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 (**?**). 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, **[[Hal: well understood as a task or in specific models?]]** with frequency and information theoretic measures used as proxies for content salience (**?**).

In this paper, we seek to better understand how deep learning models of summarization perform content selection across a variety of domains. We perform an analysis of several recent sentence extractive neural network architectures, specifically considering the design choices for sentence encoders and extractors. We compare these architectures against a simpler approach based on averaging of word embeddings in order to understand the gains derived from more sophisticated architectures. Our main results reveal:

1. Sentence position bias dominates the learning signal for news summarization, though not for other genres**[[Hal: be consistent on domain/genre. also maybe you should say above what domains you consider?]]**. Summary content for news is only slightly degraded when content words are removed.

2. Word embedding averaging is as good or better than either Recurrent Neural Net (RNN) or Convolutional Neural Net (CNN) encoders.**[[Hal: across all domains?]]**

3. Pre-trained word embeddings are as good, or better than, learned embeddings in most cases.

4. Representation of previously selected summary content does not improve overall summary content. **[[Hal: i think this one might be hard to parse for non-experts. maybe "explicitly reprsenting prev..."]]**

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.

In this paper, we concentrate on sentence extraction approaches to summarization, but believe that our findings are relevant to the abstractive summarization approaches as well. The objective criteria for abstractive and extractive summarization and the the same: produce output text with high word overlap to a reference human abstract. Moreover, most neural network-based abstractive summarization techniques are architecturally quite similar to the extractive models we consider.

[[Kathy: I'm not sure these contributions are needed. Others should weigh in here.]] **[[Hal: agreed. i think the first is already covered in the list above, and the second can probably be worked in to the second paragraph.]]** The contributions of this paper are the following:

1. An empirical study of extractive content selection in deep learning algorithms for text summarization across news, personal narratives, meetings, and medical journal domains revealing difficulties in learning useful sentence representations.

2. Two simple sentence extractor models whose performance is on par with more complex prior work.

2 Related Work

[[Hal: I think this section can be organized better. I don't have a solid proposal, but i might be tempted to go reverse chronological, and start with the recent stuff and then ground it in what came before. but generally i feel like this is basically a bulletted list and it needs some coherence/cohesion.]]

Pre-neural network approaches to single document summarization are often formulated as graph-based ranking problems, where sentences are nodes in a graph and edges are determined by pairwise similarity of bag-of-words (BOW) representations (**??**). More recently **?** 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.

An notable exception is that of **?** who learn a graph-convolutional network for multi-document summarization. However, they do not 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 (**???**). 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 (**?**) allowed for the application of large-scale training of deep learning models for summarization. **?** 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. **?** 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 (**?**) and (**?**) as points of comparison in our experiments (Section **??**).

All of the previous works focused on news summarization. To further understand the content selection process, we also explore other domains of summarization. In particular, we explore summarization of personal narratives shared on the website Reddit (**?**), workplace meeting summarization (**?**), and medical journal article summarization (**?**). While most work on these summarization tasks often exploit features relevant to the domain, e.g. speaker identification in meeting summarization (**?**), 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 list of papers that find models functioning as mere nearest neighbors search (**?**), exploiting annotator artifacts (**?**), or open to fairly trivial adversarial exploitation (**?**). 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.

3 Problem Definition

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. **[[Hal: and impose a word limit as the budget, or something]]**

We evaluate summary quality using ROUGE (**?**) which measures the n-gram overlap of a hypothesis summary with one or more human gold-standard summaries. We use ROUGE-2 recall (i.e. bigram recall) as our main evaluation metric; ROUGE-1 (unigram recall) and ROUGE-LCS (longest common substring) trend similarity to ROUGE-2 in our experiments. We also evaluate using METEOR (?), which measures precision and recall of reference words, allowing for more complicated word matchings (e.g. via synonymy or morphology).

In this paper, we model sentence extraction task as a sequence tagging problem, following () **[[Hal: need cites here]]**. Specifically, given a document containing d sentences $s_1, \ldots, s_d$ we generate a summary by predict a corresponding label sequence $y_1, \ldots, y_d \in \{0, 1\}^d$, where $y_i = 1$ indicates the i-th sentence is to be included in the summary. The word budget enforces a constraint that **[[Hal: fill in]]**. **[[Hal: maybe introduce some more notation here, like s is a sequence of ws, etc. i think that might be all you need.]]**

4 Models and Methods

[[Kathy: This section feels very long. Do you think all the formulas are needed? They take up a lot of space. Would be good for Hal and Chris H to see this comment.]] **[[Hal: agree. this section is about 3 pages long. i would suggest moving low-level details to an appendix which can be submitted as supplemental. i've added toappendix.sty to make this easy. i've given an example usage in the RNN encoder section which i think you can use throughout. i also added a makefile that will split the paper into main and supllemental]]**

For a typical deep learning model of extractive summarization there are two main design decisions: *i)* the choice of *sentence encoder* which maps each sentence s_i to an embedding $h_i \in \mathcal{R}^d$, and *ii)* 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 $p(y|h)$.

We study three architectures for the sentence encoders, namely, embedding averaging, RNNs, and CNNs. We also propose two simple models for the sentence extractor and compare to the previously proposed extractors of **?** and **?**. **[[Hal: i think it's still confusing what's new and what's not. maybe you can somewhat mark? like things with $\star$ are new and ones without are old or something?]]** The prior works differ significantly but make the same semi-Markovian factorization of the extraction decisions, i.e. $p(y|h) = \prod_{i=1}^{n} p(y_i|y_{<i}, h)$, where each prediction y_i is dependent on all previous y_j for all $j < i$. By contrast, our extractors 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 perform worse because of this, however, as we later show, this is not the case empirically.

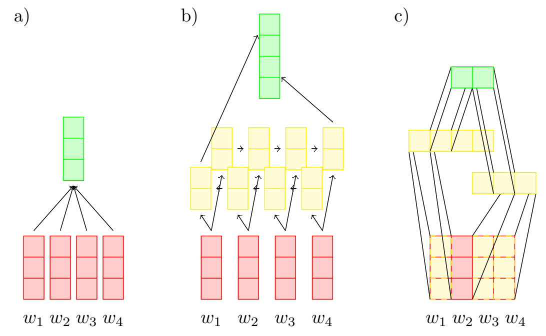

Figure 1: Sentence encoder architectures: a) averaging encoder, b) RNN encoder c) CNN encoder. Red indicates word embeddings, yellow indicates RNN hidden states or convolutional activations, and green indicates the sentence embedding that is passed to the extractor module.

[[Hal: i think you might need a subsection at the end of this section with oen or two paragraphs of compare/contrast the different models, esp if details are going to appendix]]

4.1 Sentence Encoders

We treat each sentence $s = w_1, \ldots, w_{|s|}$ as a sequence of word embeddings, where $w_i \in \mathcal{R}^{n'}$ are word embeddings and $|s|$ is the total number of words in the sentence. We experiment with three architectures for mapping sequences of word embeddings to a fixed length vector: averaging, RNNs, and CNNs. See Figure 1 for a diagram of the three encoders.

Averaging Encoder The averaging encoder (*AVG*) is the simplest method and has the added benfit of being parameter free. Under the averaging encoder, a sentence embedding $h = \frac{1}{|s|}\sum_{i=1}^{|s|} w_i$ is simply the average of its word embeddings. **[[Hal: get rid of orphans.]]**

RNN Encoder The *RNN* sentence encoder uses 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, since it has fewer parameters than the equivalent LSTM but with similar performance (?). Details in Appendix A.

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 (?) used for text classification tasks and performs a series of "one-dimensional" convolutions over word embeddings. The final sentence embedding h is **[[Hal: ???]]**; detail in Appendix B.

4.2 Sentence Extractors

Given a sequence of sentence embeddings $h = h_1, \ldots, h_n$ produced by a sentence encoder, a sentence extractor defines a conditional distribution $p(y|h)$ over the corresponding sentence extraction variables.

We first propose two simple extractor models based on bidirectional RNN and sequence-to-sequence with attention architectures, which we refer to as the RNN and Seq2Seq extractors respectively.

Next we define the extractor models of Cheng & Lapata, and Nallapati et al., whose models we refer to as the Cheng & Lapata and SummaRunner extractors respectively. See Figure 2 for a diagram of the four sentence extractor architectures.

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 (denoted by σ) to predict the probability of extracting each sentence:

$$\overrightarrow{z}_0 = \mathbf{0}; \quad \overrightarrow{z}_i = \overrightarrow{\mathrm{GRU}}(h_i, \overrightarrow{z}_{i-1}) \tag{1}$$
$$\overleftarrow{z}_{n+1} = \mathbf{0}; \quad \overleftarrow{z}_i = \overleftarrow{\mathrm{GRU}}(h_i, \overleftarrow{z}_{i+1}) \tag{2}$$
$$a_i = \mathrm{ReLU}\left(U \cdot [\overrightarrow{z}_i; \overleftarrow{z}_i] + u\right) \tag{3}$$
$$p(y_i = 1|h) = \sigma\left(V \cdot a_i + v\right) \tag{4}$$

where $\overrightarrow{\mathrm{GRU}}$ and $\overleftarrow{\mathrm{GRU}}$ indicate the forward and backward GRUs respectively, and each have separate learned parameters; U, V and u, v are learned weight and bias parameters.

[[Hal: use paragraph not textbf]]

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 (?) and abstractive summarization (?). 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 multilayer perceptron to compute the extraction probability. Formally we have:

$$\overrightarrow{z}_0 = \mathbf{0}; \quad \overrightarrow{z}_i = \overrightarrow{\mathrm{GRU}}_{enc}(h_i, \overrightarrow{z}_{i-1}) \tag{5}$$
$$\overleftarrow{z}_{n+1} = \mathbf{0}; \quad \overleftarrow{z}_i = \overleftarrow{\mathrm{GRU}}_{enc}(h_i, \overleftarrow{z}_{i+1}) \tag{6}$$
$$\overrightarrow{q}_i = \overrightarrow{\mathrm{GRU}}_{dec}(h_i, \overrightarrow{q}_{i-1}) \tag{7}$$
$$\overleftarrow{q}_i = \overleftarrow{\mathrm{GRU}}_{dec}(h_i, \overleftarrow{q}_{i+1}) \tag{8}$$

$$q_i = [\overrightarrow{q}_i; \overleftarrow{q}_i], \;\; z_i = [\overrightarrow{z}_i; \overleftarrow{z}_i] \tag{9}$$

$$\alpha_{i,j} = \frac{\exp\left(q_i \cdot z_j\right)}{\sum_{j=1}^{n} \exp\left(q_i \cdot z_j\right)}, \;\; \bar{z}_i = \sum_{j=1}^{n} \alpha_{i,j} z_j \tag{10}$$

$$a_i = \mathrm{ReLU}\left(U \cdot [\bar{z}_i; q_i] + u\right) \tag{11}$$
$$p(y_i = 1|h) = \sigma\left(V \cdot a_i + v\right). \tag{12}$$

The final outputs of each encoder direction are passed to the first decoder steps; additionally, the first step of the decoder GRUs are learned "begin decoding" vectors $\overrightarrow{q}_0$ and $\overleftarrow{q}_0$ (see Figure 2.b). Each GRU has separate learned parameters; U, V and u, v are learned weight and bias parameters.

Cheng & Lapata Extractor We compare the previously proposed architectures to the sentence extractor model of (?). The basic architecture is a unidirectional sequence-to-sequence model defined as follows:

$$z_0 = \mathbf{0}; \quad z_i = \mathrm{GRU}_{enc}(h_i, z_{i-1}) \tag{13}$$
$$q_1 = \mathrm{GRU}_{dec}(h_*, z_n) \tag{14}$$
$$q_i = \mathrm{GRU}_{dec}(p_{i-1} \cdot h_{i-1}, q_{i-1}) \tag{15}$$
$$a_i = \mathrm{ReLU}\left(U \cdot [z_i; q_i] + u\right) \tag{16}$$
$$p_i = p(y_i = 1|y_{<i}, h) = \sigma\left(V \cdot a_i + v\right) \tag{17}$$

where h_* is a learned "begin decoding" sentence embedding (see Figure 2.c). Each GRU has separate learned parameters; U, V and u, v are learned

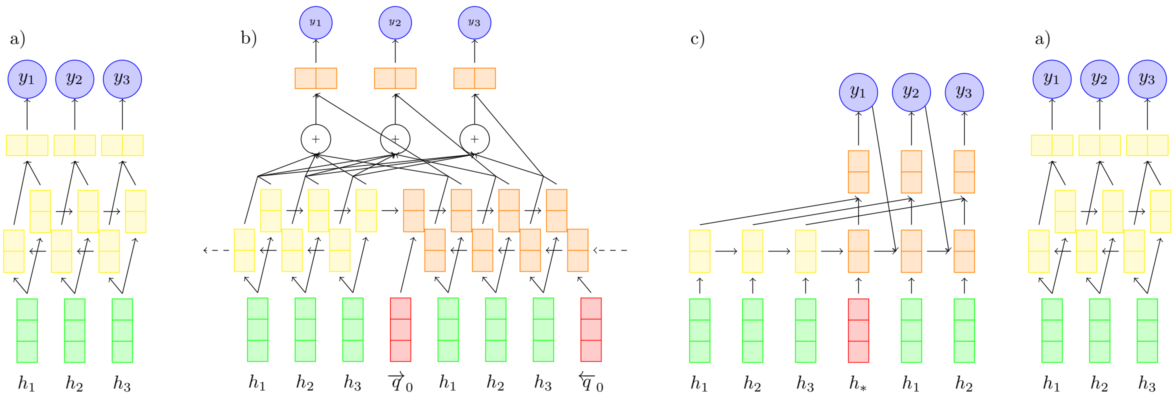

Figure 2: Sentence extractor architectures: a) RNN, b) Seq2Seq, c) Cheng & Lapata, and d) SummaRunner. The $\bigoplus$ indicates attention. Green repesents sentence encoder output, yellow and orange indicates extractor encoder and decoder hidden states respectively, and red indicates learned "begin decoding" embeddings.

weight and bias parameters. Note in Equation 15 that the decoder side GRU input is the sentence embedding from the previous time step weighted by its probabilitiy of extraction (p_{i-1}) from the previous step, inducing dependence of each output y_i on all previous outputs $y_{<i}$.

[[Kathy: In many ways the Cheng and Lapata architecture looks simpler than yours. Can you indicate here why it is more complicated? Is it because of the number of learned parameters?]]

Note that in the original paper, the Cheng & Lapata extractor was paired with a *CNN* sentence encoder, but in this work we experiment with a variety of sentence encoders.

SummaRunner Extractor Our second baseline, which we refer to as the SummaRunner extractor is taken from (**?**). Like the RNN extractor it starts with a bidirectional GRU over the sentence embeddings

$$\overrightarrow{z}_0 = \mathbf{0}; \quad \overrightarrow{z}_i = \overrightarrow{\mathrm{GRU}}(h_i, \overrightarrow{z}_{i-1}) \qquad (18)$$

$$\overleftarrow{z}_{n+1} = \mathbf{0}; \quad \overleftarrow{z}_i = \overleftarrow{\mathrm{GRU}}(h_i, \overleftarrow{z}_{i+1}), \qquad (19)$$

It then creates a representation of the whole document q by passing the averaged GRU output states through a fully connected layer:

$$q = \tanh\left(b_q + W_q \frac{1}{n}\sum_{i=1}^{n}[\overrightarrow{z}_i; \overleftarrow{z}_i]\right) \qquad (20)$$

A concatentation of the GRU outputs at each step are passed through a separate fully connected layer to create a sentence representation z_i, where

$$z_i = \mathrm{ReLU}\left(b_z + W_z[\overrightarrow{z}_i; \overleftarrow{z}_i]\right). \qquad (21)$$

The extraction probability is then determined by contributions from five sources:

$$\begin{aligned}
&\textit{content} & a_i^{(con)} &= W^{(con)} z_i, & (22)\\
&\textit{salience} & a_i^{(sal)} &= z_i^T W^{(sal)} q, & (23)\\
&\textit{novelty} & a_i^{(nov)} &= -z_i^T W^{(nov)} \tanh(g_i), & (24)\\
&\textit{position} & a_i^{(pos)} &= W^{(pos)} l_i, & (25)\\
&\textit{quartile} & a_i^{(qrt)} &= W^{(qrt)} r_i, & (26)
\end{aligned}$$

where l_i and r_i are embeddings associated with the i-th sentence position and the quarter of the document containing sentence i respectively. In Equation 24, g_i is an iterative summary representation computed as the sum of the previous $z_{<i}$ weighted by their extraction probabilities,

$$g_i = \sum_{j=1}^{i-1} p(y_j = 1 | y_{<j}, h) \cdot z_j. \qquad (27)$$

Note that the presence of this term induces dependence of each y_i to all $y_{<i}$ similarly to the Cheng & Lapata extractor.

The final extraction probability is the logistic sigmoid of the sum of these terms plus a bias,

$$p(y_i = 1 | y_{<i}, h) = \sigma\left(\begin{array}{l} a_i^{(con)} + a_i^{(sal)} + a_i^{(nov)} \\ +a_i^{(pos)} + a_i^{(qrt)} + b \end{array}\right). \qquad (28)$$

The weight matrices W_q, W_z, $W^{(con)}$, $W^{(sal)}$, $W^{(nov)}$, $W^{(pos)}$, $W^{(qrt)}$ and bias terms b_q, b_z, and

b are learned parameters; The GRUs have separate learned parameters.

Note that in the original paper, the SummaRunner extractor was paired with an *RNN* sentence encoder, but in this work we experiment with a variety of sentence encoders.

5 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 meetings (AMI), and medical journal articles (PubMed).**[[Hal: orphan]]**

CNN-DailyMail We use the preprocessing and training, validation, and test splits of (**?**) yielding 287,113/13,368/11,490 documents respectively, each with one reference abstract. This corpus is a mix of news on different topics including politics, sports, and celebrity news.**[[Hal: =entertainment?]]**

New York Times The New York Times (NYT) corpus (**?**) contains **[[Kathy: Not sure why you have this in red? Assume it won't stay in red for the final version?]]** 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**[[Hal: i'm not sure what you mean here. are you concatenating the two summaries? if so, say so. is that a little weird tho?]]**. This collection includes both straight newswire as well as opinion and long-form journalism. 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, yielding $44,382$ training, $5,523$ validation, and $6,495$ test documents.

DUC We use the single document summarization data from the 2001 and 2002 Document Understanding Conferences (DUC) (**?**). We split the 2001 data into training and validation splits and reserve the 2002 data for testing, resulting in 516/91/657 documents for training, validation, and test respectively. The test set has two or three human abstracts roughly 100 words in length per articles.

AMI The AMI corpus (**?**) is collection of real and staged office meetings annotated with text transcriptions, along with abstractive summaries. We use the proscribed**[[Hal: hehehe i think you mean prescribed :P]]** splits to get 98/19/20 training, validation, and test examples with one human abstract summary per meeting. We ignore any speaker information since we are primarily interested in studying content selection in a domain agnostic way. The summaries are about 290 words long on average and so we target this length for summary generation. **[[Hal: a) you don't need braces around paragraph text. b) this is the only place you mention task stuff rather than dataset stuff. mabye move target length information to the experiments section?]]**

Reddit **?** collected a corpus of personal stories shared on Reddit[1] along with multiple extractive and abstractive summaries. These stories are relatively short compared to the other corpora with an average sentence length of ??. **[[Kathy: Why didn't you use theirs?]]** We created our own train, validation, and test splits resulting in 404/24/48 documents respectively.

PubMed We created a corpus of 25,000 randomly samples medical journal articles from the PubMed Open Access Subset[2]. 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 21250/1250/2500 document for training, validation, test respectively. We used the article abstracts as the ground truth human summaries. **[[Hal: maybe mention this in the intro with a footnote about data release? we've recently used statements like the following in a footnote: 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.]]**

6 Experiments

[[Kathy: Could Lapata or Nallapati criticize your choice of parameters here as not being the same as what they used? (Note that I don't

[1] `www.reddit.com`

[2] `https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/`

Extractor	Encoder	CNN/DM R-2	NYT R-2	DUC 2002 R-2	Reddit R-2	AMI R-2
RNN	Average	25.42	34.67	22.65	**11.37**	**5.50**
	RNN	25.38	34.88	**22.58**	**11.43**	**5.24**
	CNN	25.08	33.70	**22.73**	**12.78**	3.16
Seq2Seq	Average	**25.56**	**35.73**	**22.84**	**13.61**	**5.52**
	RNN	25.34	**35.85**	22.47	**11.98**	**5.25**
	CNN	25.07	35.07	**22.67**	**13.21**	2.94
C&L	Average	25.29	**35.58**	**23.11**	**13.65**	**6.12**
	RNN	25.04	**35.84**	**23.03**	**12.62**	4.96
	CNN	25.13	34.97	**23.00**	**13.42**	2.81
SummaRunner	Average	25.44	35.40	22.28	**13.41**	**5.63**
	RNN	25.20	35.47	22.09	**12.51**	**5.38**
	CNN	25.00	34.41	22.22	**12.32**	3.16

Table 1: 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.

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 2: 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.

know what they used but it seems like a possible criticism.)]] We train all models to minimize the weighted negative log likelihood

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

over the training data $\mathcal{D}$ using stochastic gradient descent with the ADAM optimizer **[[Hal: maybe just say $\omega(0) = 1$ and $\omega(1) = N_0/N_1$ where N_y is the number of examples with label y.]]** (?). The term

$$\omega(y_i) = \begin{cases} \frac{\sum_{y_j} \mathbb{1}\{y_j=0\}}{\sum_{y_j} \mathbb{1}\{y_j=1\}} & \text{if } y_i = 1 \\ 1 & \text{otherwise} \end{cases}$$

upweights positive examples proportionally to the ratio of negative to positive examples in the training data in order to compensate for the sparsity of the positive class. **[[Kathy: Could you say how you arrived at these settings? Is any one system penalized by the use of uniform settings?]]** 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**[[Hal: there's prolly a bunch here that could go to the appendix]]** initializations. Weight matrix parameters are initialized using Xavier initialization with the normal distribution (?) and bias terms are set to 0. Unless specified, word embeddings are initialized using pretrained GloVe embeddings (?) 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 for which we use sizes two and four respectively. [[Kathy: why? Say.]] 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.

6.1 Ground Truth Extract Summaries

[[Hal: i feel like this should go before the optimizer dtails]] 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. Starting with an empty summary $S = \emptyset$, we add the sentence $\hat{s} = \arg\max_{s \in \{s_1,\ldots,s_d\},\, s \notin S}$ ROUGE-1$(S \cup s)$ to S stopping when the ROUGE-1 score no longer increases or the length budget is reached**[[Hal: what does this mean. does it mean that if the $\hat{s}$ that's selected puts you over length limit then you stop, or is it that you stop when there are no more sentences that you can cram in? and what if adding $\hat{s}$ doesn't make rouge go up?]]**. We choose to optimize for ROUGE-1 rather than ROUGE-2 similarly to other optimization based approaches to summarization **(???)** which found this be the easier optimization target.

[[Hal: do you say anywhere how you do significance testing and what significance level you use?]]

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 5.

[[Kathy: Save all results for results section. I deleted your sentence.]] [[Hal: i agree with kathy. put all the results together. be specific about what questions you're asking and then how you framed them as an experiment and then what the answer is. i think i'd just remove all this stuff here.]]

Word Embedding Learning To futher understand how word embeddings can effect model performance we also compared extractors when embeddings are updated during training. Both fixed and learned embedding variants are initialized with GloVe embeddings. When learning embeddings, words occurring three or fewer times in the training data are mapped to a learned unkown token.

POS Tag Ablation Additionally, we ran ablation experiments using part-of-speech (POS) tags. **[[Hal: this needs to be justified. why is this experiment interesting?]]** All datasets were automatically tagged using the SpaCy POS tagger [3]. **[[Kathy: I'm still curious what would happen if you separately removed all conjunction tags and later remaining POS.]]** We experimented with selectively removing

- nouns (NOUN and PROPN tags),
- verbs (VERB, PART, and AUX tags),
- adjectives/adverbs (ADJ and ADV tags),
- numerical expressions (NUM and SYM tags), and
- miscellaneous words (ADP, CONJ, CCONJ, DET, INTJ, and SCONJ tags)

from each sentece. The embeddings of removed words were replaced with a zero vector, preserving the order and position of the non-ablated words in the sentence.

Document Shuffling In examining the outputs of the models, 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**[[Hal: can you be more specific? like give some stats? what %age come from first quarter of doc and what %age from last half or something]]**. 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 perform a series of sentence order experiments 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.

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

7 Results

The results of our main experiment comparing the different extractors/encoders are shown in Table 1. 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. **[[Hal: i wonder if it would be worth adding another "average performance metric" column to Table 1. i'm thinking have "Average Δ-Best" meaning how far (on average across the datasets) is this setting from the best setting available on that dataset. so since the best numbers are: 25.56, 35.85, 23.11, 13.65, 5.63 and the first row numbers are: 25.42, 34.67, 22.65, 11.37, 5.50 then the deltas are: 0.14, 1.18, 0.46, 2.28, 0.13 the the average delta is 0.84 (assuming my math is right) there's an argument to do multiplicative, in which case the multipliers for first row: 0.99, 0.97, 0.98, 0.83, 0.97 and the average is 0.95 either way this gives a quick way to make comparisons between rows. you could do the same for the other tables too.]]**

[[Hal: in some of the tables you list R-2 as headers even though all the numbers are R-2. just put that in the caption.]]

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 Table 2 shows ROUGE recall when using fixed or updated word embeddings. In all but one case, fixed embeddings are as good or better than the learned embeddings. This is a somewhat surprising finding, and suggests that our models**[[Hal: why is it surprising?]]** cannot extract much generalizable learning signal from the content 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 Ablation Table 3 shows the results of the POS tag ablation experiments. **[[Hal: i'd label the rows as "all words" and then "- nouns", "-verbs", etc. also use paragraph not textbf :P.]]** While removing any word class from the representation generally hurts **[[Hal: is this removed at train and test or just test?]]** 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 Table 4 shows the results of our shuffling experiments. The newswire domain suffer 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 perfor-

Ablation	CNN/DM R-2	NYT R-2	DUC 2002 R-2	Reddit R-2	AMI R-2
–	**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
misc	25.24†	34.55†	**22.89**†	10.26†	**6.32**†
num	25.36†	34.60†	22.55	11.05	5.20†

Table 3: ROUGE-2 recall after removing different word classes. 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.

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 4: 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.

mance on the AMI.

8 Discussion

Learning in the news domain is severely inhibited by the lead bias. The output of all systems is highly similar to the lead baseline, with ??% of all system output sentences also occurring in the lead summary. Shuffling improves this (reducing to ??%) 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 (**???**). 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.

9 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.

Source Dataset	Extractor	Doc Order	cnn-dailymail R2	nyt R2	duc-sds R2	reddit R2	ami R2
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**

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

$$\overrightarrow{h}_0 = \mathbf{0}; \quad \overrightarrow{h}_i = \overrightarrow{\mathrm{GRU}}(w_i, \overrightarrow{h}_{i-1}) \tag{29}$$

$$\overleftarrow{h}_{|s|+1} = \mathbf{0}; \quad \overleftarrow{h}_i = \overleftarrow{\mathrm{GRU}}(w_i, \overleftarrow{h}_{i+1}), \tag{30}$$

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:

$$a_i^{(m,k)} = b^{(m,k)} + \sum_{j=1}^{k} W_j^{(m,k)} \cdot w_{i+j-1} \tag{31}$$

$$h^{(m,k)} = \max_{i \in 1,\ldots,|s|-k+1} \mathrm{ReLU}\left(a_i^{(m,k)}\right) \tag{32}$$

$$h = \left[h^{(m,k)} | m \in \{1, \ldots, M\}, k \in K\right] \tag{33}$$

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.