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 problems, i.e. given some context, determine which information needs to be expressed in output text (Gatt and Krahmer, 2018). While it is a key component of extractive and abstractive summarization systems, it is not generally well understood, with frequency and information theortic measures used as proxies for content salience (Hong and Nenkova, 2014).

In this paper, we seek to better understand how deep learning models of summarization are performing content selection across a variety of domains. We perform an analysis of several recent sentence extractive neural network architectures, looking particulary at the choice of sentence encoder and sentence extractor design.

Our experiments reveal several worrying obstacles for learning such models. For instance, the sentence position bias dominates the learning signal in the news domain; this can be corrected for by shuffling a document's sentence order but not without adverse effects on performance.

In our study of sentence encoder architectures, we find that word embedding averaging is as good or better than more sophisticated recurrent neural network (RNN) and convolutional neural network (CNN) encoders. Additionally fixed pretrained word embeddings are as good or better than learned embeddings in the majority of cases.

We also propose two simple extractor methods that make sentence selection decisions independently in contrast to two previously published architectures where previous selection decisions inform future sentence selection probabilities. Oddly we find that the independent predictions are as good as sequential predictions.

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.

While we are explicitly studying extractive summarization algorithms here, we think the findings will be relevant to the abstractive summarization community as well. The encoder side architectures are quite similar to typical abstractive models, and fundamentally the model objectives are the same, producing output text with high word overlap to a reference human abstract.

The contributions of this paper are the following:

1. We perform an empirical study of extractive content selection in deep learning algorithms for text summarization across news, personal narratives, meetings, and medical journal domains using both automatic and manual evaluation.

2. We propose two simple sentence extractor models whose performance is on par with more complex models.

In the following sections we discuss (Sec. ?) related work, (Sec. ?) define the problem of extractive summarization, (sec. ?) formulate our proposed ad baseline models, (Sec. ?) describe the datasets used for experiments, (Sec. ?) describe the experiments themselves, and conclude with results and analysis (Sec. ?).

2 Related Work

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 (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.

An notable exception is that of (Yasunaga et al., 2017) which learns 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, typically they have involved learning ngram 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).

The introduction of the CNN-DailyMail corpus by (Hermann et al., 2015) allowed for the application of large-scale training of deep learning models, initially by (Cheng and Lapata, 2016) who present both a sentence and word extractive model of single document summarization. The sentence extractive model uses a word level convolutional neural network (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 different model using word-level bidirectional recurrent neural networks (RNNs) along with a sentence level bidirectinoal RNN for predicting which sentences should be extractive. Additionally, the sentence extractor creates representations of the whole document 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 presented in Section **??**.

All of the previous works have focused on news summarization. To further understand the content selection process, we also explore other domains of summarization. In particular, we explore personal narratives shared on the website 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 features relevant to the domain, e.g. speaker identification in meeting summarization (Gillick et al., 2009), we eschew such features in this work in order to understand generally how content selection is learned.

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 (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.

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.

We evaluate summary quality using ROUGE (Lin, 2004) which measures the ngram overlap of our predicted extract summary with one or more human abstracts. We use ROUGE-2 recall (i.e. bigram recall) as our main evaluation metric (ROUGE-1 and ROUGE-LCS trend similarity to ROUGE-2 in our experiments). We also evaluate using METEOR (Denkowski and Lavie, 2014) which measures precision and recall of reference words, while allowing for more complicated word matchings (e.g. via synonymy or morphology).

In this paper, we model this task as a sequence tagging problem, i.e. given a document containing d sentences $s_1, \ldots, s_d$ we want to 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.

4 Models and Methods

[[Hal: i'm having a hard time understanding what's new and what's prior work here.]]

At a high level, all the models considered in this paper share the same two part structure: *i) a sentence encoder* which each sentence s_i (treated as a

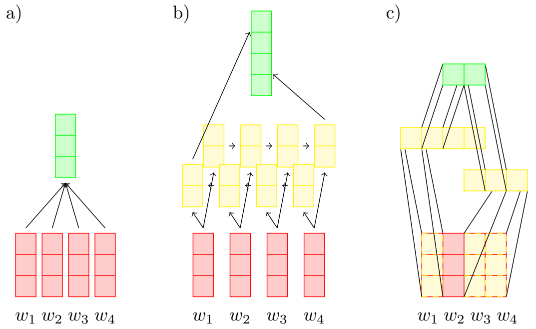

Figure 1: Sentence encoder architectures: a) averaging encoder, b) RNN encoder c) CNN encoder. Red indicates word embeddings, yellow indicates RNN hidden states or convolutional activations, and green indicates the sentence embedding that is passed to the extractor module.

sequence of tokens) to an embedding $h_i \in \mathcal{R}^d$, **[[Hal: notation class, you used d already for number of sentences]]** and *ii) a sentence extractor* which takes as input all of a document's sentence embeddings and predicts which sentences to extract to produce the extract summary. The sentence extractor is essentially a discriminative classifier $p(y_1, \ldots, y_d | h_1, \ldots, h_d)$.

Depending on the architectural choices of each component we propose we can recover the specific implementations of (**?**) and (**?**), which we outline below.

4.1 Sentence Encoders

[[Hal: these descriptions are inconsistent in notation. in the first you use enc(s) but before you say h, and in other palces you say h below. h also switches between hidden state of the encoder and the output of the encoder. you'll also need to say somewhere that w represents the word embedding of a word.]]

We treat each sentence $s = \{w_1, \ldots, w_{|s|}\}$ as a sequence of word embeddings, where $|s|$ 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: average pooling, 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. A sentence encoding is simply the average of its word embeddings: $\text{enc}(s) = \frac{1}{|s|} \sum_{w \in s} w$. **[[Hal: probably better to write $\frac{1}{|s|} \sum_{i=1}^{|s|} w_i$ since set notation makes me wonder about words that appear more than once.]]**

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. Formally, an *RNN* sentence encoding is defined as $\text{enc}(s) = [\overrightarrow{h}_{|s|}; \overleftarrow{h}_1]$ where

$$\overrightarrow{h}_i = \overrightarrow{\text{GRU}}(w_i, \overrightarrow{h}_{i-1}) \quad (1)$$
$$\overleftarrow{h}_i = \overleftarrow{\text{GRU}}(w_i, \overleftarrow{h}_{i+1}) \quad (2)$$

for all $i \in 1, \ldots, |s|$. $\overrightarrow{\text{GRU}}$ amd $\overleftarrow{\text{GRU}}$ indicate the forward and backward GRUs respectively, and each have separate learned parameters. The initial states $\overrightarrow{h}_0$ and $\overleftarrow{h}_{|s|+1}$ are not learned and are set to zero vectors.

[[Hal: maybe you can make this more brief by saying sth like:

$$\overrightarrow{h}_0 = \mathbf{0}; \quad \overrightarrow{h}_{i+1} = \ldots \quad (3)$$
$$\overleftarrow{h}_{|s|+1} = \mathbf{0}; \quad \overleftarrow{h}_{i-1} = \ldots \quad (4)$$

then you don't need to spe]]

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 *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 encoding is computed as follows:

$$a_i^{(m,k)} = b^{(m,k)} + \sum_{j=1}^{k} W_j^{(m,k)} \cdot w_{i+j-1} \quad (5)$$

$$h^{(m,k)} = \max_{i \in 1,\ldots,|s|-k+1} \text{ReLU}\left(a_i^{(m,k)}\right) \quad (6)$$

$$\text{enc}(s) = \left[h^{(m,k)} | m \in \{1, \ldots, M\}, k \in K\right] \quad (7)$$

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 $\text{ReLU}(x) = \max(0, x)$ is the rectified linear unit (**?**).

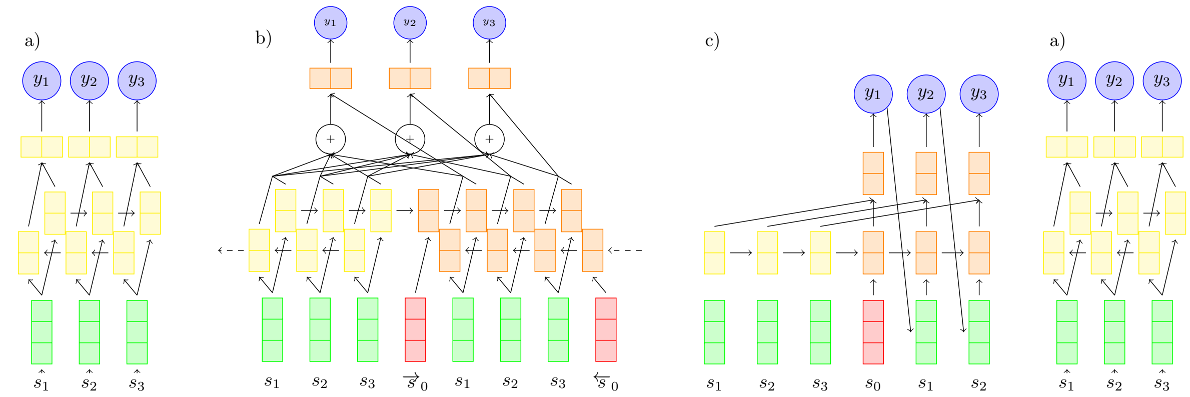

Figure 2: Sentence extractor architectures: a) **RNN**, b) **Seq2Seq**, c) **C&L**, and d) **SR**.

4.2 Sentence Extractors

Given a sequence of sentence embeddings $h_i = \text{enc}(s_i)$, a sentence extractor produces a conditional distribution over the corresponding sentence extraction variables $p(y_1, \ldots, y_{|s|}|h_1, \ldots, h_{|s|})$. We propose two simple recurrent neural network based sentence extractors that make a strong conditional independence assumption over the labels y_i, namely $p(y_1, \ldots, y_{|s|}|h_1, \ldots, h_{|s|}) = \prod_{i=1}^{|s|} p(y_i|h_1, \ldots, h_{|s|})$. This stands in contrast to our baseline models which make a weaker assumption, **[[Hal: this is really confusing cuz i don't think you've specified this yet, and it's still hard to understand what's new/yours and what's old/baseline.]]** $p(y|h) = \prod_{i=1}^{|s|} p(y_i|y_{<i}, h_1, \ldots, h_{|s|})$, at the expense of greater computational complexity. 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 the forward and backward outputs of each sentence are passed through a single layer perceptron with a logsitic sigmoid output (denoted by σ) to predict the probability of extracting each sentence: **[[Hal: i'd sugest following the same notation as before to remove the zero comment]]**

$$\overrightarrow{z}_i = \overrightarrow{\text{GRU}}(h_i, \overrightarrow{z}_{i-1}) \quad (8)$$

$$\overleftarrow{z}_i = \overleftarrow{\text{GRU}}(h_i, \overleftarrow{z}_{i+1}) \quad (9)$$

$$a_i = \text{ReLU}\left(U \cdot [\overrightarrow{z}_i; \overleftarrow{z}_i] + u\right) \quad (10)$$

$$p(y_i = 1|h) = \sigma\left(V \cdot a_i + v\right) \quad (11)$$

for all $i \in 1, \ldots, d$. $\overrightarrow{\text{GRU}}$ amd $\overleftarrow{\text{GRU}}$ indicate the forward and backward GRUs respectively, and each have separate learned parameters; U, V and u, v are learned weight and bias parameters. The initial states $\overrightarrow{z}_0$ and $\overleftarrow{z}_{d+1}$ are not learned and are set to zero vectors.

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, which we refer to as the **Seq2Seq** extractor, is based on the attentional sequence-to-sequence models 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 multi-layer percepron to compute the extraction probability. Formally we have:

$$\overrightarrow{z}_i = \overrightarrow{\text{GRU}}_{enc}(h_i, \overrightarrow{z}_{i-1}) \quad (12)$$

$$\overleftarrow{z}_i = \overleftarrow{\text{GRU}}_{enc}(h_i, \overleftarrow{z}_{i+1}) \quad (13)$$

$$\overrightarrow{q}_i = \overrightarrow{\text{GRU}}_{dec}(h_i, \overrightarrow{q}_{i-1}) \quad (14)$$

$$\overleftarrow{q}_i = \overleftarrow{\text{GRU}}_{dec}(h_i, \overleftarrow{q}_{i+1}) \quad (15)$$

$$q_i = [\overrightarrow{q}_i; \overleftarrow{q}_i], \quad z_i = [\overrightarrow{z}_i; \overleftarrow{z}_i] \quad (16)$$

$$\alpha_{i,j} = \frac{\exp\left(q_i \cdot z_j\right)}{\sum_{j=1}^{d} \exp\left(q_i \cdot z_j\right)}, \quad \bar{z}_i = \sum_{j=1}^{d} \alpha_{i,j} z_j \quad (17)$$

$$a_i = \text{ReLU}\left(U \cdot [\bar{z}_i; q_i] + u\right) \quad (18)$$

$$p(y_i = 1|h) = \sigma\left(V \cdot a_i + v\right), \quad (19)$$

for all $i \in 1, \ldots, d$. The final outputs of each encoder direction are passed to first decoder steps; additionally, the first step of the decoder GRUs are leanered "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. The initial states $\overrightarrow{z}_0$ and $\overleftarrow{z}_{d+1}$ are not learned and are set to zero vectors.

Cheng & Lapata Extractor We compare the previously proposed architectures to the sentence extractor model of (?). Unlike the previous models where sentence extraction predictions are conditionally independent given the sentence embeddings, this model uses previous extraction probabilities to influence later decisions. The basic architecture is a unidirectional sequence-to-seqeunce model defined as follows:

$$z_i = \mathrm{GRU}_{enc}(h_i, z_{i-1}) \quad (20)$$

$$q_i = \mathrm{GRU}_{dec}(p_{i-1} \cdot h_{i-1}, q_{i-1}) \quad (21)$$

$$a_i = \mathrm{ReLU}\left(U \cdot [z_i; q_i] + u\right) \quad (22)$$

$$p_i = p(y_i = 1 | y_{<i}, h) = \sigma\left(V \cdot a_i + v\right) \quad (23)$$

for all $i \in 1, \ldots, d$. The final output of the encoder is passed to the first decoder step; additionally, the first step of the decoder GRU is a learned "begin decoding" vector q_0 (see Figure 2.c). Each GRU has separate learned parameters; U, V and u, v are learned weight and bias parameters. The initial states $\overrightarrow{z}_0$ and $\overleftarrow{z}_{d+1}$ are not learned and are set to zero vectors. Note in Equation 18 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.

We refer to this extractor as the **C&L** extractor; the model architecture descrbied in (?) is equaivalent to the **C&L** extractor paired with the *CNN* encoder.

[[Hal: i'd use full names throughout if possible, so just Chang&Lapata and SummaRunner and...]]

SummaRunner Extractor Our second baseline extractor is taken from (?), which like the **RNN** extractor starts with a bidrectional GRU over the sentence embeddings

$$\overrightarrow{z}_i = \overrightarrow{\mathrm{GRU}}(h_i, \overrightarrow{z}_{i-1}) \quad (24)$$

$$\overleftarrow{z}_i = \overleftarrow{\mathrm{GRU}}(h_i, \overleftarrow{z}_{i+1}), \quad (25)$$

for all $i \in 1, \ldots, d$. The initial states $\overrightarrow{z}_0$ and $\overleftarrow{z}_{d+1}$ are not learned and are set to zero vectors.

Unlike **RNN**, however, it then creates a representation of the whole document q by passing the averaged GRU output states through a fully connected layer:

$$q = \tanh\left(b_q + W_q \frac{1}{d}\sum_{i=1}^{d}[\overrightarrow{z}_i; \overleftarrow{z}_i]\right) \quad (26)$$

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

$$z_i = \mathrm{ReLU}\left(b_z + W_z[\overrightarrow{z}_i; \overleftarrow{z}_i]\right). \quad (27)$$

The extraction probability is then determined by contributions from five sources, *i)* a content score $a_i^{(con)} = W_{con} \cdot z_i$, *ii)* a salience score $a_i^{(sal)} = z_i^T W_{sal} \cdot q$ quantifying the similarity of the sentence to the document, *iii)* a novelty score $a_i^{(nov)} = -z_i^T W_{nov} \cdot \tanh(g_i)$ that tracks negative similarity to a representation of the extract summary g_i, *iv)* a fine-grained position score $a_i^{(fpos)} = W_{fpos} \cdot p_i$ where p_i is a sentence position embedding for the i-th position in the document, and *v)* a coarse-grained position score $a_i^{(cpos)} = W_{cpos} \cdot r_i$ where r_i is a position embedding for the quarter of the document that sentence i occurs in. The final extraction probability is the logistic sigmoid of the sum of these terms plus a bias,

$$p(y_i = 1 | y_{<i}, h) = \sigma\begin{pmatrix} a_i^{(con)} + a_i^{(sal)} + a_i^{(nov)} \\ + a_i^{(fpos)} + a_i^{(cpos)} + b \end{pmatrix}. \quad (28)$$

Finally, the iterative summary representation g_i is computed as a sum of the previous $z_{<i}$ weighted by their extraction probabilities,

$$g_i = \sum_{j=1}^{i-1} p(y_j = 1 | y_{<j}, h) \cdot z_j. \quad (29)$$

Note that the presence of this term induces dependence of each y_i to all y_j for $j < i$ similarly to the **C&L** extractor. The combination of this extractor, which we refer to as **SR**, and the *RNN* encoder is most similar to the architecture described in (?) with the principal difference that we are using the RNN final states for the sentence representation and they used the average of the RNN output.

5 Datasets

We perform our experiments across six corpora from varying domains to understand how different biases with each domain can effect content selection. The corpora come from the news domain (CNN-DM, NYT, DUC), personal narratives domain (Reddit), workplace meetings (AMI), and medical journal articles (PubMed).

CNN-DailyMail The CNN-DailyMail (CNN-DM) corpus was first used for the summarization task by (**?**), when it was noted that the bulleted highlights associated with each article could serve as a document summary. We use the preprocessing and training, validation and test splits of (**?**) yielding ?/?/? documents respectively, each with one reference abstract. This corpus is a mix of news on different topics including politics, sports, and celebrity news.

New York Times The New York Times (NYT) corpus (**?**) 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. This collection includes both hard newswire as well as opinion and long-form journalism. We create training, validation, test splits by partitioning on dates yield ?/?/? 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 tesing, resulting in ?/?/? documents for training, validation, and test respectively. The test set has two or three human abstracts rougly 100 words in length per articles.

AMI The AMI corpus (**?**) is collection real and staged office meetings annotated with text transcriptions, along with abstractive summaries. We use the proscribed splits to get ?/?/? 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.

Reddit (**?**) collected a corpus a personal stories shared on Reddit[1].

We use a collection of personal stories paired with multiple abstractive and extractive summaries. As in (**?**)

reddit

pubmed

6 Experiments

[[Hal: use paragraph instead of textbf for things like the below]]

Extractor/Encoder In our main experiment we compare our proposed **[[Hal: i used xpsace to make these space properly]]** sentence extractors **RNN** and **Seq2Seq** against the **C&L** and **SR** extractors. We test all possible sentence extractor/encoder pairs across the CNN-DailyMail, New York Times, DUC 2002, Reddit, AMI, and PubMed domains. We choose ROUGE-2 recall as our main evaluation metric since it has the strongest correlation to human content selection decisions. In this we experiment we initialize the word embeddings using pretrained GloVe embeddings (**?**) and do not update them during training.

In most cases, the averaging encoder performance was as good or better than the RNN and CNN encoders, we use only the averaging encoder for the remainder of the experiments.

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 an *unkown* token.

POS Tag Ablation Additionally, we ran ablation experiments using part-of-speech (POS) tags. We experimented with selectively removing nouns, verbs, adjectives/adverbs, numerical expressions, and miscellaneous tags (anything that was not in the previously mentioned groups) 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. All datasets were automatically tagged using the SpaCy POS tagger (**?**).

Document Shuffling In examining the outputs of the models, we found most of the selected sen-

[1] www.reddit.com

tences in the news domain came from the lead paragraph 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. 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 original unshuffled models.

Cross Domain Experiments

TODO

Try shuffled and no shuffled models trained on one domain, eval the remaining.

6.1 Model Training Details

TODO: Clean up, expand, make naming consistent!

We train the average pooling, biRNN, and CNN encoders with the biRNN, seq2seq, SummaRunner, and C&L decoders. We repeat experiments across the CNN-DailyMail, New York Times, DUC, Reddit, and AMI corpus. We use the Adam optimizer for all models with a learning rate of .0001, gradient clipping, and dropout rate of .25. For the C&L model, we train for half of the maximum epochs with teacher forcing, i.e. we use the gold extractive labels for each when taking the sum of previous states (p(y—x)h = 1 If y = 1) during the first half of training and the model value during the second. We use early stopping on the validation set with ROUGE2 as our evaluation criteria. All experiments are run with 5 different random initialization seeds and results are averaged.

Since we do not typically have ground truth extract summaries from which to create the labels y_i, we construct gold label sequences for training by greedily optimizing ROUGE-1. Starting with an empty summary $S = \emptyset$, we add the sentence $\hat{s} = \arg\max_{s \in \{s_1,\ldots,s_d\},\, s \notin S} \text{ROUGE-1}(S \cup s)$ to S stopping when the ROUGE-1 score no longer increases or the length budget is reached. 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.

7 Results

Choice of Extractor Our main comparison of extractors/encoders are shown in Table 1. Overall we find that the **Seq2Seq** extractor achieves the best ROUGE scores on three out of four domains (STILL RUNNING ON AMI AND PUBMED). However, most differences are not signficant. (Need to discuss stat sig and how to show it). On the larger CNN-DailyMail dataset, especially, differences are quite smail across all extractor/encoder pairs. The **C&L** extractor achieves the best performance on the DUC 2002 dataset. It is disappointing that the **C&L** and **SR** based models do not gain any apparent advantage in conditioning on previous sentence selection decisions; this result suggests the need to improve the representation of the summary as it is being constructed iteratively.

Choice of Encoder We also find there to be no major advantage between the different sentence encoders. In most cases, there is no statistical significance between the averaging encoder and either the RNN or CNN encoders.

Learning Word Embeddings Table 2 shows ROUGE recall when using fixed or updated word embeddings. In almost all cases, fixed embeddings are as good or better than the learned embeddings.

POS Ablation Table 3 shows the results of the POS tag ablation experiments. The newswire domain does not appear to be sensative to these ablations; this suggests that the models are still able to identify the lead section of the document with the remaining word classes (Verify this with histogram analysis). The Reddit domain, which is not lead biased, is significantly effected. Notably, removing adjectives and adverbs results in a 1.8 point drop in ROUGE-2 recall.

Sentence Shuffling We find a similar result on the sentence order shuffling experiments. Table 4 shows the results. 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.

References

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 Lan-*

guage 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*.

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.

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*.

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.

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.

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.

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*.

Extractor	Encoder	CNN/DM R-2	NYT R-2	DUC 2002 R-2	Reddit R-2	AMI R-2
RNN	Avg.	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	Avg.	**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	Avg.	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	Avg.	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 1 and 2 recall results across different 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.

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	$25.31^{\dagger}$	$34.26^{\dagger}$	$22.35^{\dagger}$	$10.29^{\dagger}$	$3.76^{\dagger}$
verbs	$25.25^{\dagger}$	$34.36^{\dagger}$	$22.45^{\dagger}$	10.76	5.80
adjv	$25.28^{\dagger}$	$34.36^{\dagger}$	22.50	$9.48^{\dagger}$	5.36
misc	$25.24^{\dagger}$	$34.55^{\dagger}$	$\mathbf{22.89}^{\dagger}$	$10.26^{\dagger}$	$\mathbf{6.32}^{\dagger}$
num	$25.36^{\dagger}$	$34.60^{\dagger}$	22.55	11.05	$5.20^{\dagger}$

Table 3: ROUGE recall after removing different word classes. Ablations are performed using the averaging sentence encoder and **RNN** extractor. Table shows average results of five random initializations.

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 1 and 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.

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**