performing an intrinsic evaluation, we may consider evaluating the correctness and usefulness of abstractive summaries in a downstream natural language application, such as question answering.

Acknowledgments

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of the paper. This work was supported in part by NSF Grant IIS-1528037.

References

Allahyari, M.; Pouriyeh, S. A.; Assefi, M.; Safaei, S.; Trippe, E. D.; Gutierrez, J. B.; and Kochut, K. 2017. Text summarization techniques: A brief survey. *International Journal of Advanced Computer Science and Applications*.

Amplayo, R. K.; Lim, S.; and Hwang, S.-W. 2018. Entity commonsense representation for neural abstractive summarization. *NAACL HLT*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473 [cs]*.

Barzilay, R., and McKeown, K. R. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*.

Bing, L.; Li, P.; Liao, Y.; Lam, W.; Guo, W.; and Passonneau, R. 2015. Abstractive multi-document summarization via phrase selection and merging. *ACL-IJCNLP*.

Carletta, J.; Ashby, S.; Bourban, S.; Flynn, M.; Guillemot, M.; Hain, T.; Kadlec, J.; Karaiskos, V.; Kraaij, W.; Kronenthal, M.; Lathoud, G.; Lincoln, M.; Lisowska, A.; McCowan, I.; Post, W.; Reidsma, D.; and Wellner, P. 2006. The AMI Meeting Corpus: A pre-announcement. *MLMI*.

Çelikyilmaz, A.; Bosselut, A.; He, X.; and Choi, Y. 2018. Deep communicating agents for abstractive summarization. *NAACL HLT*.

Chang, C.; Huang, C.; and Hsu, J. Y. 2018. A hybrid word-character model for abstractive summarization. *arXiv:1802.09968 [cs]*.

Chen, Y., and Bansal, M. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *ACL*.

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; and Jiang, H. 2016. Distraction-based neural networks for document summarization. *IJCAI*.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMNLP*.

Chopra, S.; Auli, M.; and Rush, A. M. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *NAACL HLT*.

Chung, J.; Gülçehre, Ç.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555 [cs]*.

Cohan, A.; Dernoncourt, F.; Kim, D. S.; Bui, T.; Kim, S.; Chang, W.; and Goharian, N. 2018. A discourse-aware attention model for abstractive summarization of long documents. *NAACL HLT*.

Cohn, T., and Lapata, M. 2008. Sentence compression beyond word deletion. *COLING*.

Cohn, T. A., and Lapata, M. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*.

Denkowski, M., and Lavie, A. 2014. Meteor Universal: Language specific translation evaluation for any target language. *WMT*.

Fang, Y.; Zhu, H.; Muszynska, E.; Kuhnle, A.; and Teufel, S. 2016. A proposition-based abstractive summariser. *COLING*.

Filippova, K., and Strube, M. 2008. Sentence fusion via dependency graph compression. *EMNLP*.

Filippova, K. 2010. Multi-sentence compression: Finding shortest paths in word graphs. *COLING*.

Gatt, A., and Reiter, E. 2009. SimpleNLG: A realisation engine for practical applications. *ENLG*.

Genest, P.-E., and Lapalme, G. 2012. Fully abstractive approach to guided summarization. *ACL*.

Greenbacker, C. 2011. Towards a framework for abstractive summarization of multimodal documents. *ACL Student Session*.

Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R.; and Schmidhuber, J. 2017. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems* 28(10).

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. K. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *ACL*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation*.

Hsu, W. T.; Lin, C.; Lee, M.; Min, K.; Tang, J.; and Sun, M. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. *ACL*.

Hu, B.; Chen, Q.; and Zhu, F. 2015. LCSTS: A large scale Chinese short text summarization dataset. *EMNLP*.

Kim, M.; Singh, M. D.; and Lee, M. 2016. Towards abstraction from extraction: Multiple timescale gated recurrent unit for summarization. *First Workshop on Learning Representations for NLP*.

Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N.; Cowan, B.; Shen, W.; Moran, C.; Zens, R.; et al. 2007. Moses: Open source toolkit for statistical machine translation. *ACL Interactive Poster and Demonstration Sessions*.

Lebanoff, L.; Song, K.; and Liu, F. 2018. Adapting the neural encoder-decoder framework from single to multi-document summarization. *EMNLP*.

Li, W.; He, L.; and Zhuge, H. 2016. Abstractive news summarization based on event semantic link network. *COLING*.

Li, P.; Lam, W.; Bing, L.; and Wang, Z. 2017. Deep recurrent generative decoder for abstractive text summarization. *EMNLP*.

Liao, K.; Lebanoff, L.; and Liu, F. 2018. Abstract meaning representation for multi-document summarization. *COLING*.

Lin, C.-Y. 2004. ROUGE: A package for automatic evaluation of summaries. *ACL Workshop on Text Summarization Branches Out*.

Lloret, E., and Palomar, M. 2012. Text summarisation in progress: A literature review. *Artificial Intelligence Review*.

Luong, M.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. *EMNLP*.

Mani, I.; Klein, G.; House, D.; Hirschman, L.; Firmin, T.; and Sundheim, B. 2002. SUMMAC: A text summarization evaluation. *Natural Language Engineering*.

Mehdad, Y.; Carenini, G.; and Ng, R. T. 2014. Abstractive summarization of spoken and written conversations based on phrasal queries. *ACL*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. *NIPS*.

Murray, G.; Carenini, G.; and Ng, R. 2010. Generating and validating abstracts of meeting conversations: A user study. *INLG*.

Nallapati, R.; Zhou, B.; dos Santos, C. N.; Gülçehre, Ç.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. *CoNLL*.

Nallapati, R.; Zhai, F.; and Zhou, B. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. *AAAI*.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Ranking sentences for extractive summarization with reinforcement learning. *NAACL HLT*.

Nema, P.; Khapra, M. M.; Laha, A.; and Ravindran, B. 2017. Diversity driven attention model for query-based abstractive summarization. *ACL*.

Nenkova, A., and McKeown, K. 2012. A survey of text summarization techniques. In *Mining Text Data*. Springer.

Nenkova, A.; Passonneau, R.; and McKeown, K. 2007. The Pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing*.

Over, P.; Dang, H.; and Harman, D. 2007. DUC in context. *Information Processing and Management*.

Oya, T.; Mehdad, Y.; Carenini, G.; and Ng, R. T. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. *INLG*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: A method for automatic evaluation of machine translation. *ACL*.

Pasunuru, R., and Bansal, M. 2018. Multi-reward reinforced summarization with saliency and entailment. *NAACL HLT*.

Pasunuru, R.; Guo, H.; and Bansal, M. 2017. Towards improving abstractive summarization via entailment generation. *EMNLP Workshop on New Frontiers in Summarization*.

Paulus, R.; Xiong, C.; and Socher, R. 2017. A deep reinforced model for abstractive summarization. *arXiv:1705.04304 [cs]*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. GloVe: Global vectors for word representation. *EMNLP*.

Radev, D. R.; Hovy, E.; and McKeown, K. 2002. Introduction to the special issue on summarization. *Computational Linguistics*.

Ranzato, M. A.; Chopra, S.; Auli, M.; Zaremba, W. 2015. Sequence level training with recurrent neural networks. *arXiv:1511.06732 [cs]*.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. *EMNLP*.

Saggion, H., and Poibeau, T. 2012. Automatic Text Summarization: Past, Present and Future. In *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing. Springer.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. *ACL*.

Song, K.; Zhao, L.; and Liu, F. 2018. Structure-infused copy mechanisms for abstractive summarization. *COLING*.

Spärck Jones, K. 2007. Automatic summarising: The state of the art. *Information Processing and Management*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *NIPS*.

Tan, J.; Wan, X.; and Xiao, J. 2017. Abstractive document summarization with a graph-based attentional neural model. *ACL*.

Tanaka, H.; Kinoshita, A.; Kobayakawa, T.; Kumano, T.; and Kato, N. 2009. Syntax-driven sentence revision for broadcast news summarization. *ACL-IJCNLP Workshop on Language Generation and Summarisation*.

Tu, Z.; Lu, Z.; Liu, Y.; Liu, X.; and Li, H. 2016. Modeling coverage for neural machine translation. *ACL*.

Ulrich, J.; Murray, G.; and Carenini, G. 2008. A publicly available annotated corpus for supervised email summarization. *AAAI EMAIL Workshop*.

Uthus, D. C., and Aha, D. W. 2013. The Ubuntu Chat Corpus for multiparticipant chat analysis. *AAAI Spring Symposium: Analyzing Microtext*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *NIPS*.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. *NIPS*.

Wang, L., and Cardie, C. 2013. Domain-independent abstract generation for focused meeting summarization. *ACL*.

Woodsend, K., and Lapata, M. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. *EMNLP*.

Zhou, L., and Hovy, E. H. 2005. Digesting virtual "geek" culture: The summarization of technical internet relay chats. *ACL*.

Zhou, Q.; Yang, N.; Wei, F.; and Zhou, M. 2017. Selective encoding for abstractive sentence summarization. *ACL*.