Summarization II

May Jiang and Sonia Murthy

3/26/2020

Summarization: Overview

- **Task:** Creating a shorter version of one or more documents, while preserving their information content
- **Motivation:** Growing need to access and digest large amounts of textual data

Summarization: Extractive vs Abstractive

- **Extractive:** Summary created by identifying (i.e. *extracting*) and concatenating the most salient text units in a document

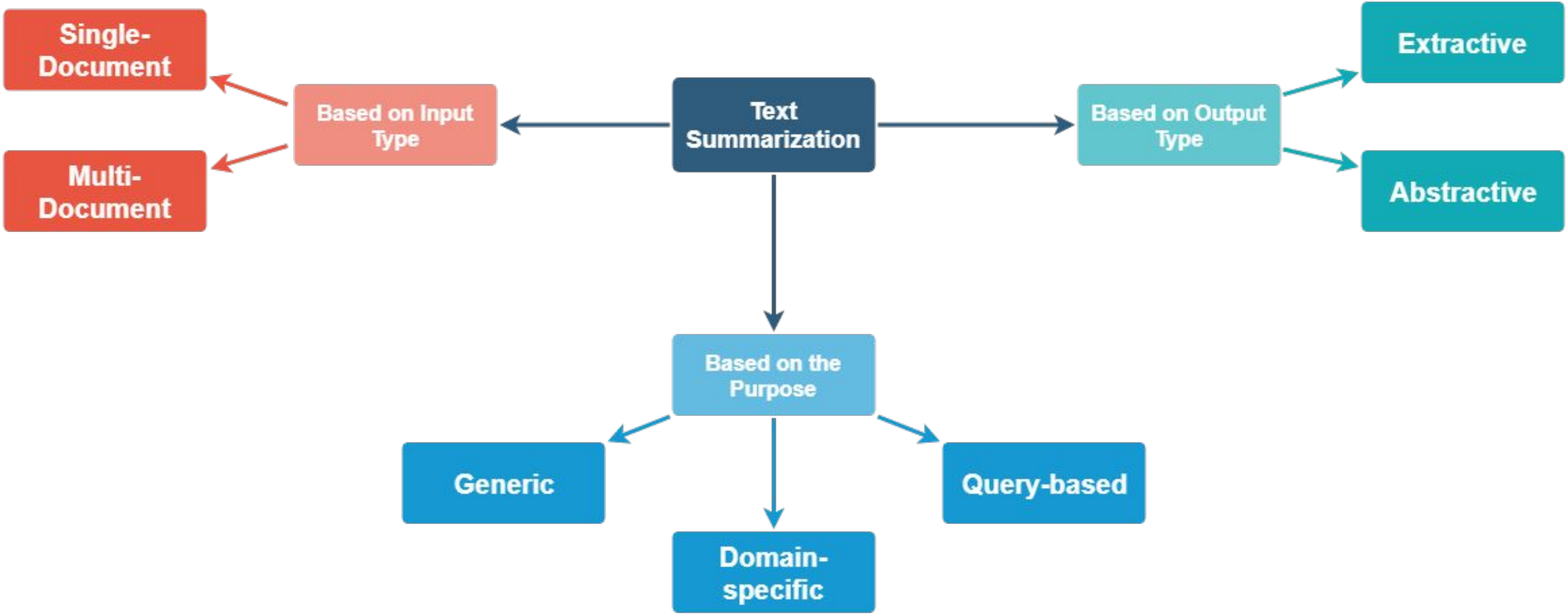
- **Abstractive:** Summary created by generating novel sentences - not restricted to source text

Recap: Extractive vs Abstractive

- Abstractive can produce higher quality summaries because it allows for paraphrasing, generalization, etc
    - But, liable to reproduce factual details inaccurately, struggles with OOV words, repeating themselves
- Extractive is easier because copying ensures basic grammaticality and accuracy

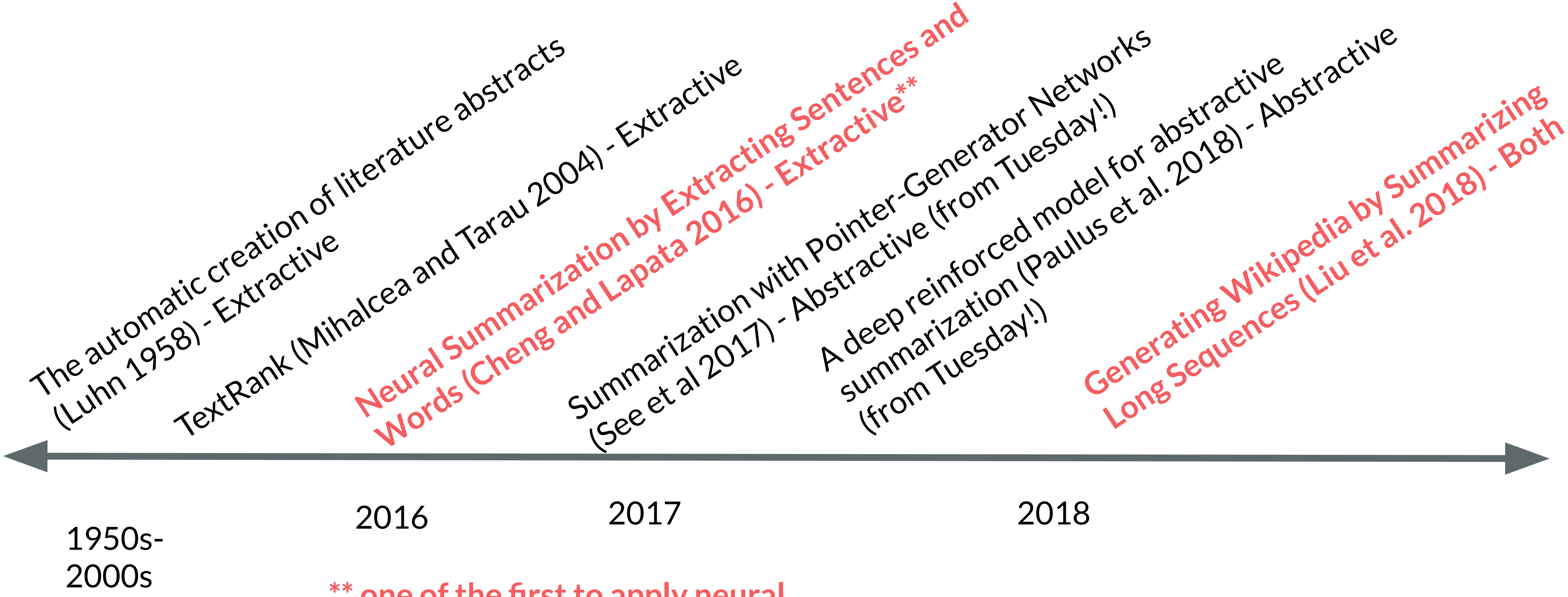
Summarization I (Tues) vs Today’s papers

- Tuesday’s papers:
    - Built on top of sequence-to-sequence for **abstractive** summarization of a single document
        - One introduced pointer-generator networks, other incorporated reinforcement learning
        - Produce a **short summary** of a news article
- Today’s papers:
    - First paper is purely **extractive** summarization
    - Second tries to scale up abstractive summarization to **long-text generation** with an extractive component



[1] Figure from Kushal Chauhan “Unsupervised Text Summarization using Sentence Embeddings”
https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1

Timeline



The automatic creation of literature abstracts (Luhn 1958) - Extractive

TextRank (Mihalcea and Tarau 2004) - Extractive

Neural Summarization by Extracting Sentences and Words (Cheng and Lapata 2016) - Extractive**

Summarization with Pointer-Generator Networks (See et al 2017) - Abstractive (from Tuesday!)

A deep reinforced model for abstractive summarization (Paulus et al. 2018) - Abstractive (from Tuesday!)

Generating Wikipedia by Summarizing Long Sequences (Liu et al. 2018) - Both

1950s-
2000s

2016

2017

2018

**** one of the first to apply neural methods for extractive summarization**

Neural Summarization by Extracting Sentences and Words

Cheng and Lapata 2016

Previous Extractive Approaches

- Identify sentences based on human-engineered features such as sentence position/length, words in title, word frequency
- Then score the sentences and select them using binary classifiers, graph algorithms

This work: Contributions

- Data-driven extractive approach **based on neural networks (NN)** rather than manually engineered features
    - NN-based hierarchical document reader/encoder
    - Attention-based content extractor
- Use DailyMail article highlights to make **large scale training dataset**

Problem Formulation: Sentence Extraction

- Given a document *D* consisting of a sequence of sentences $\{s_1, s_2, ..., s_m\}$ and a word set $\{w_1, w_2, ..., w_n\}$:

- **Sentence extraction** aims to create a summary from *D* by selecting and scoring a subset of *j* sentences predicting a label $y_L \in \{0, 1\}$ indicating whether the sentence should be included

Adelaide Crows defender Daniel Talia has kept his driving license, telling a court he was speeding 36km over the limit because he was distracted by his sick cat. The 22-year-old AFL star, who drove 96km/h in a 60km/h road works zone on the South Eastern expressway in February, said he didn’t see the reduced speed sign because he was so distracted by his cat vomiting violently in the back seat of his car. In the Adelaide magistrates court on Wednesday, Magistrate Bob Harrap fined Talia $824 for exceeding the speed limit by more than 30km/h. He lost four demerit points, instead of seven, because of his significant training commitments.

Summary produced with sentence extraction:

Adelaide Crows defender Daniel Talia has kept his driving license, telling a court he was speeding 36km over the limit because he was distracted by his sick cat. In the Adelaide magistrates court on Wednesday, Magistrate Bob Harrap fined Talia $824 for exceeding the speed limit by more than 30km/h.

Problem Formulation: Word Extraction

- Given a document *D* consisting of a sequence of sentences $\{s_1, s_2, ..., s_m\}$ and a word set $\{w_1, w_2, ..., w_n\}$:

- **Word extraction** aims to find a subset of words in *D* and their optimal ordering to form a summary

$$\mathbf{y}_s = (w'_1, \cdots, w'_k), w'_i \in D.$$

Adelaide Crows defender Daniel Talia has kept his driving license, telling a court he was speeding 36km over the limit because he was distracted by his sick cat. The 22-year-old AFL star, who drove 96km/h in a 60km/h road works zone on the South Eastern expressway in February, said he didn't see the reduced speed sign because he was so distracted by his cat vomiting violently in the back seat of his car. In the Adelaide magistrates court on Wednesday, Magistrate Bob Harrap fined Talia $824 for exceeding the speed limit by more than 30km/h. He lost four demerit points, instead of seven, because of his significant training commitments.

Summary produced with word extraction:
defender Daniel Talia was speeding distracted by his sick cat. didn't see reduced speed sign. Magistrate Bob Harrap fined Talia

Training Objective: Sentence Extraction

- Maximize the likelihood of all sentence labels $\mathbf{y}_L = (y_L^1, \cdots, y_L^m)$ given the input document *D* and model parameters θ:

$$\log p(\mathbf{y}_L|D;\theta) = \sum_{i=1}^{m} \log p(y_L^i|D;\theta)$$

Training Objective: Word Extraction

- Maximize the likelihood of the generated sentences, which can be further decomposed by enforcing conditional dependencies among their constituent words:

$$\log p(\mathbf{y}_s|D;\theta)=\sum_{i=1}^{k}\log p(w'_i|D,w'_1,\cdots,w'_{i-1};\theta)$$

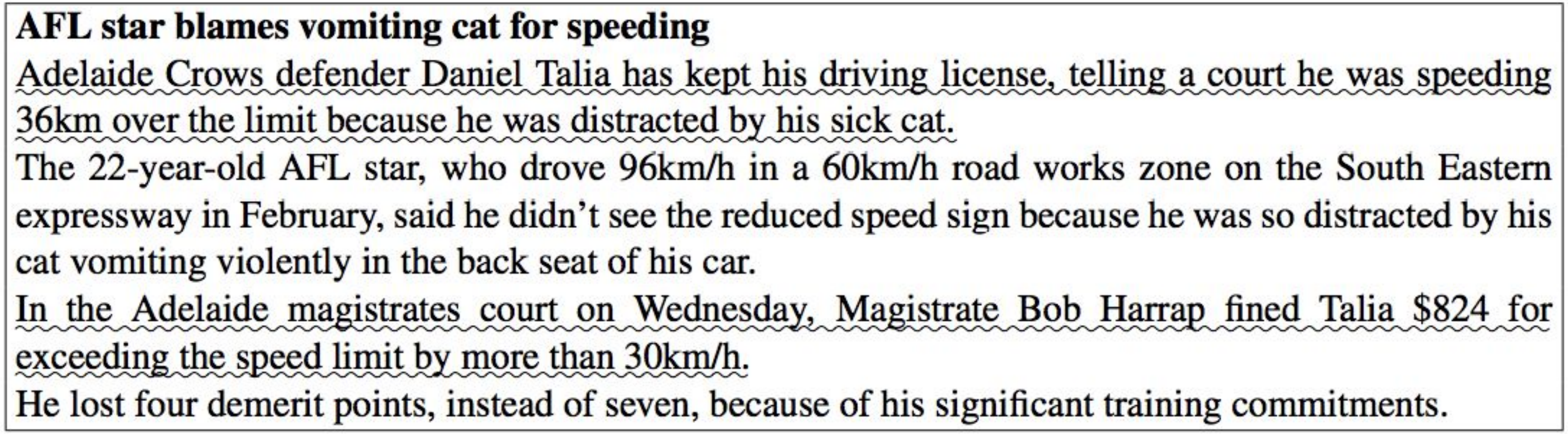
Word Extraction

- Most existing extractive approaches extract sentences
- Why word extraction?
    - Extractive summaries contain redundant info → word extraction could be middle ground between full abstractive summarization which can exhibit a wide range of rewrite operations and extractive which has none

Training Data

- Limitation: summarization training data
  - Existing dataset DUC-2002 only has 567 documents
- Create two large-scale datasets by reverse-approximating gold standard summary using DailyMail article highlights
  - Sentence Extraction (200K docs)
  - Word Extraction (170K docs)

DailyMail article



AFL star blames vomiting cat for speeding

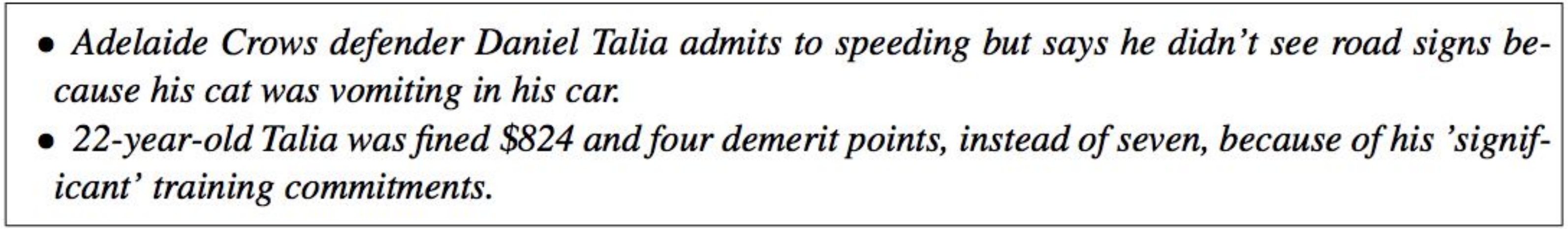
Adelaide Crows defender Daniel Talia has kept his driving license, telling a court he was speeding 36km over the limit because he was distracted by his sick cat.

The 22-year-old AFL star, who drove 96km/h in a 60km/h road works zone on the South Eastern expressway in February, said he didn’t see the reduced speed sign because he was so distracted by his cat vomiting violently in the back seat of his car.

In the Adelaide magistrates court on Wednesday, Magistrate Bob Harrap fined Talia $824 for exceeding the speed limit by more than 30km/h.

He lost four demerit points, instead of seven, because of his significant training commitments.

Highlights



- *Adelaide Crows defender Daniel Talia admits to speeding but says he didn’t see road signs because his cat was vomiting in his car.*
- *22-year-old Talia was fined $824 and four demerit points, instead of seven, because of his ’significant’ training commitments.*

Training Data: Sentence Extraction

- Designed a rule based system to determine whether a document sentence matches a highlight and should be in the gold-standard summary
- Rules take into account the position of the sentence, unigram and bigram overlap, number of entities
- Rule-based system was 85% accurate

Training Data: Word Extraction

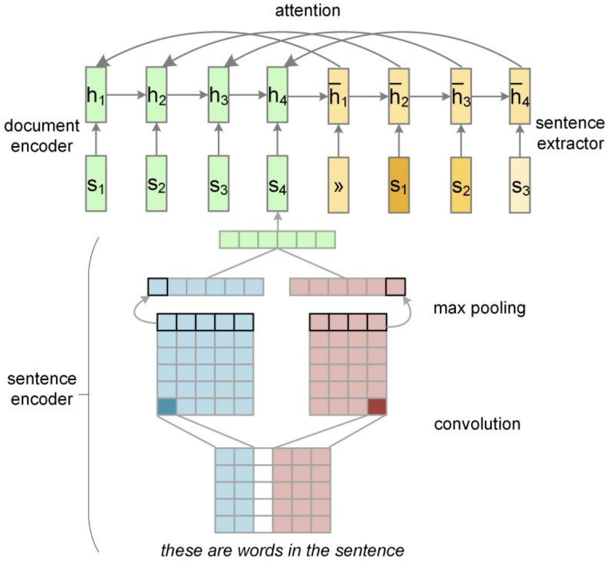
- Lexical overlap between highlights and news article
  - All highlight words come from original document → valid training example
  - For OOV words try to find semantically equivalent replacement in news article using pre-trained embeddings

Neural Summarization Model

- Document reader
- Sentence extractor
- Word extractor

Model: Document Reader

- **Hierarchical structure**: CNN at word level to acquire sentence-level representations → input to the RNN to acquire document level representations
  - **Convolutional sentence encoder**
  - **Recurrent document encoder**



Convolutional Sentence Encoder

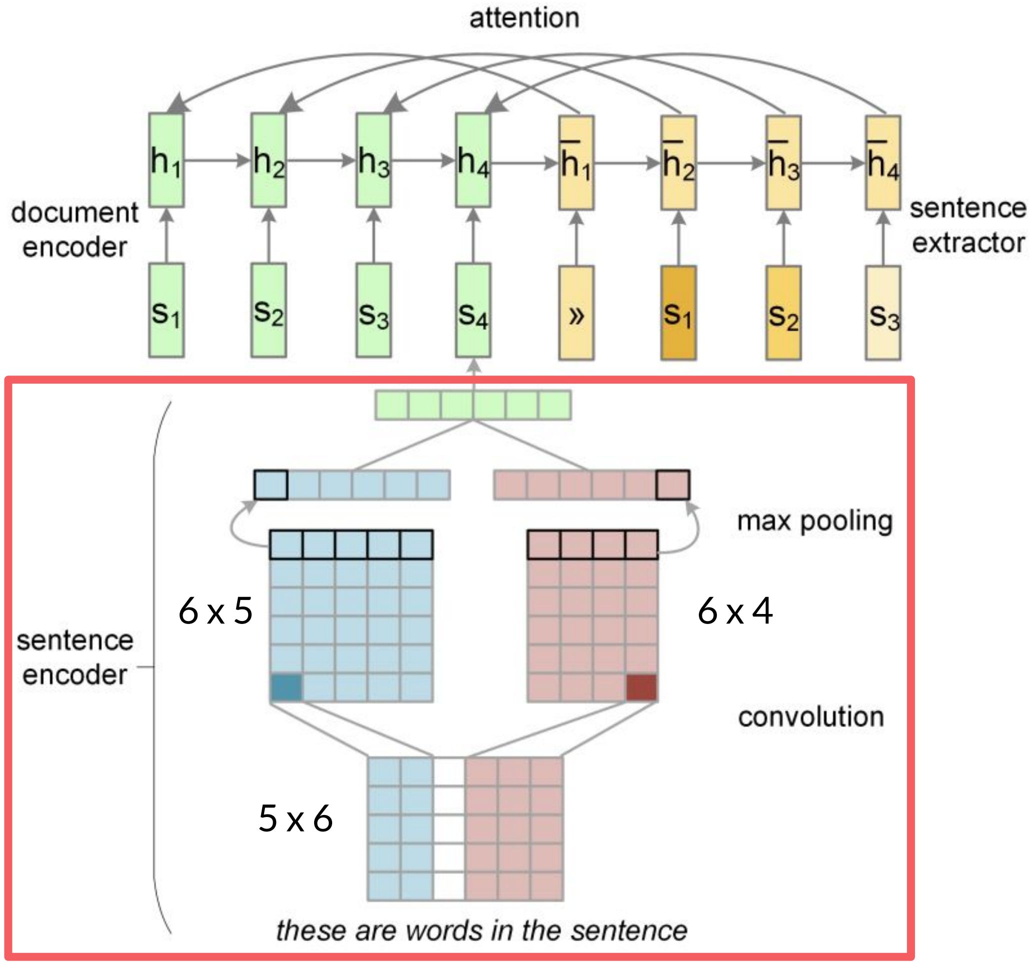
Setup:

Sentence has 6 words, word embeddings have 5 dimensions

Sentence embeddings have 6 dimensions → 6 feature maps per width

Blue feature maps have width 2 → 5 elements

Red feature maps have width 3 → 4 elements

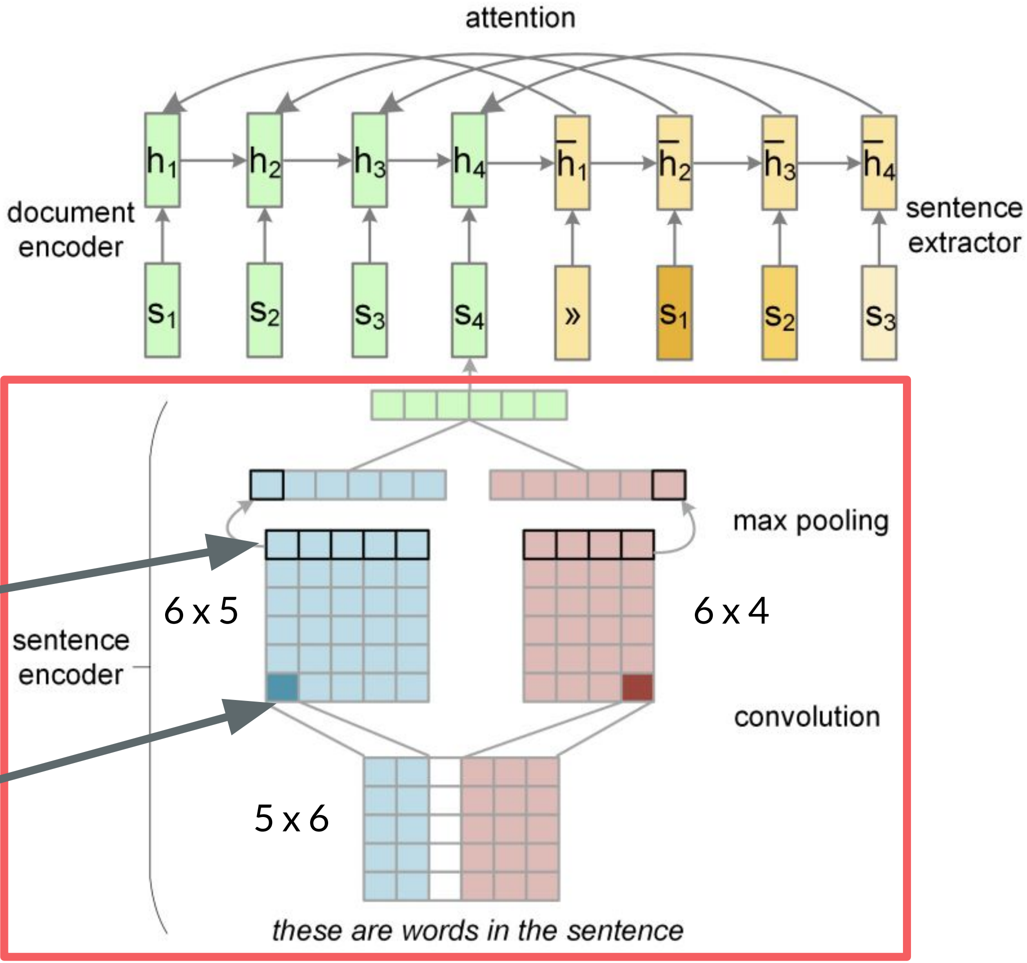


Convolutional Sentence Encoder

$\mathbf{f}^i_j$ is jth element of ith feature map, calculated by applying convolution between **W** (word embeddings) and kernel **K**

a feature map f^1 for width c=2

$$\mathbf{f}^i_j = \tanh(\mathbf{W}_{j:j+c-1} \otimes \mathbf{K} + b)$$



Hadamard Product

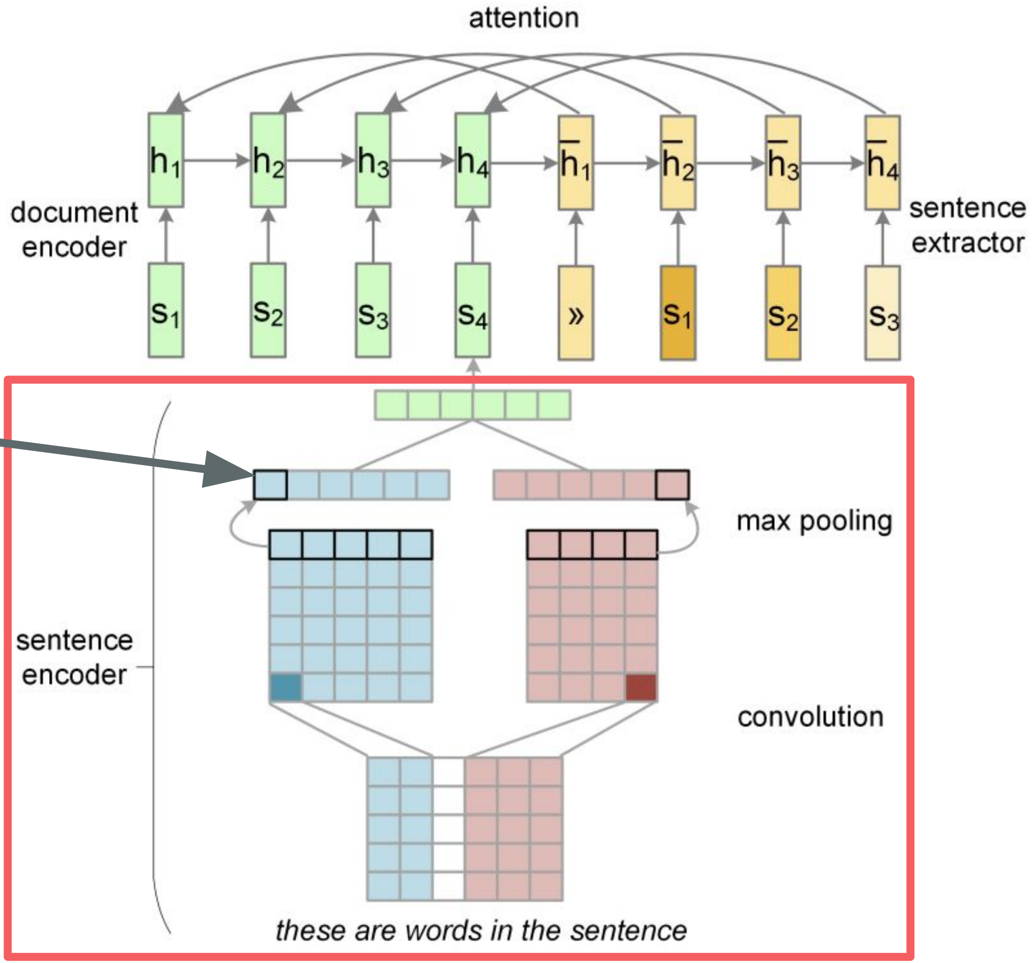
$$\mathbf{f}_j^i = \tanh(\mathbf{W}_{j:j+c-1} \otimes \mathbf{K} + b)$$

$$(A \circ B)_{ij} = (A \odot B)_{ij} = (A)_{ij}(B)_{ij}.$$

Convolutional Sentence Encoder

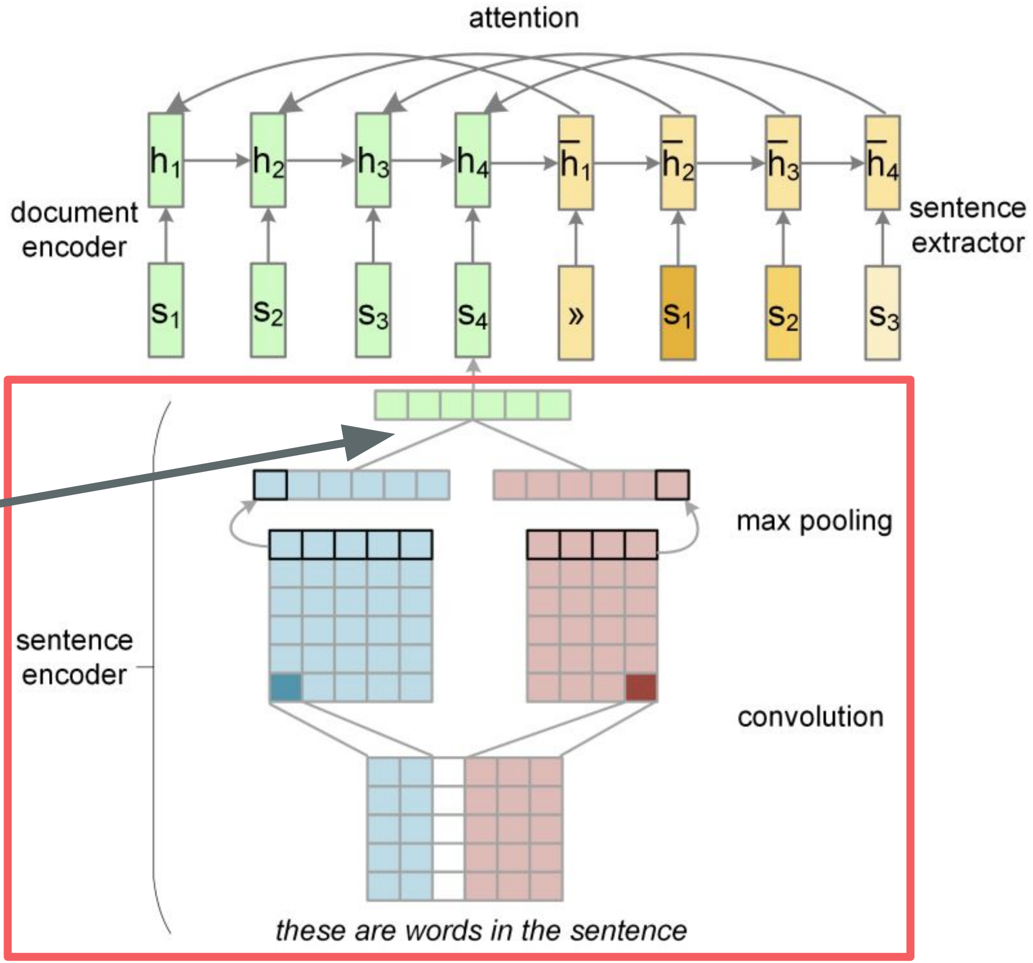
Max pooling over j to obtain the i-th feature $\mathbf{s}_{i,\mathbf{K}} = \max_{j} \mathbf{f}^{i}_{j}$

These features $s_{i,K}$ make up the sentence vector for width c



Convolutional Sentence Encoder

Sum these sentence vectors for different widths to obtain the final sentence representation

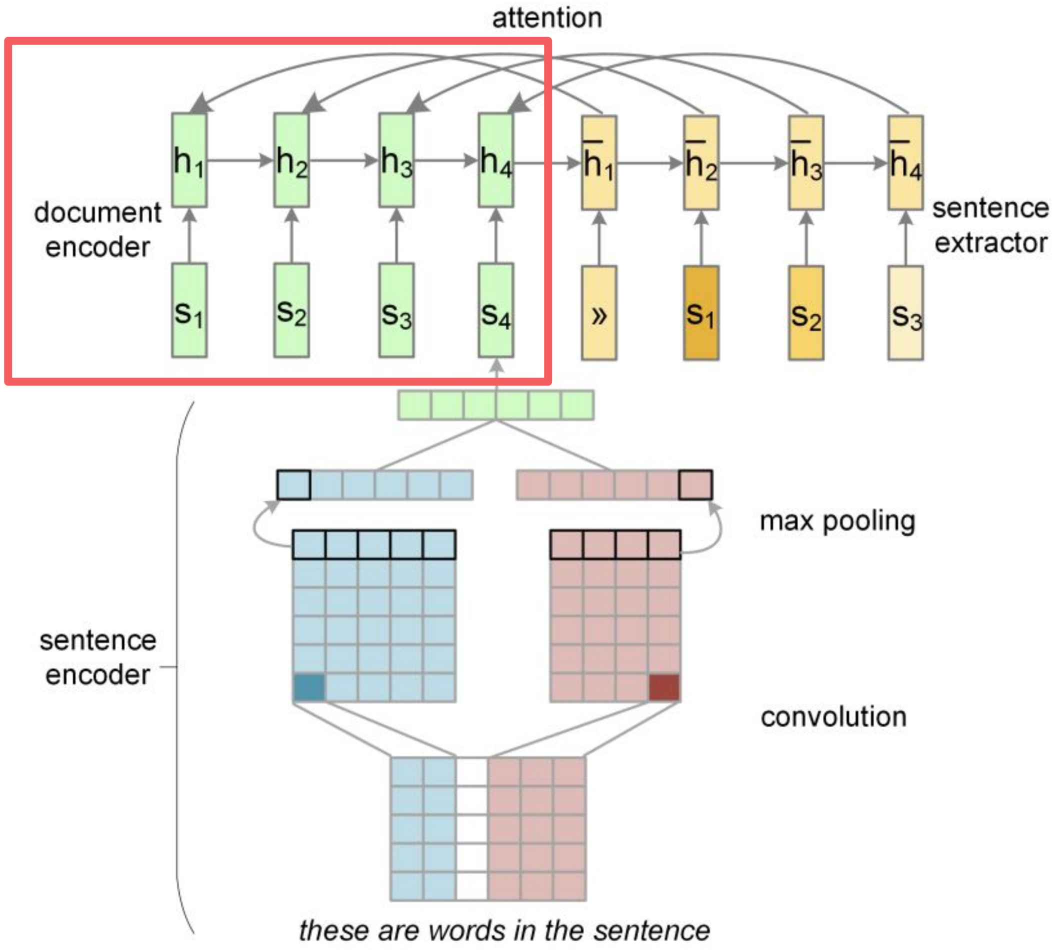

attention
document encoder
h_1 h_2 h_3 h_4 $\bar{h}_1$ $\bar{h}_2$ $\bar{h}_3$ $\bar{h}_4$
sentence extractor
s_1 s_2 s_3 s_4 » s_1 s_2 s_3
max pooling
sentence encoder
convolution
these are words in the sentence

Why CNN?

- More efficient than RNN
- Can be trained effectively without any long-term dependencies in the model
- Have been successfully used for sentence-level classification tasks

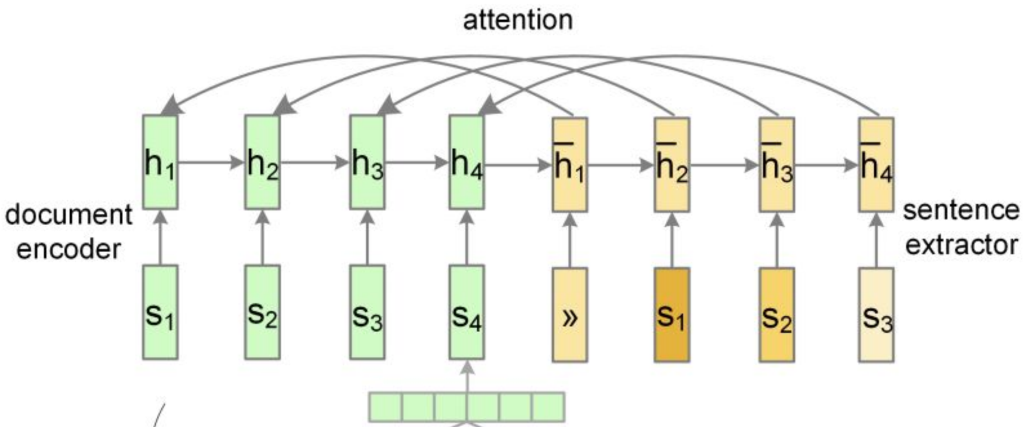
Recurrent Document Encoder

- Standard RNN
- Composes a sequence of sentence vectors into a document representation (h1 h2 h3 h4)
- Captures document organization at the level of sentence-to-sentence transitions



Model: Sentence Extractor

- RNN that labels sentences sequentially, applies attention, predicts a label for the next sentence at each time step



Model: Sentence Extractor

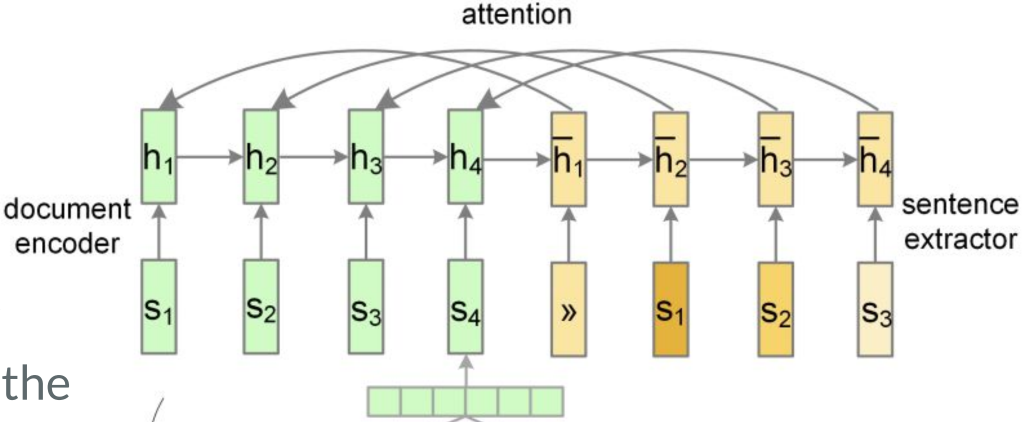
Given encoder hidden states

$$(h_1, \cdots, h_m)$$

and extractor hidden states

$$(\bar{h}_1, \cdots, \bar{h}_m)$$

decoder attends the t-th sentence by relating its current decoding state to the corresponding encoding state



$$\bar{\mathbf{h}}_t = \text{LSTM}(p_{t-1}\mathbf{s}_{t-1}, \bar{\mathbf{h}}_{t-1})$$

$$p(y_L(t) = 1|D) = \sigma(\text{MLP}(\bar{\mathbf{h}}_t : \mathbf{h}_t))$$

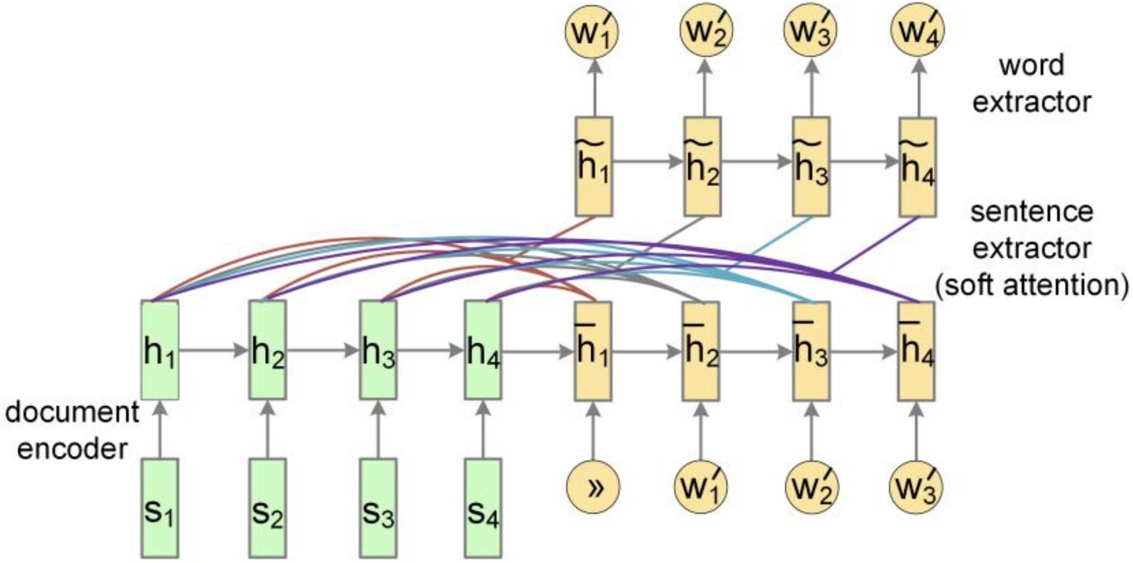
p_{t-1} represents the degree to which the extractor believes the previous sentence should be extracted and memorized

The sentence extraction model essentially regards the problem as sequence labeling: whether each sentence in the source document should be selected or not (labeled as 0 or 1). Why did they still adopt an encoder-decoder framework instead of a more direct sequence tagger model?

- Sequence tagging doesn't have **long term dependencies** or require context from what was previously tagged
- Choosing a sentence for a summary is **not independent** of the other sentences' labels since **redundancy matters**, better to treat all sentences as a whole
    - Next labeling decision made with both the encoded document and previously labeled sentences in mind

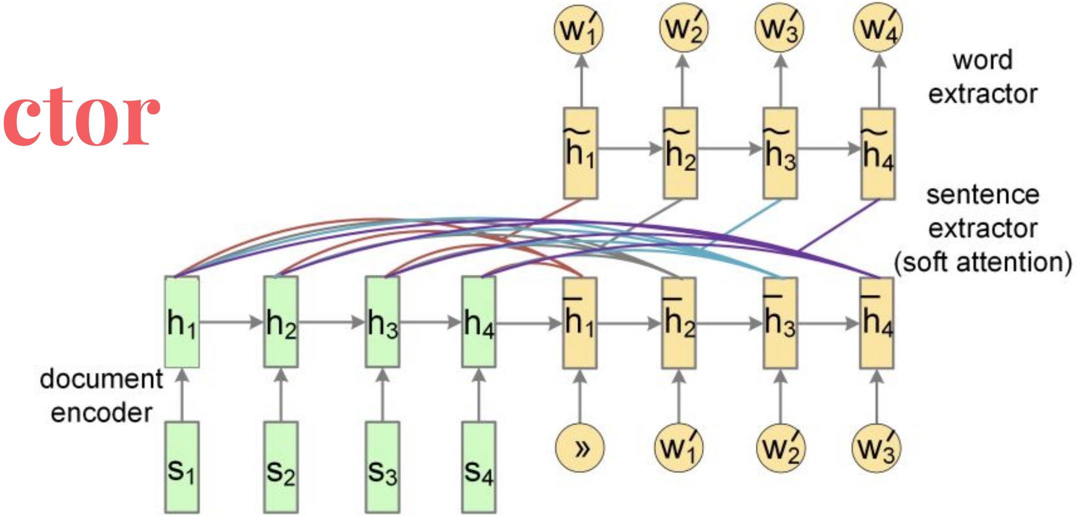
Model: Word Extractor

- Generation task instead of sequence labeling - instead of predicting label for next sentence, output next word in summary



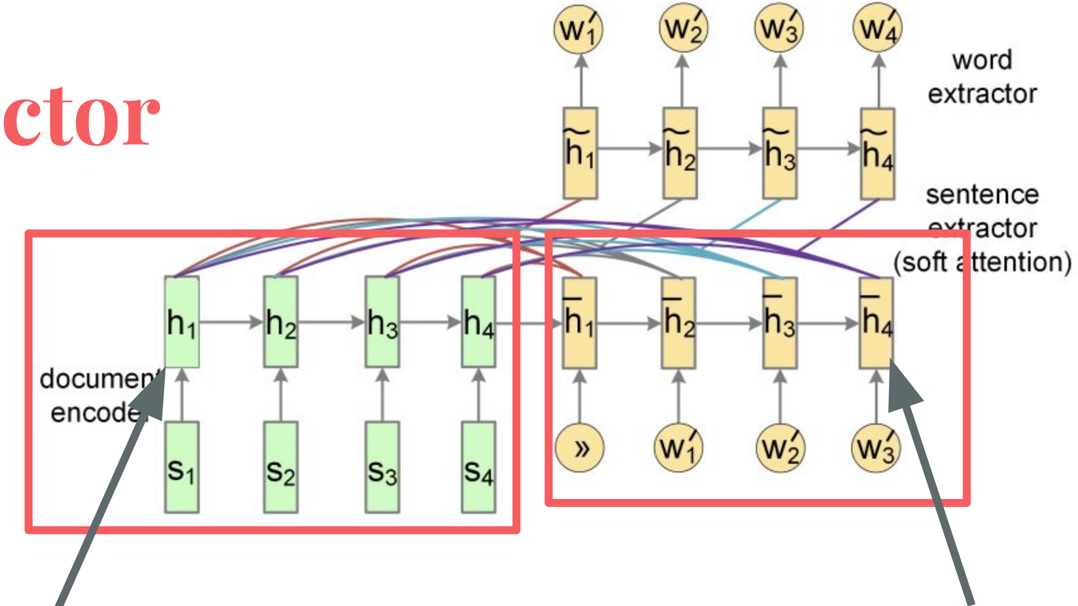
Model: Word Extractor

- **Hierarchical attention architecture**: at time step t:
  - the decoder softly attends each sentence and subsequently each word in the document
  - computes probability of the next word to be included in the summary with a softmax classifier



Model: Word Extractor

- **Hierarchical attention architecture**: at time step t:
  - the decoder softly attends each sentence and subsequently each word in the document
  - computes probability of the next word to be included in the summary with a softmax classifier



Encoder hidden states (h1 h2 ...)

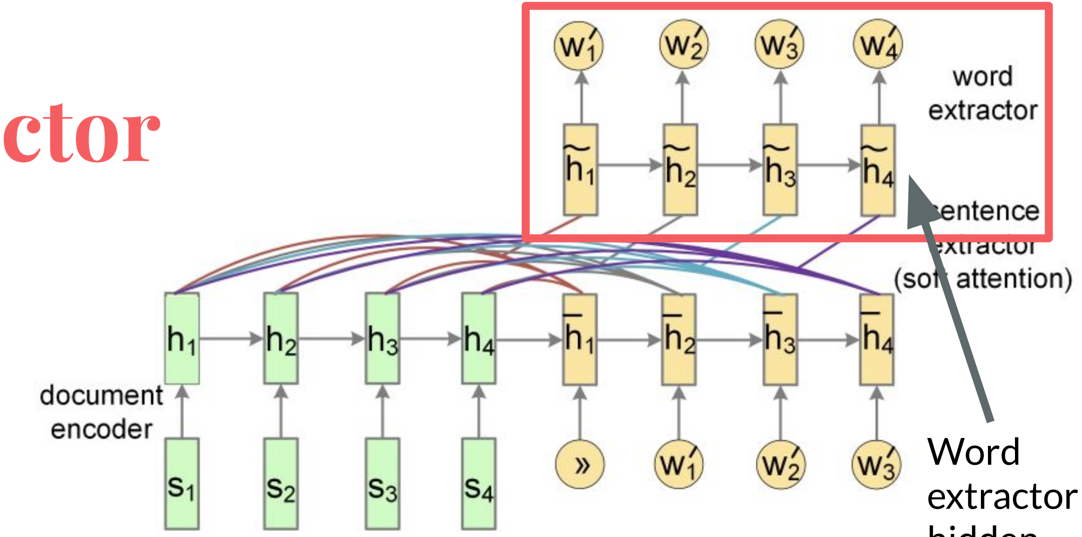
Sentence extractor hidden states ($\bar{h}$1 $\bar{h}$2 ...)

$$\bar{\mathbf{h}}_t = \mathrm{LSTM}(\mathbf{w'}_{t-1}, \bar{\mathbf{h}}_{t-1})^6$$

$$a_j^t = \mathbf{z}^{\mathrm{T}} \tanh(\mathbf{W}_e \bar{\mathbf{h}}_t + \mathbf{W}_r \mathbf{h}_j), h_j \in D$$

Model: Word Extractor

- **Hierarchical attention architecture**: at time step t:
  - the decoder softly attends each sentence **and subsequently each word in the document**
  - computes probability of the next word to be included in the summary with a softmax classifier



Word extractor hidden states (ĥ1 ĥ2 ...)

$$a^t_j = \mathbf{z}^{\mathrm{T}} \tanh(\mathbf{W}_e \bar{\mathbf{h}}_t + \mathbf{W}_r \mathbf{h}_j), h_j \in D$$

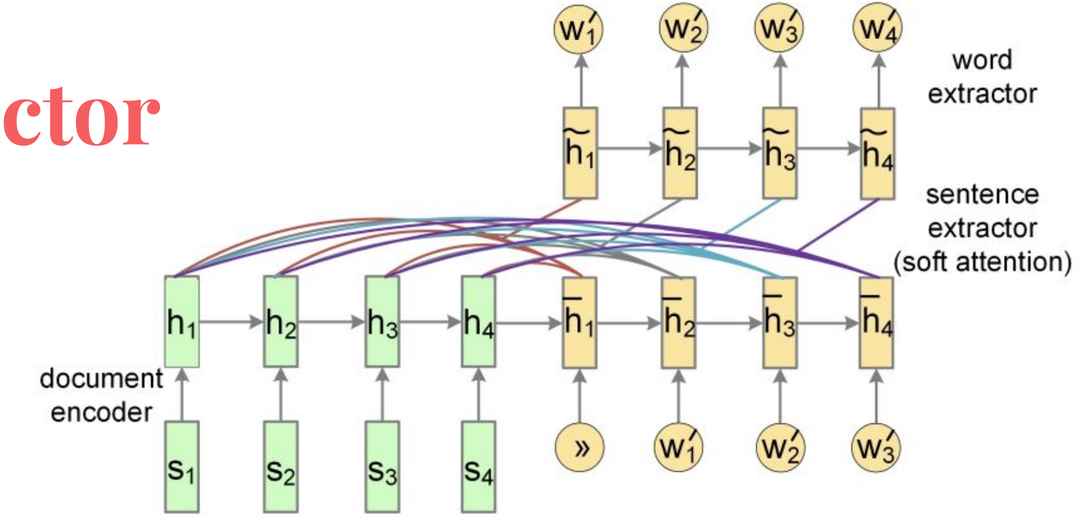
$$b^t_j = \operatorname{softmax}(a^t_j)$$

$$\tilde{\mathbf{h}}_t = \sum_{j=1}^{m} b^t_j \mathbf{h}_j$$

$$u^t_i = \mathbf{v}^{\mathrm{T}} \tanh(\mathbf{W}_{e'} \tilde{\mathbf{h}}_t + \mathbf{W}_{r'} \mathbf{w}_i), w_i \in D$$

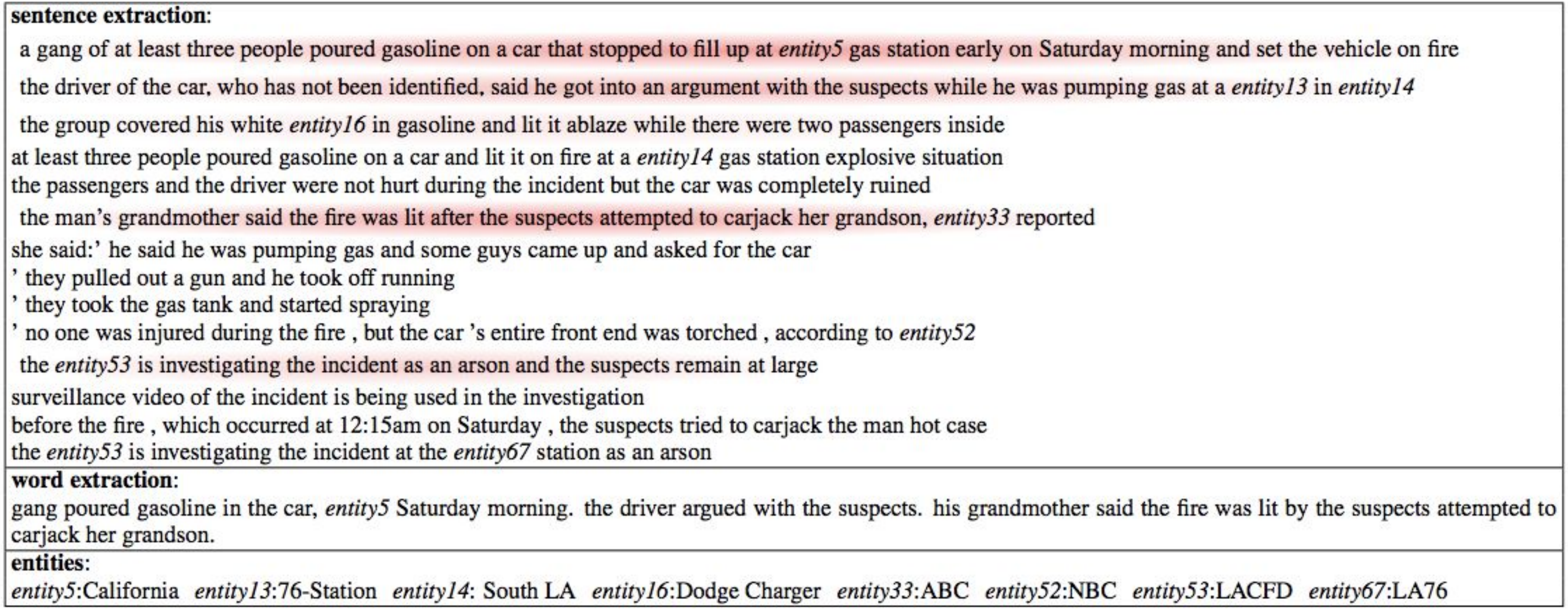
Model: Word Extractor

- **Hierarchical attention architecture**: at time step t:
  - the decoder softly attends each sentence and subsequently each word in the document
  - **computes probability of the next word to be included in the summary with a softmax classifier**



$$u_i^t = \mathbf{v}^{\mathrm{T}} \tanh(\mathbf{W}_{e'} \tilde{\mathbf{h}}_t + \mathbf{W}_{r'} \mathbf{w}_i), w_i \in D$$

$$p(w'_t = w_i | D, w'_1, \cdots, w'_{t-1}) = \mathrm{softmax}(u_i^t)$$



sentence extraction:

a gang of at least three people poured gasoline on a car that stopped to fill up at *entity5* gas station early on Saturday morning and set the vehicle on fire

the driver of the car, who has not been identified, said he got into an argument with the suspects while he was pumping gas at a *entity13* in *entity14*

the group covered his white *entity16* in gasoline and lit it ablaze while there were two passengers inside

at least three people poured gasoline on a car and lit it on fire at a *entity14* gas station explosive situation

the passengers and the driver were not hurt during the incident but the car was completely ruined

the man's grandmother said the fire was lit after the suspects attempted to carjack her grandson, *entity33* reported

she said:' he said he was pumping gas and some guys came up and asked for the car

' they pulled out a gun and he took off running

' they took the gas tank and started spraying

' no one was injured during the fire , but the car 's entire front end was torched , according to *entity52*

the *entity53* is investigating the incident as an arson and the suspects remain at large

surveillance video of the incident is being used in the investigation

before the fire , which occurred at 12:15am on Saturday , the suspects tried to carjack the man hot case

the *entity53* is investigating the incident at the *entity67* station as an arson

word extraction:

gang poured gasoline in the car, *entity5* Saturday morning. the driver argued with the suspects. his grandmother said the fire was lit by the suspects attempted to carjack her grandson.

entities:

entity5:California *entity13*:76-Station *entity14*: South LA *entity16*:Dodge Charger *entity33*:ABC *entity52*:NBC *entity53*:LACFD *entity67*:LA76

Figure 4: Visualization of the summaries for a DailyMail article. The top half shows the relative attention weights given by the sentence extraction model. Darkness indicates sentence importance. The lower half shows the summary generated by the word extraction.

Experimental Setup

- Proper nouns: named entity recognition
- Number of sentences to extract - use 3, relative ranking
- Compare to lead-3 sentences, logistic and human engineered feature classifier, neural abstractive baseline, 3 previous systems

Models

This paper

- **NN-SE**: sentence extraction
- **NN-WE**: word extraction

Baselines

- **NN-ABS** (*Rush et al.* 2015): neural abstractive baseline
- **LEAD**: first 3 sentences
- **LREG**: logistic regression
- **ILP** (*Woodsend and Lapata* 2010): phrase-based constraints
- **URANK** (*Wan* 2010): graph-based sentence ranking
- **TGRAPH** (*Parveen et al.* 2015): graph-based using topic models

Evaluation

- ROUGE scores
- Human judgement

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

- Compare an automatically produced summary against a reference or a set of references (human-produced) summary
- ROUGE-N: Overlap of N-grams between the system and reference summaries
- ROUGE-L: Longest Common Subsequence (LCS) based

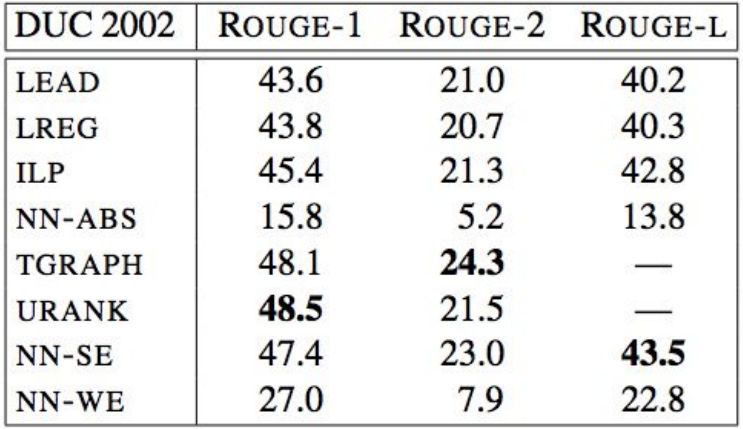
Human Judgement

- Mechanical Turk participants asked to rank a set of summaries in order of informativeness and fluency for randomly sampled news articles
- Set of summaries included NN-SE, NN-WE, baselines and the human authored summary

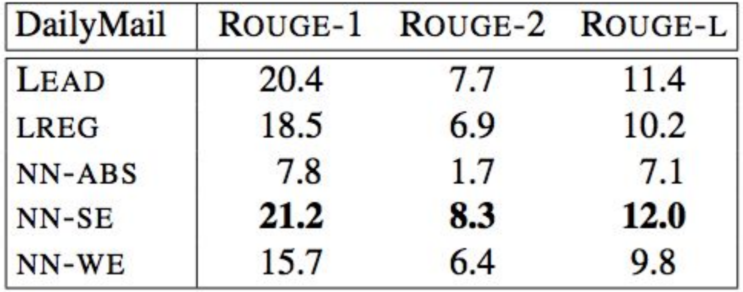
Results: ROUGE

NN and LREG models trained on DailyMail news set and evaluated on DUC-2002 and DailyMail

TGRAPH, URANK, ILP from previously published results

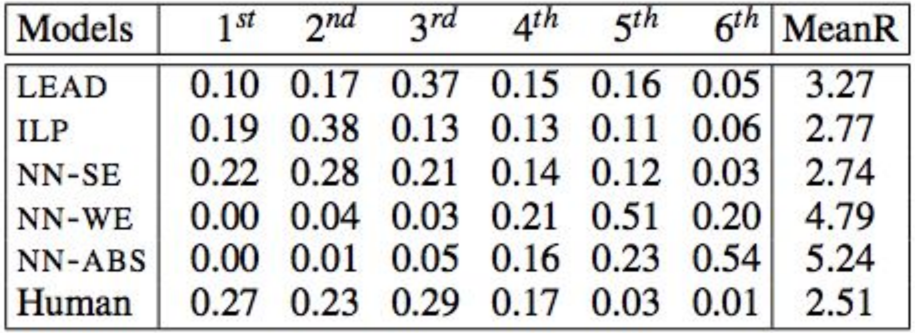


DUC 2002	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	43.6	21.0	40.2
LREG	43.8	20.7	40.3
ILP	45.4	21.3	42.8
NN-ABS	15.8	5.2	13.8
TGRAPH	48.1	**24.3**	—
URANK	**48.5**	21.5	—
NN-SE	47.4	23.0	**43.5**
NN-WE	27.0	7.9	22.8



DailyMail	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	20.4	7.7	11.4
LREG	18.5	6.9	10.2
NN-ABS	7.8	1.7	7.1
NN-SE	**21.2**	**8.3**	**12.0**
NN-WE	15.7	6.4	9.8

Results: Human Judgement



Models	1st	2nd	3rd	4th	5th	6th	MeanR
LEAD	0.10	0.17	0.37	0.15	0.16	0.05	3.27
ILP	0.19	0.38	0.13	0.13	0.11	0.06	2.77
NN-SE	0.22	0.28	0.21	0.14	0.12	0.03	2.74
NN-WE	0.00	0.04	0.03	0.21	0.51	0.20	4.79
NN-ABS	0.00	0.01	0.05	0.16	0.23	0.54	5.24
Human	0.27	0.23	0.29	0.17	0.03	0.01	2.51

Results

- NN-SE does well in ROUGE score
- NN-WE does less well because ROUGE not suited to paraphrasing but does better than NN-ABS
- Human evaluation - human summaries best, then NN-SE

Takeaways

- **Hierarchical neural structures** that reflect the nature of the summarization task
- **Generation by extraction**
- **Large-scale dataset** using DailyMail highlights

Generating Wikipedia by Summarizing Long Sequences

Liu et al., 2018

Previous Abstractive Approaches

- Datasets/tasks:
  - Gigaword (Graff & Cieri, 2003): used for sentence to headline generation pioneered in (Rush et al., 2015)
  - CNN/DailyMail (Nallapati et al., 2016): news article to story highlights
- Models:
  - RNN-based models that mirrored MT techniques
  - Transformer encoder-decoder models (Vaswani et al., 2017)

This Work: Contributions

- Generating English Wikipedia articles as a multi-document summarization of source documents
- Model: a decoder-only Transformer architecture that can scalably attend to very long sequences
    - Handles input length of 11,000 words
- Dataset: WikiSum dataset is orders-of-magnitude larger than previous summarization datasets in terms of **input/output length**

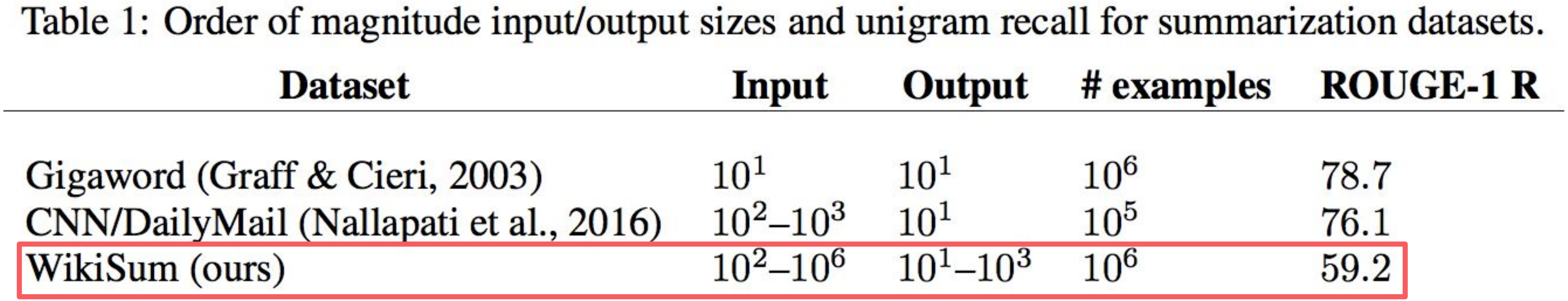


Table 1: Order of magnitude input/output sizes and unigram recall for summarization datasets.

Dataset	Input	Output	# examples	ROUGE-1 R
Gigaword (Graff & Cieri, 2003)	10^1	10^1	10^6	78.7
CNN/DailyMail (Nallapati et al., 2016)	10^2–10^3	10^1	10^5	76.1
WikiSum (ours)	10^2–10^6	10^1–10^3	10^6	59.2

Problem Formulation

- Supervised machine learning task
    - Input: Wikipedia topic (article title) + collection of non-Wikipedia reference documents
    - Target: Wikipedia article text

Dataset

- English Wikipedia as a multi-document summarization dataset:
  - Wikipedia = a collection of summaries given by title
  - All reputable documents = source material
    - Sources cited in references section of Wikipedia articles
    - Top 10 web search results with low level of unigram overlap with target article

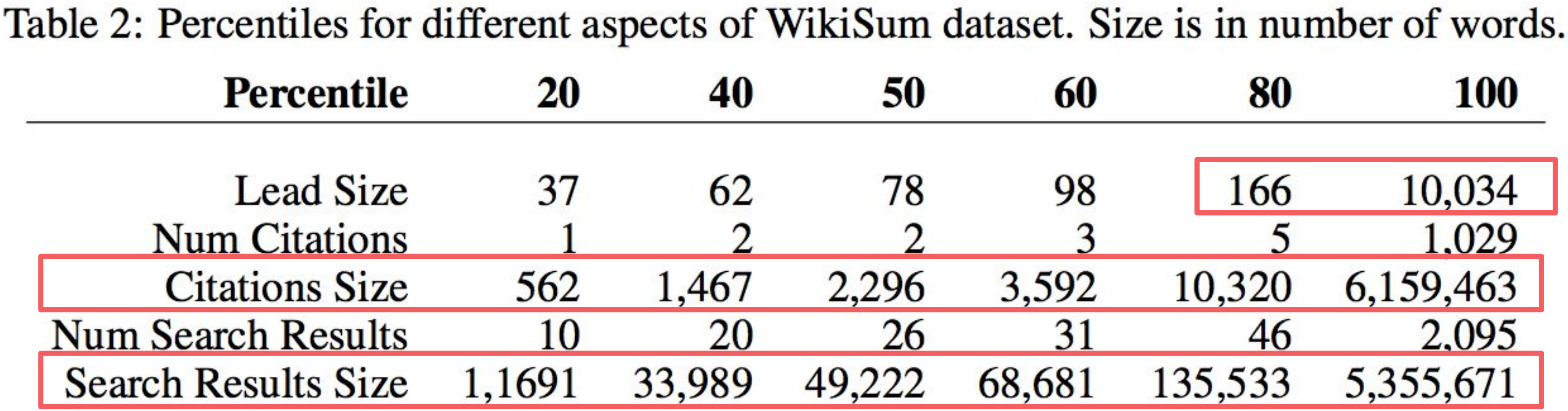


Table 2: Percentiles for different aspects of WikiSum dataset. Size is in number of words.

Percentile	**20**	**40**	**50**	**60**	**80**	**100**
Lead Size	37	62	78	98	166	10,034
Num Citations	1	2	2	3	5	1,029
Citations Size	562	1,467	2,296	3,592	10,320	6,159,463
Num Search Results	10	20	26	31	46	2,095
Search Results Size	1,1691	33,989	49,222	68,681	135,533	5,355,671

Methods

- Stage 1: Extractive summarization
  - Select a subset of the (very large) input
- Stage 2: Abstractive summarization
  - Train abstractive model that generates Wikipedia text by conditioning on the output of the extractive stage

Extractive Stage

- Investigated 5 extractive methods that aim to return L tokens as the input to the abstractive stage:
    - **Identity**
    - **Tf-idf**
    - **Cheating**
    - TextRank (Mihalcea & Tarau, 2004): rank paragraphs using similarity measure based on word overlap
    - SumBasic (Nenkova & Vanderwede, 2005): rank sentences by assigning scores to words using word frequencies in input text

Extractive Stage (cont.)

- Identity: use first L tokens of input
- Tf-idf: rank paragraphs as documents in a query-retrieval problem

$$N_w \cdot log(\frac{N_d}{N_{dw}})$$

N_w = count of the word in the document
N_d = total number of documents
N_{dw} = total number of documents containing the word

- Cheating: rank paragraphs using recall of bigrams in ground truth text

$$d(p_j^i, a_i) = \frac{bigrams(p_j^i) \cap bigrams(a_i)}{bigrams(a_i)}$$

a_i = article
p^i_j = j^{th} paragraph of i^{th} input reference document

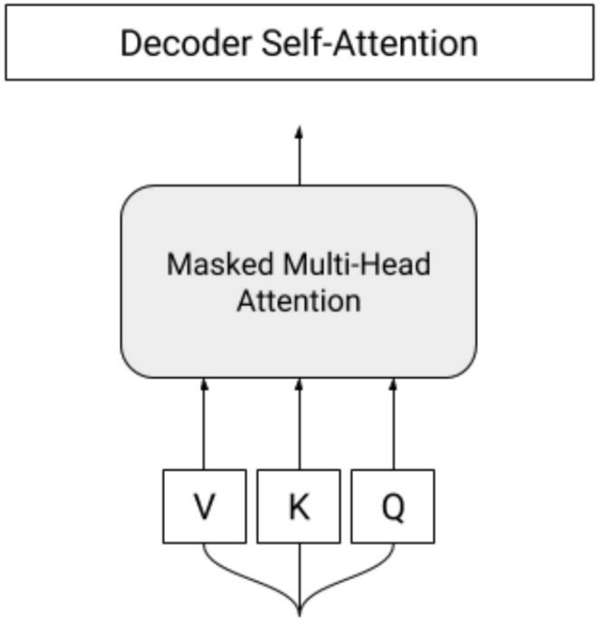
Abstractive Stage

- Input: title + concatenation of ordered paragraphs
- Output: Wikipedia lead text
- Formulated as a sequence transduction problem:
  very long input sequences (<= 11,000) → medium output sequences (<500)

Abstractive Stage (cont.)

- Models tested:
    - Standard LSTM encoder-decoder with attention (seq2seq-att)
    - Transformer encoder-decoder (T-ED)
    - **Transformer decoder (T-D)**
    - **Transformer decoder with memory-compressed attention (T-DMCA)**

Abstractive Stage: T-D



- Transformer decoder (T-D):
  - Drop the encoder module
  - Combine input and output sequences into a single “sentence”
  - Train as a standard language model

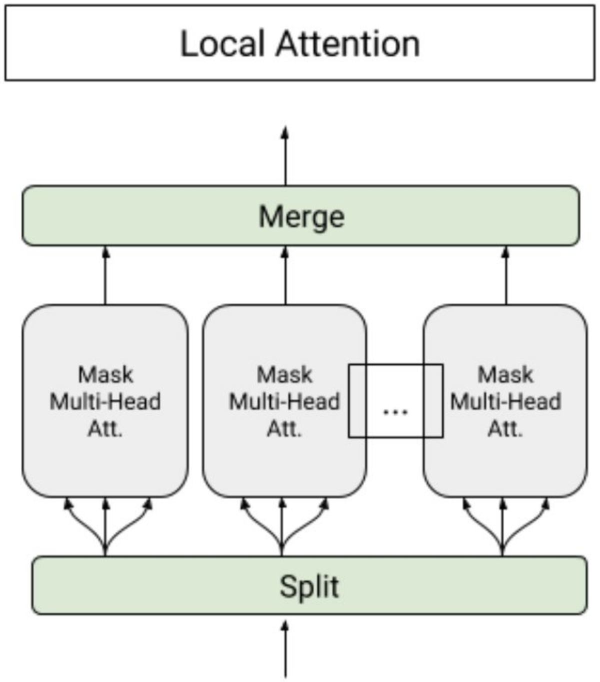
Abstractive Stage: T-DMCA

- T-D with memory-compressed attention (T-DMCA)
  - Motivation: reduce memory usage to handle longer sequences
  - Modify multi-head self-attention of Transformer

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

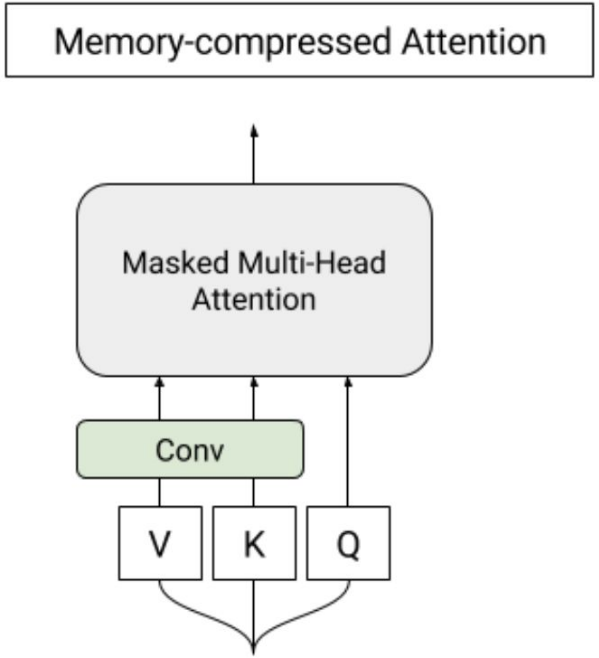
  - Consists of two kinds of attention layers:
    - Local attention
    - Memory-compressed attention

Abstractive Stage: T-DMCA



- Local attention: perform attention individually within a block of 256 sequence tokens
  - Attention memory cost per block becomes constant
  - Allows the number of activations to stay linear with respect to the sequence length

Abstractive Stage: T-DMCA



- Memory-compressed attention: exchange information globally on the entire sequence
    - Project tokens into query, key, value embeddings
    - Use strided convolution to reduce number of keys and values

Abstractive Stage: T-DMCA

- Final architecture: 5-layer network (LMLML) alternating between local-attention (L) layers and memory-compressed attention (M) layers
- Mixture of experts (MoE) layer (Shazeer et al., 2017) to increase the network's capacity

Evaluation Metrics

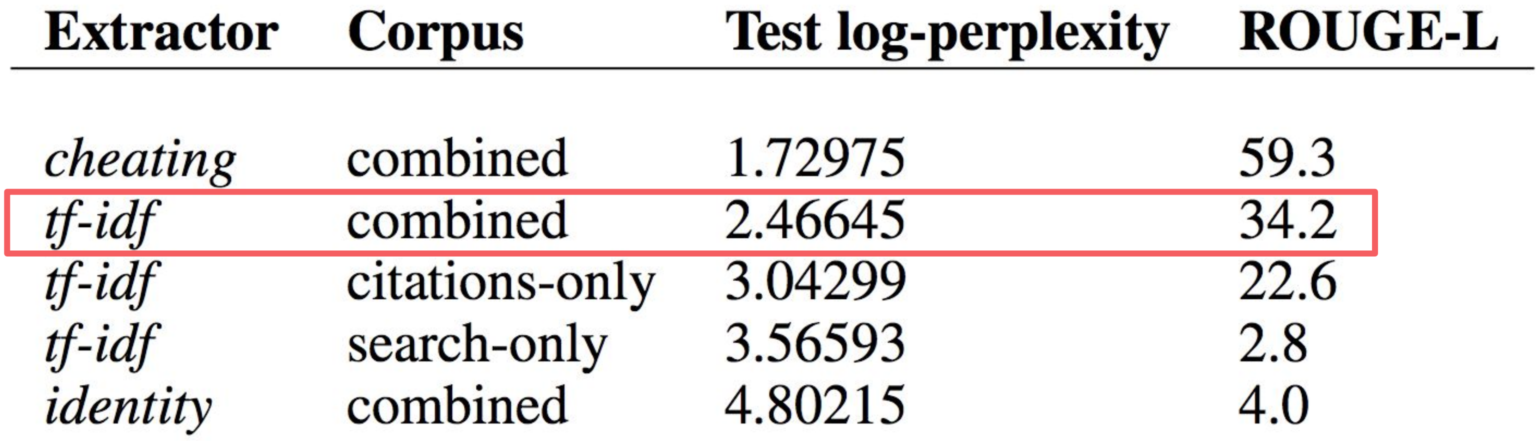
- Perplexity
  - Low perplexity indicates the probability distribution is good at predicting the sample
- ROUGE-L F1

Experiments

- Varied along 4 dimensions:
    - Extractive method: SumBasic, TextRank, tf-idf, identity, cheating extractor
    - Input corpus: citations, search results, combined
    - Abstractive model input length, L: values between 100 and 11000
    - Abstractive model architecture: seq2seq-att, T-ED, T-D, T-DMCA

Results

- Extractive-only is not enough
- Extractive method matters



Extractor	Corpus	Test log-perplexity	ROUGE-L
cheating	combined	1.72975	59.3
tf-idf	combined	2.46645	34.2
tf-idf	citations-only	3.04299	22.6
tf-idf	search-only	3.56593	2.8
identity	combined	4.80215	4.0

- Input corpus: Combined dataset (cited sources + search results) performs best

Results (cont.)

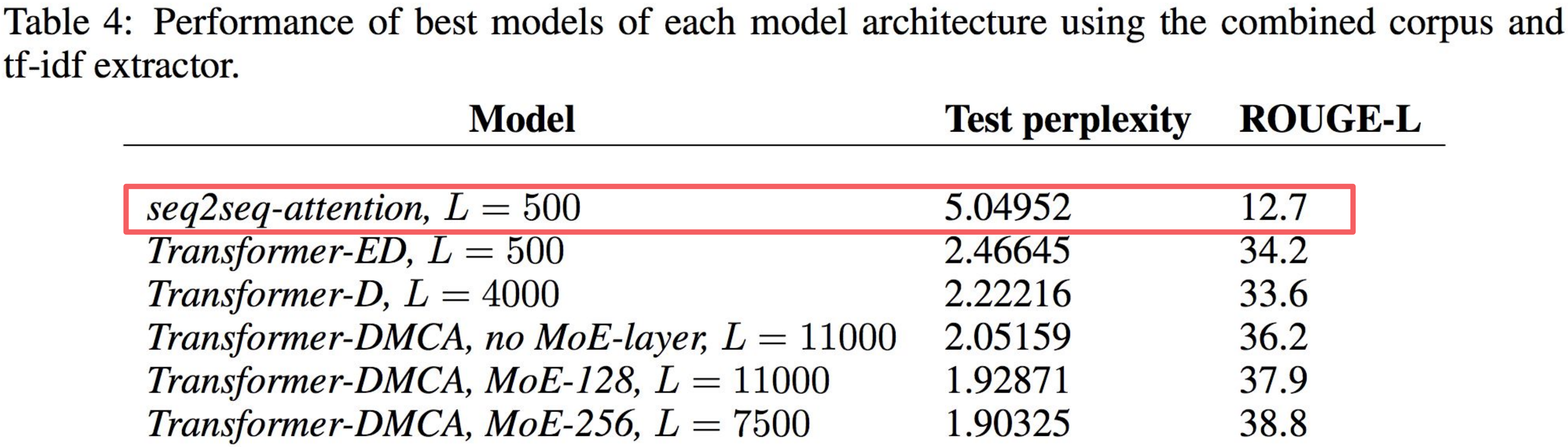


Table 4: Performance of best models of each model architecture using the combined corpus and tf-idf extractor.

Model	Test perplexity	ROUGE-L
seq2seq-attention, $L = 500$	5.04952	12.7
Transformer-ED, $L = 500$	2.46645	34.2
Transformer-D, $L = 4000$	2.22216	33.6
Transformer-DMCA, no MoE-layer, $L = 11000$	2.05159	36.2
Transformer-DMCA, MoE-128, $L = 11000$	1.92871	37.9
Transformer-DMCA, MoE-256, $L = 7500$	1.90325	38.8

Results (cont.)

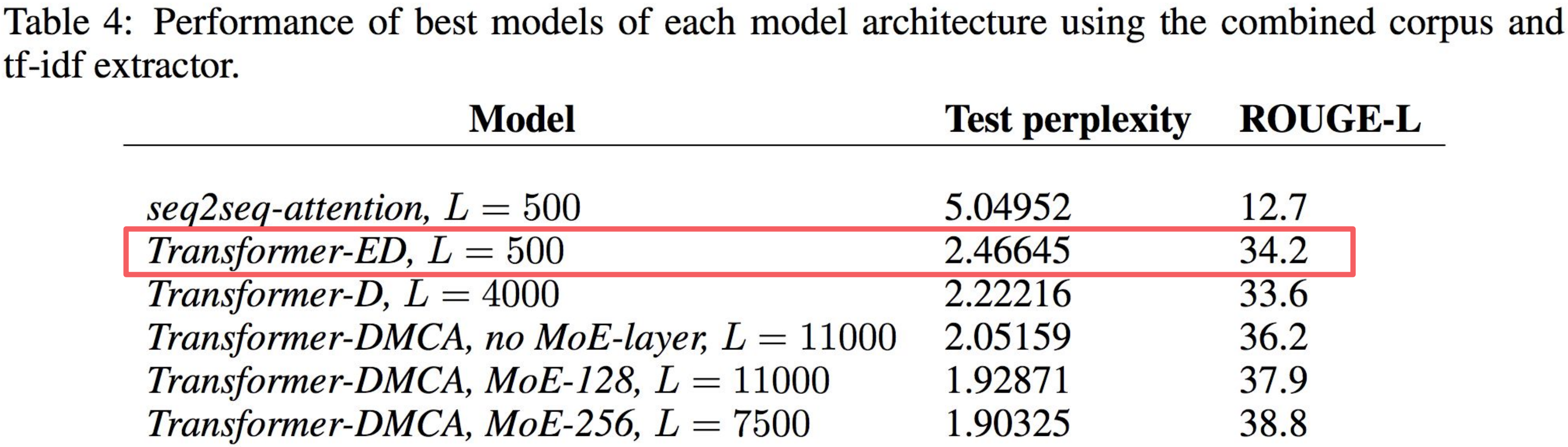


Table 4: Performance of best models of each model architecture using the combined corpus and tf-idf extractor.

Model	**Test perplexity**	**ROUGE-L**
seq2seq-attention, $L = 500$	5.04952	12.7
Transformer-ED, $L = 500$	2.46645	34.2
Transformer-D, $L = 4000$	2.22216	33.6
Transformer-DMCA, no MoE-layer, $L = 11000$	2.05159	36.2
Transformer-DMCA, MoE-128, $L = 11000$	1.92871	37.9
Transformer-DMCA, MoE-256, $L = 7500$	1.90325	38.8

Results (cont.)

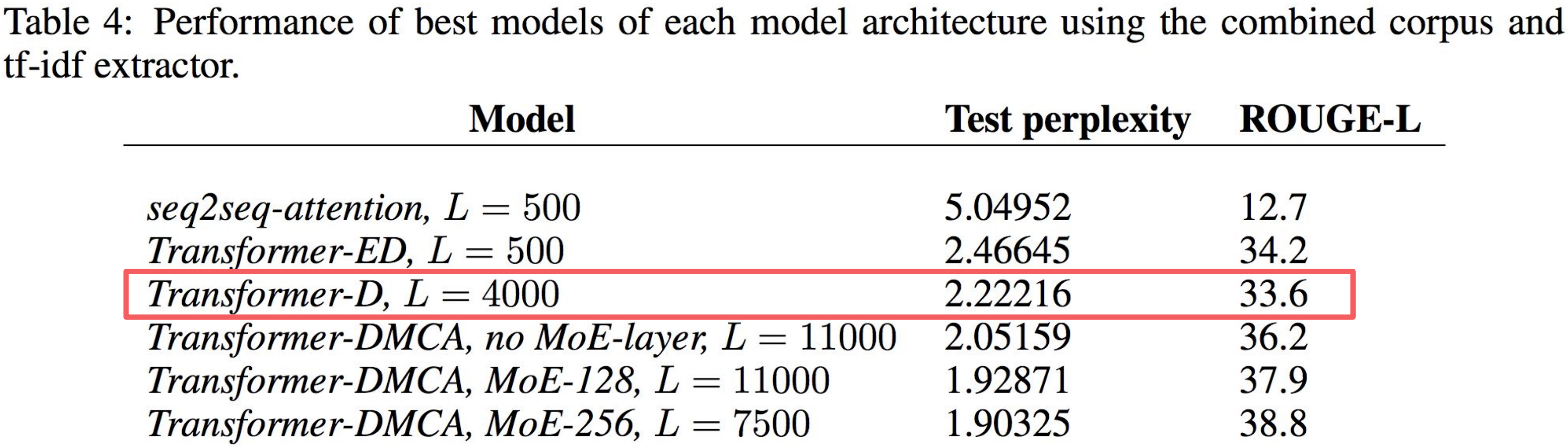


Table 4: Performance of best models of each model architecture using the combined corpus and tf-idf extractor.

Model	Test perplexity	ROUGE-L
seq2seq-attention, $L = 500$	5.04952	12.7
Transformer-ED, $L = 500$	2.46645	34.2
Transformer-D, $L = 4000$	2.22216	33.6
Transformer-DMCA, no MoE-layer, $L = 11000$	2.05159	36.2
Transformer-DMCA, MoE-128, $L = 11000$	1.92871	37.9
Transformer-DMCA, MoE-256, $L = 7500$	1.90325	38.8

Results (cont.)

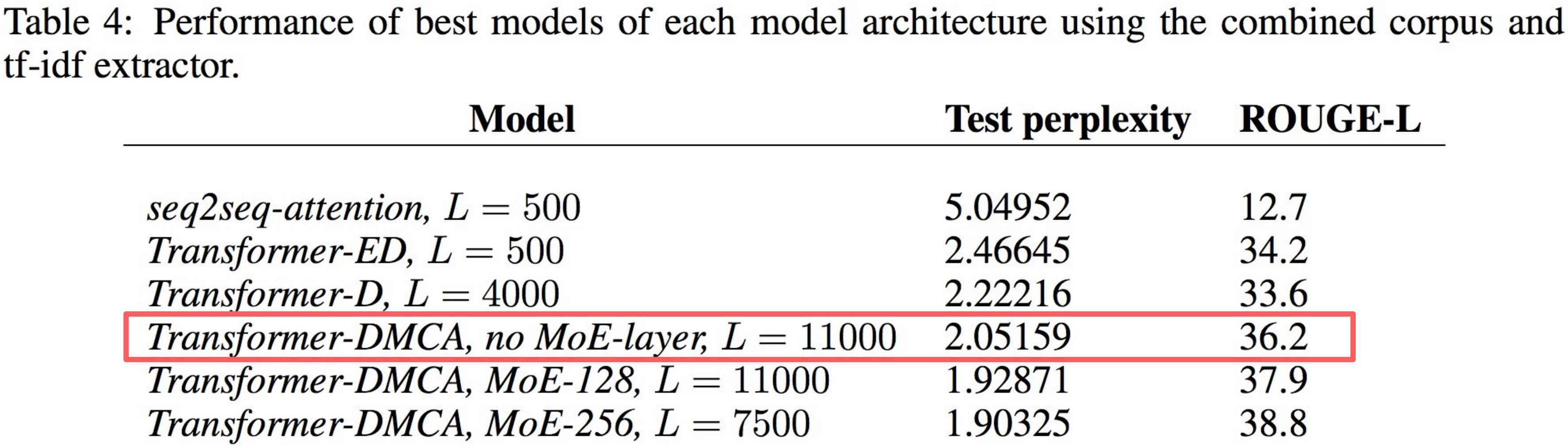


Table 4: Performance of best models of each model architecture using the combined corpus and tf-idf extractor.

Model	**Test perplexity**	**ROUGE-L**
seq2seq-attention, $L = 500$	5.04952	12.7
Transformer-ED, $L = 500$	2.46645	34.2
Transformer-D, $L = 4000$	2.22216	33.6
Transformer-DMCA, no MoE-layer, $L = 11000$	2.05159	36.2
Transformer-DMCA, MoE-128, $L = 11000$	1.92871	37.9
Transformer-DMCA, MoE-256, $L = 7500$	1.90325	38.8

Results (cont.)

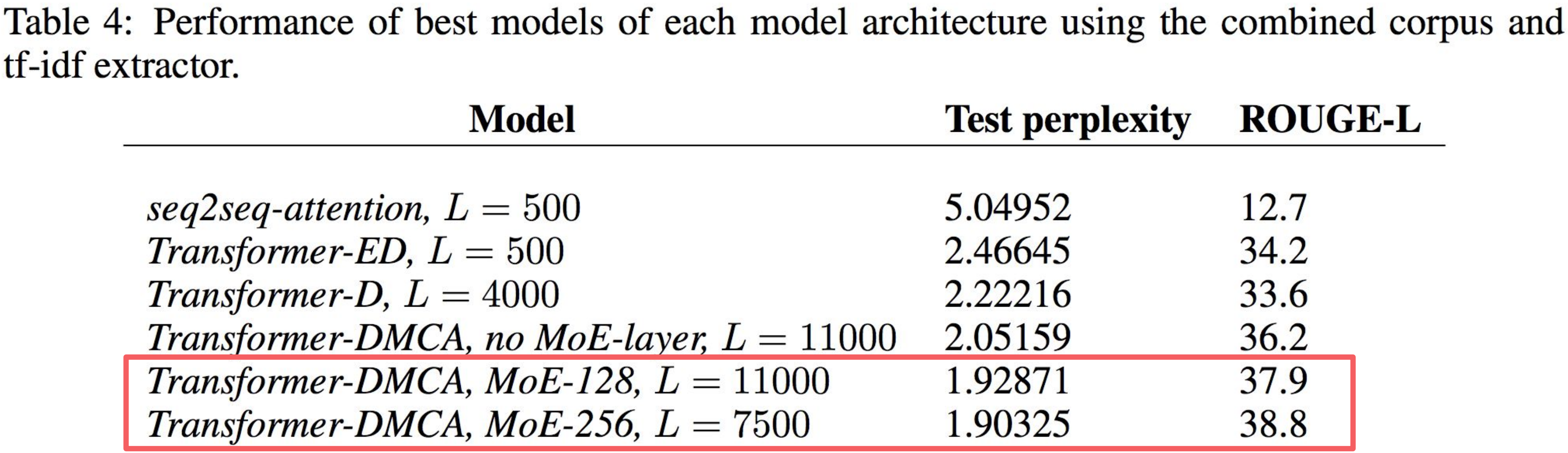
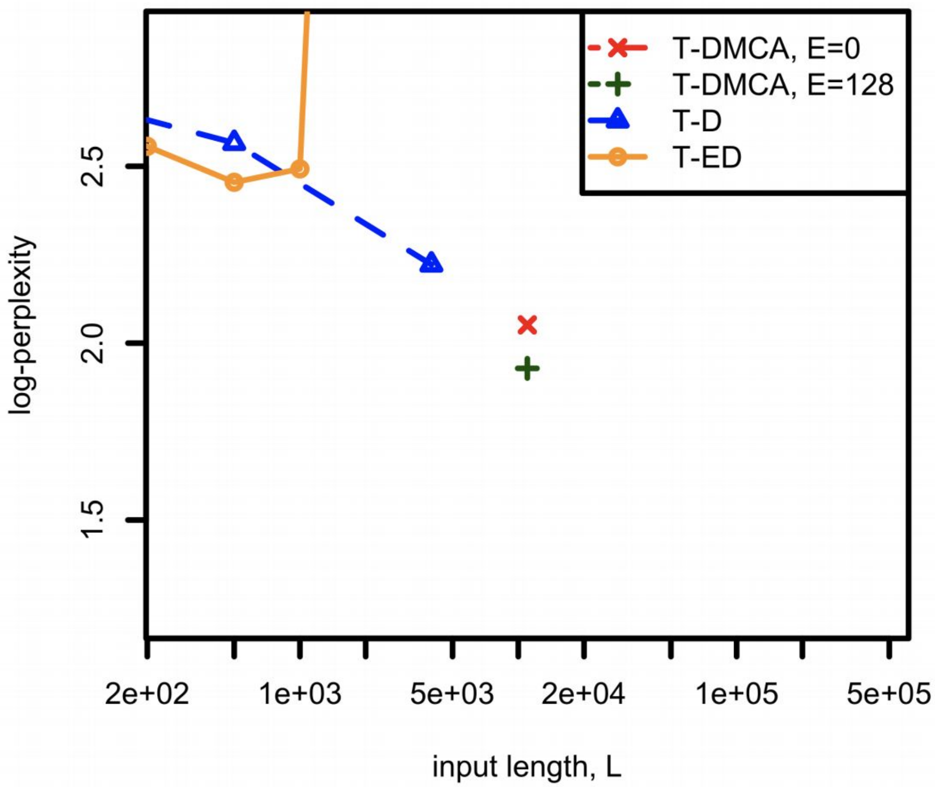


Table 4: Performance of best models of each model architecture using the combined corpus and tf-idf extractor.

Model	**Test perplexity**	**ROUGE-L**
seq2seq-attention, $L = 500$	5.04952	12.7
Transformer-ED, $L = 500$	2.46645	34.2
Transformer-D, $L = 4000$	2.22216	33.6
Transformer-DMCA, no MoE-layer, $L = 11000$	2.05159	36.2
Transformer-DMCA, MoE-128, $L = 11000$	1.92871	37.9
Transformer-DMCA, MoE-256, $L = 7500$	1.90325	38.8

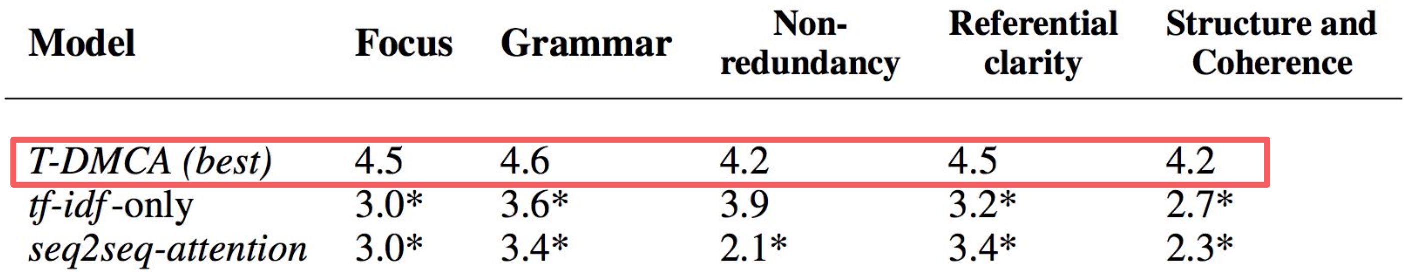
Results (cont.)



- Perplexity vs. L (tf-idf extraction, combined corpus)
    - E = size of mixture-of-experts layer
        - Added model capacity for high L

Results (cont.): Human Evaluations

- Evaluation of linguistic quality on 5 dimensions:
    - Raters assign randomly selected samples a score from 1 to 5 (higher is better)

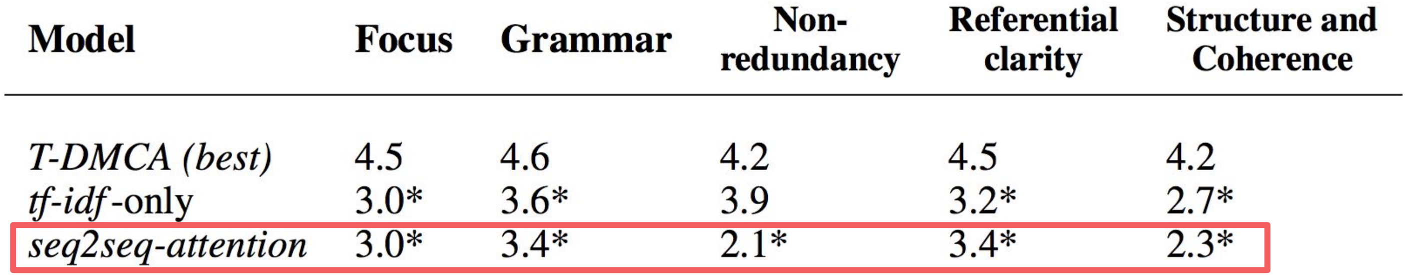


Model	**Focus**	**Grammar**	**Non-redundancy**	**Referential clarity**	**Structure and Coherence**
T-DMCA (best)	4.5	4.6	4.2	4.5	4.2
tf-idf-only	3.0*	3.6*	3.9	3.2*	2.7*
seq2seq-attention	3.0*	3.4*	2.1*	3.4*	2.3*

- Side-by-side preference experiments: human judgement correlates with automatic metrics

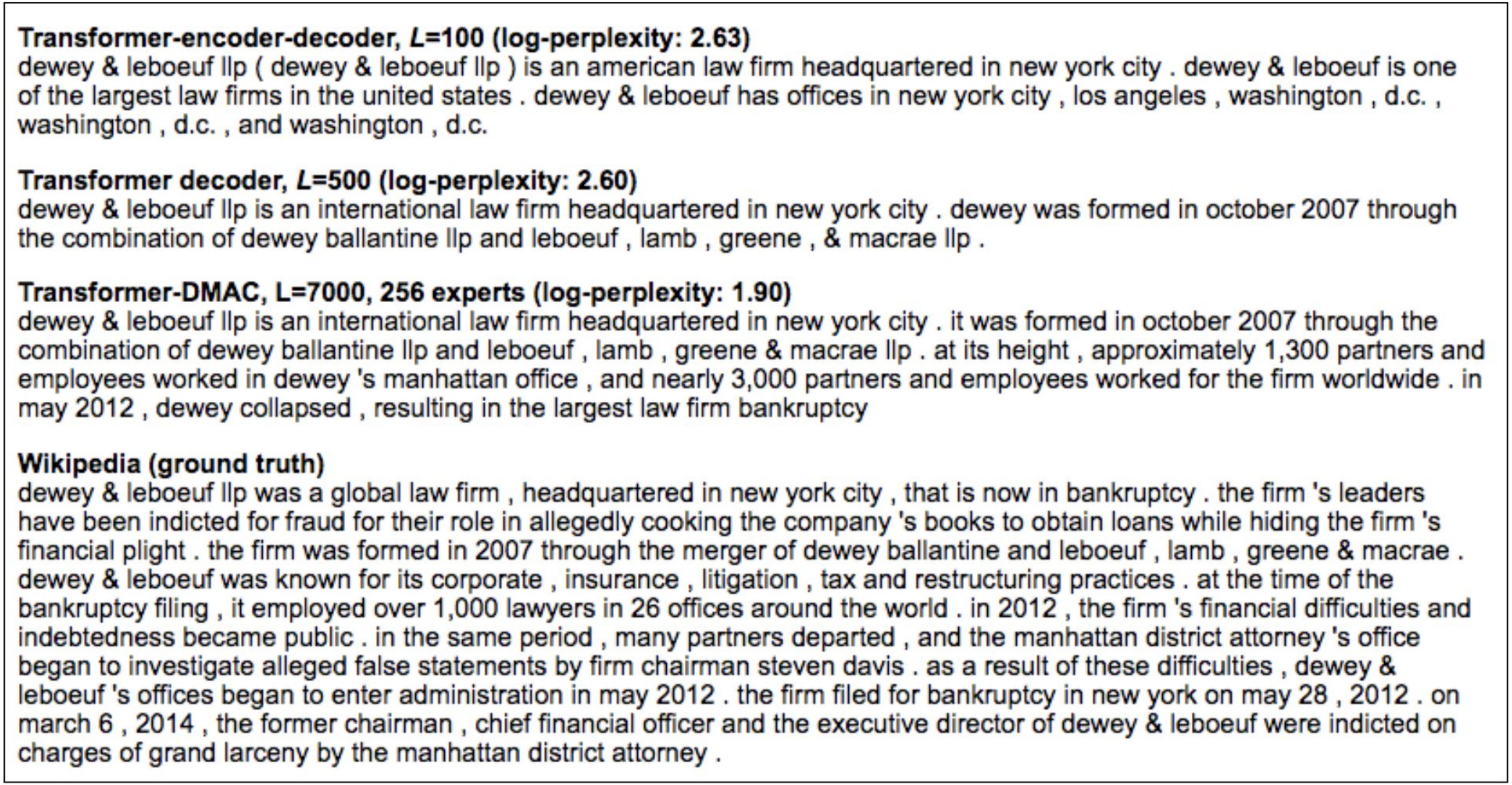
Results (cont.): Human Evaluations

- Evaluation of linguistic quality on 5 dimensions:
  - Raters assign randomly selected samples a score from 1 to 5 (higher is better)



Model	**Focus**	**Grammar**	**Non-redundancy**	**Referential clarity**	**Structure and Coherence**
T-DMCA (best)	4.5	4.6	4.2	4.5	4.2
tf-idf-only	3.0*	3.6*	3.9	3.2*	2.7*
seq2seq-attention	3.0*	3.4*	2.1*	3.4*	2.3*

  - Side-by-side preference experiments: human judgement correlates with automatic metrics



Transformer-encoder-decoder, *L*=100 (log-perplexity: 2.63)
dewey & leboeuf llp (dewey & leboeuf llp) is an american law firm headquartered in new york city . dewey & leboeuf is one of the largest law firms in the united states . dewey & leboeuf has offices in new york city , los angeles , washington , d.c. , washington , d.c. , and washington , d.c.

Transformer decoder, *L*=500 (log-perplexity: 2.60)
dewey & leboeuf llp is an international law firm headquartered in new york city . dewey was formed in october 2007 through the combination of dewey ballantine llp and leboeuf , lamb , greene , & macrae llp .

Transformer-DMAC, L=7000, 256 experts (log-perplexity: 1.90)
dewey & leboeuf llp is an international law firm headquartered in new york city . it was formed in october 2007 through the combination of dewey ballantine llp and leboeuf , lamb , greene & macrae llp . at its height , approximately 1,300 partners and employees worked in dewey 's manhattan office , and nearly 3,000 partners and employees worked for the firm worldwide . in may 2012 , dewey collapsed , resulting in the largest law firm bankruptcy

Wikipedia (ground truth)
dewey & leboeuf llp was a global law firm , headquartered in new york city , that is now in bankruptcy . the firm 's leaders have been indicted for fraud for their role in allegedly cooking the company 's books to obtain loans while hiding the firm 's financial plight . the firm was formed in 2007 through the merger of dewey ballantine and leboeuf , lamb , greene & macrae . dewey & leboeuf was known for its corporate , insurance , litigation , tax and restructuring practices . at the time of the bankruptcy filing , it employed over 1,000 lawyers in 26 offices around the world . in 2012 , the firm 's financial difficulties and indebtedness became public . in the same period , many partners departed , and the manhattan district attorney 's office began to investigate alleged false statements by firm chairman steven davis . as a result of these difficulties , dewey & leboeuf 's offices began to enter administration in may 2012 . the firm filed for bankruptcy in new york on may 28 , 2012 . on march 6 , 2014 , the former chairman , chief financial officer and the executive director of dewey & leboeuf were indicted on charges of grand larceny by the manhattan district attorney .

Figure 4: Shows predictions for the same example from different models.

Takeaways

- WikiSum dataset is orders-of-magnitude larger than previous summarization datasets
- Possible to learn sequence transduction models on combined input-output sequence lengths of ~12000 (T-D)
- Generated articles (of constrained length) are organized into plausible sections, exhibit global coherence
  - Still, not as good as Wikipedia articles or generated leads

Future Work

- Improve extractive methods: train a supervised model to predict relevance
- Extend decoder-only architecture to learn from larger L while maintaining sufficient model capacity
- Focus on full-article task