Content Selection in Deep Learning Models of Summarization

Anonymous EMNLP submission

Abstract

1 Introduction

Content selection is an important sub-task in many natural language generation tasks, where, given a generation goal, the system must determine which information needs to be expressed in output text (Gatt and Krahmer, 2018). In summarization, content selection usually is accomplished through sentence (and, less frequently, phrase) extraction. While it is a key component of both extractive and abstractive summarization systems, it is not generally well understood how deep learning models perform this task using only word and sentence embedding based representations**[[Hal: i'm not sure what this sentence means, especially the part after "using"]]**. Non-neural network approaches often use frequency and information theoretic measures as proxies for content salience (Hong and Nenkova, 2014), but these are not typically used **[[Hal: directly? explicitly?]]** in most recent work on summarization**[[Hal: maybe instead of recent work say nn systems?]]**.

[[Hal: there are lots of widows/orphans throughout that need to be cleaned up.]]

In this paper, we seek to better understand how deep learning models of summarization perform content selection across the news, personal stories, workplace meetings, and medical journal article domains.[1] We perform an analysis of several recent sentence extractive neural network architectures, specifically considering the design choices for sentence encoders and extractors. We compare Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) based sentence representations to the simpler approach of word embedding averaging to understand the gains derived from more sophisticated architectures. We also consider **[[Hal: use a diff word than consider]]** the necessity of auto-regressive sentence extraction, which the previous approaches have assumed, and propose two alternative models that extract sentences independently. Our main results reveal:

1. Sentence position bias dominates the learning signal for news summarization, though not for other domains. Summary content for news is only slightly degraded when content words are removed. **[[Hal: this sounds like you remove content words and the summary is still as good. clearly not what's meant. i hope :P.]]**

2. Word embedding averaging is as good or better than either RNNs or CNNs for sentence embedding across all domains.

3. Pre-trained word embeddings are as good, or better than, learned embeddings in most cases.**[[Hal: you were specific in the other ones, be specific here too if possible]]**

4. Non auto-regressive sentence extraction performs as good or better than auto-regressive extraction in all domains.

Taken together, these and other results in the paper suggest that we are overestimating the ability of deep learning models to learn robust and meaningful features for summarization. **[[Hal: i think there need to be stronger take-aways here. why do i care about these things?]]**

2 Related Work

Pre-neural network approaches to single document summarization are often formulated as graph-

[1] **Code and data release:** Upon publication, all code and pre-processing scripts will be released under an MIT (or more liberal) license; all data will be made available after publication when allowed by original licenses.

based ranking problems, where sentences are nodes in a graph and edges are determined by pairwise similarity of bag-of-words (BOW) representations (Erkan and Radev, 2004; Mihalcea and Tarau, 2005). More recently Wan (2010) jointly performed single and multi-document summarization in this framework. Generally, this line of work does not learn sentence representations for computing the underlying graph structures, which is the focus of this paper. A recent extension of this line of work is that of Yasunaga et al. (2017) who learn a graph-convolutional network for multi-document summarization. However, they do not extensively **[[Hal: can you drop extensively?]]** study the choice of sentence encoder, focusing more on the importance of the graph structure, which is orthogonal to this work.

To the extent that learning-based approaches have been applied to summarization prior to the use of neural networks, typically they have involved learning n-gram feature weights in linear models along with other non-lexical word or structure features (Berg-Kirkpatrick et al., 2011; Sipos et al., 2012; Durrett et al., 2016). In this paper, we study representation learning in highly non-linear neural networks that can capture more complex word level feature interactions and whose dense representations are more compatible with current practices in NLP.

The introduction of the CNN-DailyMail corpus by Hermann et al. (2015) **[[Hal: bad citation style, be sure these are all fixed. (i fixed this one and will fix others that i notice.)]]** allowed for the application of large-scale training of deep learning models for summarization. Cheng and Lapata (2016) developed a sentence extractive model that uses a word level CNN to encode sentences and a sentence level sequence-to-sequence model to predict which sentences to include in the summary. Nallapati et al. (2017) proposed a different model using word-level bidirectional recurrent neural networks (RNNs) along with a sentence level bidirectinoal RNN for predicting which sentences should be extracted. Additionally, the sentence extractor creates representations of the whole document and computes separate scores for salience, novelty, and location.

Since these two models are very different in design, it is unclear what model choices are most important for indentifying summary content in the input document. We use the sentence extractor designs of Cheng and Lapata (2016) and Nallapati et al. (2017) as points of comparison in our experiments (Section **??**). **[[Hal: can probably drop this paragraph for space]]**

All of the previous works **[[Hal: all previous works mentioned? or in the whole universe? :)]]** focused on news summarization. To further understand the content selection process, we also explore other domains of summarization. In particular, we explore personal narrative summarization based on stories shared on Reddit (Ouyang et al., 2017), workplace meeting summarization (Carletta et al., 2005), and medical journal article summarization (Mishra et al., 2014). While most work on these summarization tasks often exploit domain-specific features (e.g. speaker identification in meeting summarization (Gillick et al., 2009)), we purposefully avoid such features in this work in order to understand the extent to which deep learning models can learn useful content selection features.

[[Hal: since you're going tot alk about redundancy, i think you need to cite MMR.]]

As is also the case in other NLP tasks, it is not immediately obvious how a deep learning model is making its predictions, or what correlations are being exploited. There is a concerning and growing body of work that find models functioning as mere **[[Hal: why mere? nneighbor is good in a lot of settings!]]** nearest neighbors search (Chen et al., 2016), exploiting annotator artifacts (Gururangan et al., 2018), or open to fairly trivial adversarial exploitation (Jia and Liang, 2017). These lines of research are critical for finding model shortcomings, and over time, guiding improvements in technique. Unfortunately, to the best of our knowledge, there has been no such undertaking for the summarization task. **[[Hal: are we doing it? if not, perhaps this does belong here?]]**

3 Methods

The goal of extractive text summarization is to select a subset of a document's text to use as a summary, i.e. a short gist or excerpt of the central content. Typically, we impose a budget on the length of the summary in either words or bytes. In this work, we focus on *sentence* extractive summarization, where the basic unit of extraction is a sentence and impose a word limit as the budget.

We model the sentence extraction task as a sequence tagging problem, following (). **[[Hal:**

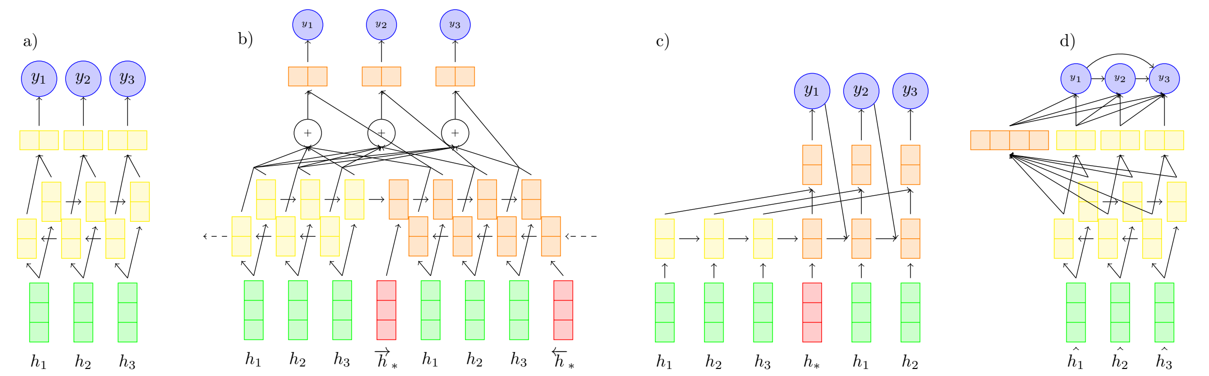

Figure 1: Sentence extractor architectures: a) RNN, b) Seq2Seq, c) Cheng & Lapata, and d) SummaRunner. The $\bigoplus$ indicates attention. Green repesents sentence encoder output, yellow and orange indicates extractor encoder and decoder hidden states respectively, and red indicates learned "begin decoding" embeddings.

missing cite]] Specifically, given a document containing n sentences $s_1, \ldots, s_n$ we generate a summary by predicting a corresponding label sequence $y_1, \ldots, y_n \in \{0, 1\}^n$, where $y_i = 1$ indicates the i-th sentence is to be included in the summary. Each sentence is itself a sequence of word embeddings $s_i = w_1^{(i)}, \ldots, w_{|s_i|}^{(i)}$ where $|s_i|$ is the length of the sentence in words. The word budget $c \in \mathbb{N}$ enforces a constraint that the total summary word length $\sum_{i=1}^{n} y_i \cdot |s_i| \leq c$.

For a typical deep learning model of extractive summarization there are two main design decisions: *a)* the choice of *sentence encoder* which maps each sentence s_i to an embedding h_i, and *b)* the choice of *sentence extractor* which maps a sequence of sentence embeddings $h = h_1, \ldots, h_n$ to a sequence of extraction decisions $y = y_1, \ldots, y_n$. The sentence extractor is then a discriminative classifier $p(y|h)$.

Previous neural network approaches to sentence extraction have assumed an auto-regressive model, leading to a semi-Markovian factorization of the extractor probabilities $p(y_{1:n}|h) = \prod_{i=1}^{n} p(y_i|y_{<i}, h)$, where each prediction y_i is dependent on *all* previous y_j for all $j < i$. We compare two such models proposed by Cheng and Lapata (2016) and Nallapati et al. (2017). **[[Hal: a simpler approach (explain why simpler) is a fully factored representation where we model $p(y_{1:n}|h) = \prod_{i=1}^{n} p(y_i|h)$, which we explrore etc]]**

3.1 Previously Proposed Sentence Extractors

Sentence extractors take hidden sentence representations $h_{1:n}$ and produce an extract $y_{1:n}$. We consider two recent state-of-the-art extractors. **[[Hal: if you're hurting for space, you could probably describe these both in one paragraph, leaving off the stuff about what they use as encoders, etc., and really just making the point that they use $y < i$ to predict yi]]**

Cheng & Lapata Extractor The extractor proposed by Cheng and Lapata (2016) is built around a sequence-to-sequence model. First, each sentence embedding[2] is fed into an encoder side RNN, with the final encoder state passed to the first step of the decoder RNN. On the decoder side, the same sentence embeddings are fed as input to the decoder, but are delayed by one step and weighted by their prediction probability, i.e. at decoder step t, $p(y_{t-1}|y_{<t-1}, h_{<t-1}) \cdot h_{t-1}$**[[Hal: why did you switch from i to t?]]** is fed into the decoder**[[Hal: i don't udnerstand what this means. what op is $\cdot$?]]**. The decoder output at step t is concatenated to the encoder output step t and fed through a multi-layer perceptron with one hidden layer and sigmoid unit output computing the t-th extraction probability $p(y_t|y_{<t}, h_{<t})$. See Figure 2.c. for a graphical view. Full model details are presented in ??.

SummaRunner Extractor Nallapati et al. (2017) proposed a sentence extractor, which we refer to as the SummaRunner Extractor, that factorizes the extraction probability into contributions from different sources. First, a bidirectional RNN is run over the sentence embeddings and the output concatenated. A document representation

[2] Cheng and Lapata (2016) used an CNN sentence encoder with this extractor architecture; in this work we pair the Cheng & Lapata extractor with several different encoders.

q is created by passing the averaged RNN output through a fully connected layer.**[[Hal: make the prev two sentences more brief.]]**[3] Given the RNN output z_t at the step t, the following scores are created:

1. a content score $W^{(con)} z_t$,
2. a salience score $z_t^T W^{(sal)} q$,
3. a novely score $-z_t^T W^{(nov)} \tanh(g_t)$,

where $g_t = \sum_{i=1}^{t-1} p(y_i = 1 | y_{<i}, h_{<i}) \cdot z_i$. These scores are summed along with a bias term and a bias for sentence position and the quarter of the document**[[Hal: what does "the quarter of the document" mean? sentence position quartile?]]** and fed through a sigmoid activation to compute $p(y_t = 1 | y_{<t}, h_{<t})$.

3.2 Proposed Sentence Extractors

We propose two sentence extractor models that make a stronger conditional independence assumption $p(y|h) = \prod_{i=1}^{n} p(y_i|h)$, essentially making independent predictions conditioned on h. In theory, our models should **[[Hal: why should they?]]** perform worse because of this, however, as we later show, this is not the case empirically.

RNN Extractor Our first proposed model is a very simple bidirectional RNN based tagging model. As in the RNN sentence encoder we use a GRU cell. The forward and backward outputs of each sentence are passed through a multi-layer perceptron with a logsitic sigmoid output to predict the probability of extracting each sentence. See details in **??**.

Seq2Seq Extractor One shortcoming of the RNN extractor is that long range information from one end of the document may not easily be able to effect extraction probabilities of sentences at the other end. Our second proposed model, the Seq2Seq extractor mitigates this problem with an attention mechanism commonly used for neural machine translation (Bahdanau et al., 2014) and abstractive summarization (See et al., 2017). The sentence embeddings are first encoded by a bidirectional GRU. A separate decoder GRU transforms each sentence into a query vector which attends to the encoder output. The attention weighted encoder output and the decoder GRU output are concatenated and fed into a multi-layer percepron to compute the extraction probability. Detail in **??**.

[3] Nallapati et al. (2017) use an RNN sentence encoder with this extractor architecture; in this work we pair the SummaRunner extractor with different encoders.

Dataset	Train	Valid	Test	Refs
CNN/DM	287,113	13,368	11,490	1
NYT	44,382	5,523	6,495	2
DUC	516	91	657	2
Reddit	404	24	48	2
AMI	98	19	20	1
PubMed	21,250	1,250	2,500	1

Table 1: Sizes of the training, validation, test splits for each dataset and the average number of human reference summaries per document.

3.3 Sentence Encoders

[[Hal: why not put this before extractors?]]

We experiment with three architectures for mapping sequences of word embeddings to a fixed length vector: averaging, RNNs, and CNNs. Hyperparameter settings and implementation details can be found in the appendix**[[Hal: include a ref to appendix section]]**

Averaging Encoder Under the averaging encoder, a sentence embedding h is simply the average of its word embeddings, i.e. $h = \frac{1}{|s|} \sum_{i=1}^{|s|} w_i$.

RNN Encoder When using the *RNN* sentence encoder, a sentenc embedding is the concatenation of the final output states of a forward and backward RNN over the sentence's word embeddings. We use a Gated Recurrent Unit (GRU) for the RNN cell (Chung et al., 2014).

CNN Encoder The *CNN* sentence encoder uses a series of convolutional feature maps to encode each sentence. This encoder is similar to the convolutional architecture of Kim (2014) used for text classification tasks and performs a series of "one-dimensional" convolutions over word embeddings. The final sentence embedding h is a concatenation of all the convolutional filter outputs after max pooling over time.

4 Datasets

We perform our experiments across six corpora from varying domains to understand how different biases within each domain can affect content selection. The corpora come from the news domain (CNN-DailyMail, New York Times, DUC), personal narratives domain (Reddit), workplace

Ext.	Enc.	CNN/DM		DUC 2002		Reddit		AMI		Δ-Best	
		M	R-2	M	R-2	M	R-2	M	R-2	M	R-2
Rand.	–	18.18	12.66	21.72	15.81	**20.43**	**12.90**	12.73	2.00	0.81	0.61
Lead	–	24.13	24.44	25.05	21.54	**20.05**	**10.88**	12.27	2.04	0.89	0.76
Tail	–	13.67	5.39	17.81	9.13	**19.82**	**9.95**	11.53	2.73	0.70	0.45
RNN	Avg.	**25.19**	25.42	**26.83**	22.65	**20.44**	**11.37**	**17.02**	**5.50**	0.99	0.93
	RNN	25.11	25.38	**26.79**	22.58	**20.17**	**11.43**	16.20	**5.24**	0.97	0.92
	CNN	24.98	25.07	**26.68**	**22.73**	**20.88**	**12.78**	14.37	3.16	0.95	0.85
Seq2Seq	Avg.	**25.18**	**25.56**	**26.95**	**22.84**	**20.88**	**13.61**	**16.99**	**5.52**	**1.00**	0.97
	RNN	**25.12**	25.33	**26.75**	22.47	**20.51**	**11.98**	16.11	**5.25**	0.98	0.93
	CNN	24.96	25.07	**26.73**	**22.67**	**20.72**	**13.21**	14.20	2.94	0.95	0.85
Cheng & Lapata	Avg.	25.04	25.28	**27.07**	**23.11**	**20.94**	**13.65**	**16.72**	**6.12**	0.99	**1.00**
	RNN	25.04	25.03	**26.99**	**23.03**	**20.27**	**12.62**	16.28	**4.96**	0.98	0.93
	CNN	**25.15**	25.13	**26.93**	**23.00**	**20.52**	**13.42**	14.27	2.81	0.95	0.86
Summa Runner	Avg.	25.12	25.44	26.69	22.28	**21.04**	**13.41**	**16.97**	**5.63**	1.00	0.97
	RNN	25.07	25.21	26.46	22.09	**20.89**	**12.51**	**16.46**	**5.38**	0.98	0.93
	CNN	24.90	25.00	26.37	22.22	**20.43**	**12.32**	14.54	3.16	0.95	0.84

Table 2: ROUGE 2 recall results across all sentence encoder/extractor pairs. All results are averaged over five random initializations. Results that are statistically indistinguishable from the best system are shown in bold face.

meetings (AMI), and medical journal articles (PubMed). See Table 1 for dataset statistics.

CNN-DailyMail We use the preprocessing and training, validation, and test splits of (See et al., 2017). This corpus is a mix of news on different topics including politics, sports, and entertainment.

New York Times The New York Times (NYT) corpus (Sandhaus, 2008) contains two types of abstracts for a subset of its articles. The first summary is an abstract produced by an archival librarian and the second is an online teaser meant to elicit a viewer on the webpage to click to read more. From this collection, we take all articles that have a combined summary length of at least 100 words We create training, validation, and test splits by partitioning on dates; we use the year 2005 as the validation data, with training and test partitions including documents before and after 2005 respectively.

DUC We use the single document summarization data from the 2001 and 2002 Document Understanding Conferences (DUC) (Over and Liggett, 2002). We split the 2001 data into training and validation splits and reserve the 2002 data for testing.

AMI The AMI corpus (Carletta et al., 2005) is a collection of real and staged office meetings annotated with text transcriptions, along with abstractive summaries. We use the prescribed splits.

Reddit Ouyang et al. (2017) collected a corpus of personal stories shared on Reddit[4] along with multiple extractive and abstractive summaries.

PubMed We created a corpus of 25,000 randomly samples medical journal articles from the PubMed Open Access Subset[5]. We only included articles if they were at least 1000 words long and had an abstract of at least 50 words in length. We used the article abstracts as the ground truth human summaries.

4.1 Ground Truth Extract Summaries

Since we do not typically have ground truth extract summaries from which to create the labels y_i, we construct gold label sequences by greedily optimizing ROUGE-1, as follows.

$I = \{s_1, \ldots, s_n\}; \quad y_i = 0 \quad \forall i \in 1, \ldots, n$
empty summary $S = \emptyset$; word budget c
while $\sum_{s_i \in S} |s_i| \leq c$ **do**
  $\hat{i} = \arg\max_{i \in \{j \in [n] | s_j \notin S\}} \text{ROUGE}(S \cup s_i)$
  if $\text{ROUGE}(S \cup s_{\hat{i}}) > \text{ROUGE}(S)$ **then**
    $S \leftarrow S \cup s_{\hat{i}}; \quad y_{\hat{i}} = 1$
  else
    break

[4] www.reddit.com
[5] https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

end if
end while
return $y = y_1, \ldots, y_n$

We choose to optimize for ROUGE-1 rather than ROUGE-2 similarly to other optimization based approaches to summarization (Durrett et al., 2016; Sipos et al., 2012; Nallapati et al., 2017) which found this be the easier optimization target.

5 Experiments

We train all models to minimize the weighted negative log likelihood

$$\mathcal{L} = -\sum_{s,y \in \mathcal{D}} \sum_{i=1}^{n} \omega(y_i) \log p\left(y_i | y_{<i}, \mathrm{enc}(s)\right)$$

over the training data $\mathcal{D}$ using stochastic gradient descent with the ADAM optimizer (Kingma and Ba, 2014). $\omega(0) = 1$ and $\omega(1) = N_0/N_1$ where N_y is the number of training examples with label y. We use a learning rate of .0001 and a dropout rate of .25 for all dropout layers. We also employ gradient clipping ($-5 < \nabla_\theta < 5$). We train for a maximum of 30 epochs and the best model is selected with early stopping on the validation set according to ROUGE-2. All experiments are repeated with five random initializations. Weight matrix parameters are initialized using Xavier initialization with the normal distribution (Glorot and Bengio, 2010) and bias terms are set to 0. Unless specified, word embeddings are initialized using pretrained GloVe embeddings (Pennington et al., 2014) and we do not update them during training. Unknown words are mapped to a zero embedding. We use a batch size of 32 for all datasets except AMI and PubMed, which are often longer and consume more memory, for which we use sizes two and four respectively. For the Cheng & Lapata model, we train for half of the maximum epochs with teacher forcing, i.e. we set $p_i = 1$ if $y_i = 1$ in the gold data and 0 otherwise when computing the decoder input $p_i \cdot h_i$; we revert to the true model probability during the second half training.

Extractor/Encoder Comparions In our main experiment, we compare our proposed sentence extractors, RNN and Seq2Seq, to those of Cheng & Lapata and SummaRunner. We test all possible sentence extractor/encoder pairs across all the datasets described in Section 4.

We evaluate summary quality using ROUGE-2 recall (Lin, 2004) as our main evaluation metric; ROUGE-1 and ROUGE-LCS trend similarity to ROUGE-2 in our experiments. We use target word lengths of 100 words for the news domains, and 75, 290, and 200 words for Reddit, AMI, and PubMed respectively. We also evaluate using METEOR (Denkowski and Lavie, 2014), which measures precision and recall of reference words, allowing for more complicated word matchings (e.g. via synonymy or morphology).

To perform statistical significance, we first average each document level score (either ROUGE-2 or METEOR) over the five random initializations to get one score per document per system per dataset. We then test the difference between the best system on each dataset and all other systems using the approximate randomization test (Riezler and Maxwell, 2005) with the Bonferroni correction for multiple comparisons, testing for significance at the $\alpha = .05$ level.

6 Results

The results of our main experiment comparing the different extractors/encoders are shown in Table 2. Overall, we find no major advantage when using the CNN and RNN sentence encoders. The best performing encoder/extractor pair uses the averaging encoder (4 out of 5 datasets) or the differences are not statistically significant. When only comparing within the same extractor choice, the averaging encoder is the better choice in 14 of 20 cases.

When looking at extractors, the Seq2Seq extractor is either part of the best performing system (2 out of 5 datasets) or is not statistically distinguishable from the best extractor.

Overall, on the news domain, the differences are quite small with the differences between worst and best systems on the CNN/DM dataset spanning only .56 of a ROUGE point. While there is more performance variability in the non-news domains, there is less distinction among systems: all systems are statistically indistinguishable from the best system on Reddit and every extractor has at least one configuration that is indistinguishable from the best system on the AMI corpus.**[[Hal: this is probably at least partially because of test set size. maybe mention this.]]**

Word Embedding Learning Given that learning a sentence encoder (averaging has no learned structure) does not yield significant improvement, it is natural to consider whether learning word em-

Extractor	Embeddings	CNN/DM R-2	NYT R-2	DUC 2002 R-2	Reddit R-2	AMI R-2
RNN	Fixed	**25.42**	**34.67**	**22.65**	**11.37**	**5.50**
	Learned	25.21	34.31	**22.60**	**11.30**	**5.30**
Seq2Seq	Fixed	**25.56**	**35.73**	**22.84**	**13.61**	5.52
	Learned	25.34	**35.65**	**22.88**	**13.78**	**5.79**
C&L	Fixed	**25.29**	**35.58**	**23.11**	**13.65**	**6.12**
	Learned	24.95	35.41	**22.98**	**13.39**	**6.16**
SummaRunner	Fixed	**25.44**	**35.40**	**22.28**	**13.41**	**5.63**
	Learned	25.11	35.20	**22.15**	12.64	**5.85**

Table 3: ROUGE-2 recall across sentence extractors when using learned or pretrained embeddings. In both cases embeddings are initialized with pretrained GloVe embeddings. All results are averaged from five random initializations. All extractors use the averaging sentence encoder. When both learned and unlearned settings are bolded, there is no signifcant performance difference.

Abl.	CNN/DM R-2	NYT R-2	DUC R-2	Reddit R-2	AMI R-2
all words	**25.42**	**34.67**	22.65	**11.37**	5.50
-nouns	25.31†	34.26†	22.35†	10.29†	3.76†
-verbs	25.25†	34.36†	22.45†	10.76	5.80
-adjv	25.28†	34.36†	22.50	9.48†	5.36
-func	25.24†	34.55†	**22.89**†	10.26†	**6.32**†

Table 4: ROUGE-2 recall after removing nouns, verbs, adjectives/adverbs, and function words. Ablations are performed using the averaging sentence encoder and the RNN extractor. Table shows average results of five random initializations. Bold indicates best performing system. † indicates significant difference with the non-ablated system.

beddings is necessary either. In Table 3 we compare the performance of different extractors using the averaging encoder, when the word embeddings are held fixed or learned during training. In both cases, word embeddings are initialized with GloVe embeddings. When learning embeddings, words occurring fewer than three times in the training data are mapped to a learned unknown token.

In all but one case, fixed embeddings are as good or better than the learned embeddings. This is a somewhat surprising finding on the CNN/DM data since it is reasonably large by most standards and learning embeddings should give the models more flexibility to identify important word features. This suggests that we cannot extract much generalizable learning signal from the content other than what is already present from initialization.

The AMI corpus is an exception here where learning does lead to small performance boosts, however, only in the Seq2Seq extractor is this diference significant. The language of this corpus is quite different from the data that the GloVe embeddings were trained on and so it makes sense that there would be more benefit to learning word representations; one explanation for only seeing modest improvements is purely the small size of the dataset (NOTE TO CK – expect learning to help on pubmed). **[[Hal: yes, the dataset size is certainly an issue here. probably worth pointing this out. also when you learned the embeddings, did you initialize to pretrained embeddings? did you regularize toward them?]]**

POS Tag Ablation It is also not well explored what word features are being used by the encoders. To understand which classes of words were most important we ran an ablation study, selectively removing nouns, verbs (including participles and auxiliaries), adjectives & adverbs, and function words (adpositions, determiners, conjunctions). All datasets were automatically tagged using the spaCy part-of-speech (POS) tagger[6]. The em-

[6] https://github.com/explosion/spaCy

Extractor	Sentence Order	CNN/DM R-2	NYT R-2	DUC 2002 R-2	Reddit R-2	AMI R-2
RNN	In-Order	**25.42**	**34.67**	**22.65**	**11.37**	**5.50**
	Shuffled	22.80	24.96	18.24	**11.83**	**5.70**
Seq2Seq	In-Order	**25.56**	**35.73**	**22.84**	**13.61**	5.52
	Shuffled	21.66	25.61	21.21	**13.45**	**5.98**

Table 5: ROUGE-2 recall using models trained on in-order and shuffled documents. All extractors use the averaging sentence encoder. Table shows average results of five random initializations. When both in-order and shuffled settings are bolded, there is no signifcant performance difference.

beddings of removed words were replaced with a zero vector, preserving the order and position of the non-ablated words in the sentence. Ablations were performed on training, validation, and test partitions, using the RNN extractor with averaging encoder. Table 4 shows the results of the POS tag ablation experiments. While removing any word class from the representation generally hurts performance (with statistical significance), on the news domains, the absolute values of the differences are quite small (.18 on CNN/DM, .41 on NYT, .3 on DUC) suggesting that the model's predictions are not overly dependent on any particular word types.**[[Hal: i'm not sure i agree. if it's significant it's significant, no?]]** On the non-news datasets, the ablations have a larger effect (max differences are 1.89 on Reddit, 2.56 on AMI). **[[Hal: if you're mentioning absolute differences here i'd do that in the tables too]]** Removing the content words leads to the largest drop on AMI. Removing adjectives and adverbs leads to the largest drop on Reddit, suggesting the intensifiers and descriptive words are useful for identifying important content in personal narratives. Curiously, removing the miscellaneous POS class yields a significant improvement on DUC 2002 and AMI.

Document Shuffling Sentence position is a well known and powerful feature for news summarization (Hong and Nenkova, 2014), owing from the intentional lead bias in the news article writing[7]; it also explains the difficulty in beating the lead baseline for single-document summarization (Nenkova, 2005). In examining the generated summaries, we found most of the selected sentences in the news domain came from the lead paragraph**[[Hal: i feel like there must be citations to dig up here from like the 90s about lead summarization in news... it's also an intentional bias: maybe the right thing is to cite a style guide from a newsppaer that says to write this way]]** of the document. This is despite the fact that there is a long tail of sentence extractions from later in the document in the ground truth extract summaries (31%, 28.3%, and 11.4% of DUC, CNN/DM, and NYT training extract labels come from the back half of the document). Because this lead bias is so strong, it is questionable whether the models are learning to identify important content or just find the start of the document. We conduct a sentence order experiment where each document's sentences are randomly shuffled during training. We then evaluate each model performance on the unshuffled test data, comparing to the model trained on unshuffled data; if the models trained on shuffled data drop in performance, then this indicates the lead bias is relevant factor in learning content selection.

Table 5 shows the results of the shuffling experiments. The news domain suffers a significant drop in performance when the document order is shuffled. By comparison, there is no significant difference between the shuffled and in-order models on the Reddit domain, and shuffling actually improves performance on the AMI. is being learned by the models in the news domain even when the model has no explicit position features, and that this feature is more important than either content or function words.

7 Discussion

Learning in the news domain is severely inhibited by the lead bias. The output of all systems is highly similar to each other and to the lead baseline. Using the averaging encoder, the Seq2Seq and Cheng & Lapata extractors share 87.8% of output sentences on average on the CNN/DM data, with similar numbers for the other news domains

[7] link to ap style guide on inverted pyramid

			cnn-dailymail	nyt	duc-sds	reddit	ami
			R2	R2	R2	R2	R2
Source Dataset	Extractor	Doc Order					
cnn-dailymail	RNN	In-Order	25.4172	32.6914	22.7538	14.5222	4.0718
nyt	RNN	In-Order	24.1500	34.6666	22.9564	11.2408	4.2336
duc-sds	RNN	In-Order	23.5106	32.2476	22.6520	12.3368	3.0092
reddit	RNN	In-Order	23.0648	27.9204	20.6258	11.3940	2.5646
ami	RNN	In-Order	9.6806	7.7956	12.6174	10.7048	5.4958
cnn-dailymail	RNN	Shuffled	22.8038	25.5584	21.3410	**14.7008**	3.4252
nyt	RNN	Shuffled	17.5148	24.9578	20.6572	10.4836	3.8924
duc-sds	RNN	Shuffled	16.9730	18.6806	18.2378	11.7118	3.2894
reddit	RNN	Shuffled	22.7654	27.7366	20.5632	11.8348	2.6030
ami	RNN	Shuffled	9.9302	8.2276	12.7670	10.2996	5.7002
cnn-dailymail	Seq2Seq	In-Order	**25.5560**	32.5312	22.7042	14.3916	3.8886
nyt	Seq2Seq	In-Order	24.4888	**35.7280**	**23.0692**	10.7674	3.5056
duc-sds	Seq2Seq	In-Order	24.3412	32.7244	22.8384	12.9758	3.5844
reddit	Seq2Seq	In-Order	18.7552	23.7096	19.8706	13.7194	3.0072
ami	Seq2Seq	In-Order	12.1790	9.7512	13.7606	9.9028	5.5226
cnn-dailymail	Seq2Seq	Shuffled	21.6618	22.6544	19.9770	14.0084	3.8104
nyt	Seq2Seq	Shuffled	18.2358	25.6094	20.5528	12.0216	3.6208
duc-sds	Seq2Seq	Shuffled	19.6998	25.5266	21.2116	12.0884	3.4744
reddit	Seq2Seq	Shuffled	19.4136	23.8588	19.8218	13.4550	2.8182
ami	Seq2Seq	Shuffled	13.5160	13.3164	15.6102	10.2888	**5.9806**

Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo. **On Saturday, Hurricane Florence was downgraded to a tropical storm and its remnants pushed inland from the U.S. Gulf Coast.** Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night.	Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph. An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo. Tropical Storm Gilbert formed in the eastern Caribbean and strengthened into a hurricane Saturday night. **Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds and up to 12 feet feet to Puerto Rico's south coast.**

Table 6: Example output of Seq2Seq extractor (left) and Cheng & Lapata Extractor (right). This is a typical example, where only one sentence is different between the two (show in bold).

(see Table 6 for a typical example). Also on CNN/DM, 58% of the Seq2Seq with averaging encoder outputs sentences also occurring in the lead summary, with similar numbers for DUC, NYT, and Reddit. Shuffling reduces lead overlap to 35.2% but the overall system performance drops significantly. The relative robustness of the news domain to POS ablation also suggests suggests that models are mostly learning to recognize the stylistic features unique to the beggining of the article, and not the content. Additionally, drop in performance when learning word embeddings on the news domain suggests that word embeddings alone do not provide very generalizable content features compared to recognizing the lead.

The picture is rosier for non-news summarization where POS ablation leads to larger performance differences and shuffling either does not inhibit content selection significantly or leads to modest gains. In order to learn better word-level representations on these domains will likely require much larger corpora, somthing which might remain unlikely for personal narratives and meetings.

The lack of distinction amongst sentence encoders is interesting because it echoes findings in the generic sentence embedding literature where word embedding averaging is frustratingly difficult to outperform (Wieting et al., 2015; Arora et al., 2016; Wieting and Gimpel, 2017). The inability to learn useful sentence representations is also borne out in the SummaRunner model, where there are explicit similarity computations between document or summary representations and sentence embeddings; these computations do not seem to add much to the performance as the Cheng & Lapata and Seq2Seq models which lack these features generally perform as well or better. Furthermore, the Cheng & Lapata and SummaRunner extractors both construct a history of previous selection decisions to inform future choices but this does not seem to significantly improve performance over the Seq2Seq extractor which does not, suggesting that we need to rethink or find novel forms of sentence representation for the summarization task.

A manual examination of the outputs revealed some interesting failure modes although in general it was hard to discern clear patterns of behaviour other than lead bias. On the news domain, the models consistently learned to ignore quoted material in the lead, as often the quotes provide color to the story but are unlikely to be included in the summary (e.g.*"It was like somebody slugging a punching bag."*). This behavior was most likely triggered by the presence of quotes, as the quote attributions, which were often tokenized as separate sentences, would subsequently be included in the summary despite also not containing much information (e.g. *Gil Clark of the National Hurricane Center said Thursday*).

The Reddit corpus was particularly challenging as often there was no concise way to extractively represent the story. Authors often inserted a fairly brief summary of the story, either at the end or beginning, but this was so abstractive as to not have much overlap with the reference summaries. For example, the first sentence, *I took a dog off the street, and she changed my grandparent's lives*, has little overlap with the reference *The dog I found for my grandparents still gets really excited to see me whenever I come to visit*, but is still functionally a reasonable summary of the story. These "micro summaries" do not seem to be consistently found by any summarization model.

8 Conclusion

We have presented an empirical study of deep learning based content selection algorithms for summarization. Our findings suggest more work is needed on sentence representation.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

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

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. 2005. The ami meeting corpus: A pre-announcement. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 28–39. Springer.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

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

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *journal of artificial intelligence research*, 22:457–479.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Dan Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tur. 2009. A global optimization framework for meeting summarization. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4769–4772. IEEE.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data.

Karl Moritz Hermann, Tomáš Kočiský, 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 (NIPS)*.

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

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Rada Mihalcea and Paul Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts*.

Rashmi Mishra, Jiantao Bian, Marcelo Fiszman, Charlene R Weir, Siddhartha Jonnalagadda, Javed Mostafa, and Guilherme Del Fiol. 2014. Text summarization in the biomedical domain: a systematic review of recent research. *Journal of biomedical informatics*, 52:457–467.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Ani Nenkova. 2005. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *AAAI*, volume 5, pages 1436–1441.

Jessica Ouyang, Serina Chang, and Kathy McKeown. 2017. Crowd-sourced iterative annotation for narrative summarization corpora. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 46–51.

Paul Over and Walter Liggett. 2002. Introduction to duc: An intrinsic evaluation of generic news text summarization systems. *Proc. DUC. http://wwwnlpir. nist. gov/projects/duc/guidelines/2002. html*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Stefan Riezler and John T Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 57–64.

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

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin learning of submodular summarization models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 224–233. Association for Computational Linguistics.

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1137–1145. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. *arXiv preprint arXiv:1705.00364*.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. *arXiv preprint arXiv:1706.06681*.

Supplementary Material For: Content Selection in Deep Learning Models of Summarization

A Details on RNN Encoder

Under the *RNN* encoder, a sentence embedding is defined as $h = [\overrightarrow{h}_{|s|}; \overleftarrow{h}_1]$ where

$$\overrightarrow{h}_0 = \mathbf{0}; \quad \overrightarrow{h}_i = \overrightarrow{\mathrm{GRU}}(w_i, \overrightarrow{h}_{i-1}) \tag{1}$$

$$\overleftarrow{h}_{|s|+1} = \mathbf{0}; \quad \overleftarrow{h}_i = \overleftarrow{\mathrm{GRU}}(w_i, \overleftarrow{h}_{i+1}), \tag{2}$$

and $\overrightarrow{\mathrm{GRU}}$ amd $\overleftarrow{\mathrm{GRU}}$ indicate the forward and backward GRUs respectively, each with separate parameters.

B Details on CNN Encoder

The *CNN* encoder has hyperparameters associated with the window sizes $K \subset \mathcal{N}$ of the convolutional filter (i.e. the number of words associated with each convolution) and the number of feature maps M associated with each filter (i.e. the output dimension of each convolution). The *CNN* sentence embedding h is computed as follows:

$$a_i^{(m,k)} = b^{(m,k)} + \sum_{j=1}^{k} W_j^{(m,k)} \cdot w_{i+j-1} \tag{3}$$

$$h^{(m,k)} = \max_{i \in 1,\ldots,|s|-k+1} \mathrm{ReLU}\left(a_i^{(m,k)}\right) \tag{4}$$

$$h = \left[h^{(m,k)} | m \in \{1, \ldots, M\}, k \in K\right] \tag{5}$$

where $b^{(m,k)} \in \mathcal{R}$ and $W^{(m,k)} \in \mathcal{R}^{k \times n'}$ are learned bias and filter weight parameters respectively, and $\mathrm{ReLU}(x) = \max(0, x)$ is the rectified linear unit activation.