Content Selection in Deep Learning Models of Summarization

Anonymous EMNLP submission

Abstract

1 Introduction

While there has a been a recent flurry of work on abstractive summarization (**????**), these papers treat this problem as a pure sequence to sequence transduction task. Admittedly, this view allows us to apply very powerful, general-purpose deep learning archictures to generate summaries. At the same time, it obscures a principal subtask in summarization, the process of selecting the most salient units of meaning in the source material, i.e. the key ingredients in the final summary, a process which we broadly refer to as content selection (**?**).

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 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.

In this paper, we seek to better understand how deep learning models of summarization are performing content selection. We perform an analysis of several recent sentence extractive neural network architecures, looking particulary at the impact of sentence position bias, the necessity of learning embeddings, the unreasonable effectiveness of averaging for sentence embedding, and the cross domain generalizability of such models. Additionally, we propose two simpler models that are on average statistically indistinguishable from their more complex counterparts.

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, blogs, meetings, and journal articles domains, and small/medium/large datasets.

2. Propose two simple deep learning models whose performance is on par with more complex deep learning 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

Extractive Deep Learning Based Summarization

Nallapati et. al
Cheng and Lapata

Abstractive Deep Learning Based Summarization

Rush
Chopra
See at al.

Socher

Extractive Single Doc Summarization Durret et. al

Non Newswire Summarization meeting summarization
reddit stories
journal articles/pubmed

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 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.

Unlike sequence tagging, however, we do not evaluate model performance by label accuracy or F-measure but ROUGE (**?**) which measures the ngram overlap of our predicted extract summary with one or more human abstracts. While this metric has many shortcomings, ROUGE-2 recall (i.e. bigram recall) has been empirically demonstrated to have high correlation with human content selection decisions for summarization (**?**).

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 (**????**) which found this be the easier optimization target.

4 Models and Methods

At a high level, all of our models share the same two part structure: *i) a sentence encoder* which maps an arbitrary sequence of tokens to an embedding $h \in \mathcal{R}^d$, 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 $p(y_1, \ldots, y_d | h_1, \ldots, h_d)$.

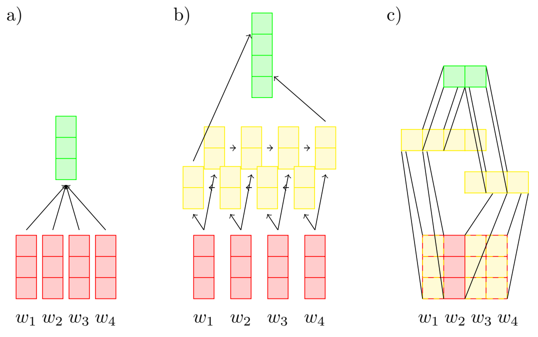

Figure 1: Sentence encoder architectures: a) averaging encoder, b) RNN encoder c) CNN encoder. Red indicates word embeddings, yellow indicates RNN hidden states or convolutional activations, and green indicates the sentence embedding that is passed to the extractor module.

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

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: $\text{enc}(s) = \frac{1}{|s|}\sum_{w \in s} w$.

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.

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:

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

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

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

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

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,chiefly $p(y_1, \ldots, y_{|s|} | h_1, \ldots, h_{|s|}) = \prod_{i=1}^{|s|} p(y_i | h_1, \ldots, h_{|s|})$. This stands in contrast to our baseline models which make a weaker assumption, $p(y|h) = \prod_{i=1}^{|s|} p(y_i | y_{<i}, h_1, \ldots, h_{|s|})$, 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:

$$\overrightarrow{z}_i = \overrightarrow{\text{GRU}}(h_i, \overrightarrow{z}_{i-1}) \qquad (6)$$

$$\overleftarrow{z}_i = \overleftarrow{\text{GRU}}(h_i, \overleftarrow{z}_{i+1}) \qquad (7)$$

$$a_i = \text{ReLU}\left(U \cdot [\overrightarrow{z}_i; \overleftarrow{z}_i] + u\right) \qquad (8)$$

$$p(y_i = 1 | h) = \sigma\left(V \cdot a_i + v\right) \qquad (9)$$

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:

$$\overrightarrow{z}_i = \overrightarrow{\text{GRU}}_{enc}(h_i, \overrightarrow{z}_{i-1}) \qquad (10)$$

$$\overleftarrow{z}_i = \overleftarrow{\text{GRU}}_{enc}(h_i, \overleftarrow{z}_{i+1}) \overrightarrow{q}_i \quad = \overrightarrow{\text{GRU}}_{dec}(h_i, \overrightarrow{q}_{i-1}) \qquad (11)$$

$$\overleftarrow{q}_i = \overleftarrow{\text{GRU}}_{dec}(h_i, \overleftarrow{q}_{i+1}) \qquad (12)$$

$$q_i = [\overrightarrow{q}_i; \overleftarrow{q}_i], \quad z_i = [\overrightarrow{z}_i; \overleftarrow{z}_i] \qquad (13)$$

$$\alpha_{i,j} = \frac{\exp\left(q_i \cdot z_j\right)}{\sum_{j=1}^{d} \exp\left(q_i \cdot z_j\right)}, \quad \bar{z}_i = \sum_{j=1}^{d} \alpha_{i,j} z_j \qquad (14)$$

$$a_i = \text{ReLU}\left(U \cdot [\bar{z}_i; q_i] + u\right) \qquad (15)$$

$$p(y_i = 1 | h) = \sigma\left(V \cdot a_i + v\right), \qquad (16)$$

for all $i \in 1, \ldots, d$. The final outputs of each encoder direction are passed to first decoder steps;

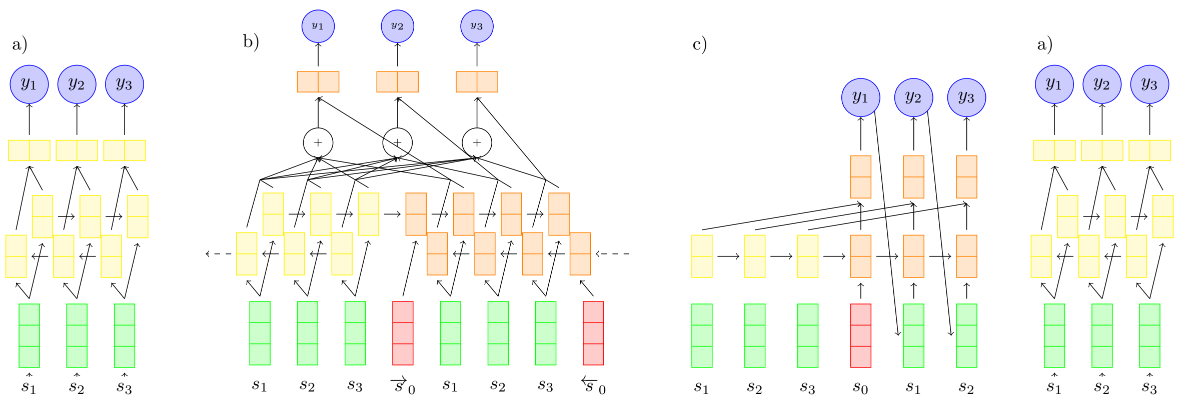

Figure 2: Sentence extractor architectures: a) **RNN**, b) **Seq2Seq**, c) **C&L**, and d) **SR**.

additionally, the first step of the decoder GRUs are learened "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:

$$z_i = \text{GRU}_{enc}(h_i, z_{i-1}) \quad (17)$$
$$q_i = \text{GRU}_{dec}(p_{i-1} \cdot h_{i-1}, q_{i-1}) \quad (18)$$
$$a_i = \text{ReLU}\left(U \cdot [z_i; q_i] + u\right) \quad (19)$$
$$p_i = p(y_i = 1 | y_{<i}, h) = \sigma\left(V \cdot a_i + v\right) \quad (20)$$

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.

SummaRunner Extractor

$$\overrightarrow{z}_i = \overrightarrow{\text{GRU}}(h_i, \overrightarrow{z}_{i-1}) \quad (21)$$
$$\overleftarrow{z}_i = \overleftarrow{\text{GRU}}(h_i, \overleftarrow{z}_{i+1}) \quad (22)$$
$$q = \tanh\left(b_q + W_q \frac{1}{d} \sum_{i=1}^{d} [\overrightarrow{z}_i; \overleftarrow{z}_i]\right) \quad (23)$$
$$z_i = \text{ReLU}\left(b_z + W_z[\overrightarrow{z}_i; \overleftarrow{z}_i]\right) \quad (24)$$

$$\begin{aligned} p(y_i = 1 | y_{<i}, h) = \sigma(&W_{con} \cdot z_i \\ &+ z_i^T W_{sal} \cdot q \\ &- z_i^T W_{nov} \cdot \tanh(g_i) \\ &+ b_{rp_i} \\ &+ b_{ap_i} \\ &+ b) \end{aligned}$$
$$g_j = \sum_{i=1}^{j-1} p(y_j = 1 | y_{<j}, h) \cdot z_j$$

5 Datasets

cnn-dailymail
nyt
duc
ami
reddit
pubmed

6 Experiments

Extractor/Encoder In our main experiment we compare our proposed 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 sentences 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.

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 ab-

		cnn-dailymail		nyt		duc-sds		reddit	
Extractor	Encoder	R-1	R-2	R-1	R-2	R-1	R-2	R-1	R-2
RNN	avg	55.3	25.4	51.4	34.7	44.1	22.6	45.2	11.4
	rnn	55.3	25.4	51.7	35.0	44.0	22.6	44.4	11.4
	cnn	54.7	25.1	50.3	33.7	44.3	22.7	47.5	12.7
Seq2Seq	avg	**55.6**	**25.6**	52.5	35.7	44.4	22.8	**49.1**	**13.6**
	rnn	55.2	25.3	52.5	**35.9**	43.8	22.5	45.4	12.1
	cnn	54.8	25.1	51.7	35.1	44.0	22.7	46.8	13.1
C&L	avg	55.1	25.3	52.3	35.6	**44.8**	**23.1**	48.3	13.6
	rnn	54.8	25.0	52.5	35.8	44.5	23.0	46.3	12.6
	cnn	55.0	25.1	51.5	35.0	44.5	23.0	46.9	13.4
SR	avg	55.3	25.4	52.1	35.4	44.0	22.3	48.8	13.4
	rnn	55.0	25.2	52.1	35.5	43.6	22.1	46.5	12.6
	cnn	54.6	25.0	51.0	34.4	43.6	22.2	46.6	12.3

Table 1: Rouge Recall 1 and 2 results across all sentence encoder/extractor pairs. All results are averaged over five random initializations. Best result per metric/dataset are bolded.

		cnn-dailymail		nyt		duc-sds		reddit	
system	embeddings	R1	R2	R1	R2	R1	R2	R1	R2
RNN	fixed	55.3	25.4	51.4	34.7	44.1	22.6	45.2	11.4
	learned	55.1	25.2	51.1	34.3	44.1	22.6	45.3	11.3
Seq2Seq	fixed	55.6	25.6	52.5	35.7	44.4	22.8	49.1	13.6
	learned	55.2	25.3	52.4	35.7	44.5	22.9	49.4	13.8
C&L	fixed	55.1	25.3	52.3	35.6	44.8	23.1	48.3	13.6
	learned	54.8	25.0	52.1	35.4	44.6	23.0	48.6	13.5
SummaRunner	fixed	55.3	25.4	52.1	35.4	44.0	22.3	48.8	13.4
	learned	55.0	25.1	52.0	35.2	43.8	22.1	47.8	12.6

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.

	cnn-dailymail		nyt		duc-sds		reddit	
Ablation	R1	R2	R1	R2	R1	R2	R1	R2
–	55.3	25.4	51.4	34.7	44.1	22.6	45.2	11.4
-nouns	55.2	25.3	50.8	34.3	43.8	22.3	43.0	10.3
-verbs	55.1	25.3	51.0	34.4	43.8	22.4	44.3	10.7
-adjv	55.2	25.3	51.0	34.4	44.0	22.5	42.5	9.6
-misc	55.1	25.2	51.2	34.5	44.5	22.9	43.1	10.3
-num	55.3	25.4	51.3	34.6	44.1	22.6	45.2	11.0

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.

lations; 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.

		nyt		duc-sds		reddit	
Extractor	sentence order	R1	R2	R1	R2	R1	R2
RNN	in-order	51.4	34.7	44.1	22.6	45.2	11.4
	shuffled	41.9	25.0	39.7	18.2	45.1	11.9
Seq2Seq	in-order	52.5	35.7	44.4	22.8	49.1	13.6
	shuffled	42.6	25.6	42.9	21.2	48.7	13.6

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.