NEURAL SUMMARIZATION BY EXTRACTING SENTENCES AND WORDS

Jianpeng Cheng, Mirella Lapata (Mar 2016)

presented by Pascal Perle

OVERVIEW

- Related Work
- Problem Formulation
- Training Data for Summarization
- Reminder: Convolution Layer + LSTM
- Neural Summarization Model
  - Document Reader
  - Sentence Extractor
  - Word Extractor
- Implementation Details
- Results
- Conclusions
- Discussion

RELATED WORK

- Most extractive summarization methods relied on human-engineered features.
  - Surface (Radev et al., 2004), content (Nenkova et al., 2006), event (Filatova and Hatzivassiloglou, 2004) features
  - Score assigned to sentence

- Selection of sentences to be the summarization
  - binary classifiers (Kupiec et al., 1995)
  - hidden Markov models (Conroy and O'Leary, 2001)
  - graph-based algorithms (Erkan and Radev, 2004; Mihalcea, 2005)
  - integer linear programming (Woodsend and Lapata, 2010)

RELATED WORK

- neural network architectures for NLP
  - machine translation (Sutskever et al., 2014)
  - question answering (Hermann et al., 2015)
  - sentence compression (Rush et al., 2015)

  - ➔ encoder-decoder architecture

- attention mechanism (Bahdanau et al., 2015)
  - introduced for translation
  - weighted combination of the input

PROBLEM FORMULATION

- Summaries at sentences level (sentence extraction)

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

  - scoring each sentence within D (the source document)
  - predicting a label $y_L \in \{0,1\}$
  - θ are the model parameters
  - m is the number of sentences in D

    + little linguistic analysis for naturally grammatical summaries required
    - long summaries containing much redundant information

PROBLEM FORMULATION

- Summaries at word level (word extraction)

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

  - $y_s = (w'_1, \cdots, w'_k), w'_i \in D$
  - subset of words (can also include a small set of commonly-used words) in D and their optimal ordering

TRAINING DATA

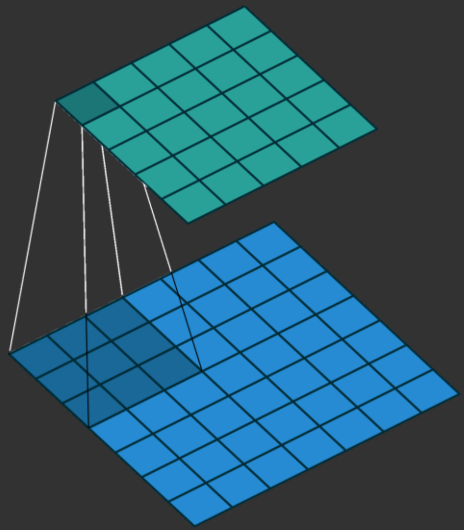
- Problem: large training corpus with labels needed
  - DUC 2002 (567 documents) only for testing

- DailyMail dataset (Hermann et al., 2015)
  - sentence extraction (200K articles)
    - rulebased algorithms to match highlights to document content
      - position of the sentence in the document
      - the unigram and bigram overlap between sentences and highlights
      - the number of entities appearing in the highlight and in the sentence
  - word extraction (170K articles)
    - check if all highlight words (after stemming) come from the original document
    - out-of-vocabulary words, check for semantically equivalent replacement in the article (word2vec-GoogleNews-300dim-vectors)

REMINDER: CONVOLUTION LAYER

- used in image recognition/classifications
- maintains the relationship between pixels by using small squares of input data to learn features

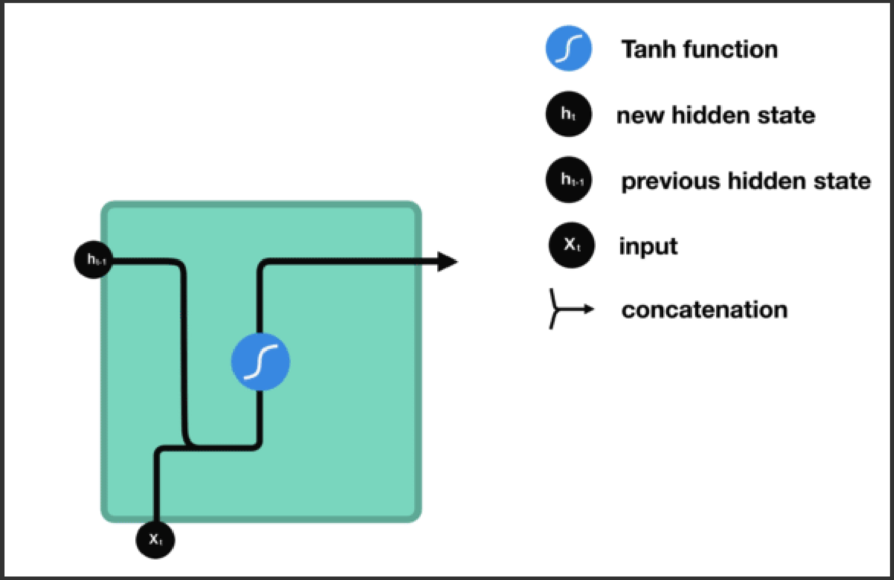
- Kernel:
  - learnable filter
- Stride:
  - number of pixels over the input matrix



(Dumoulin et al., 2016)

REMINDER: LSTM

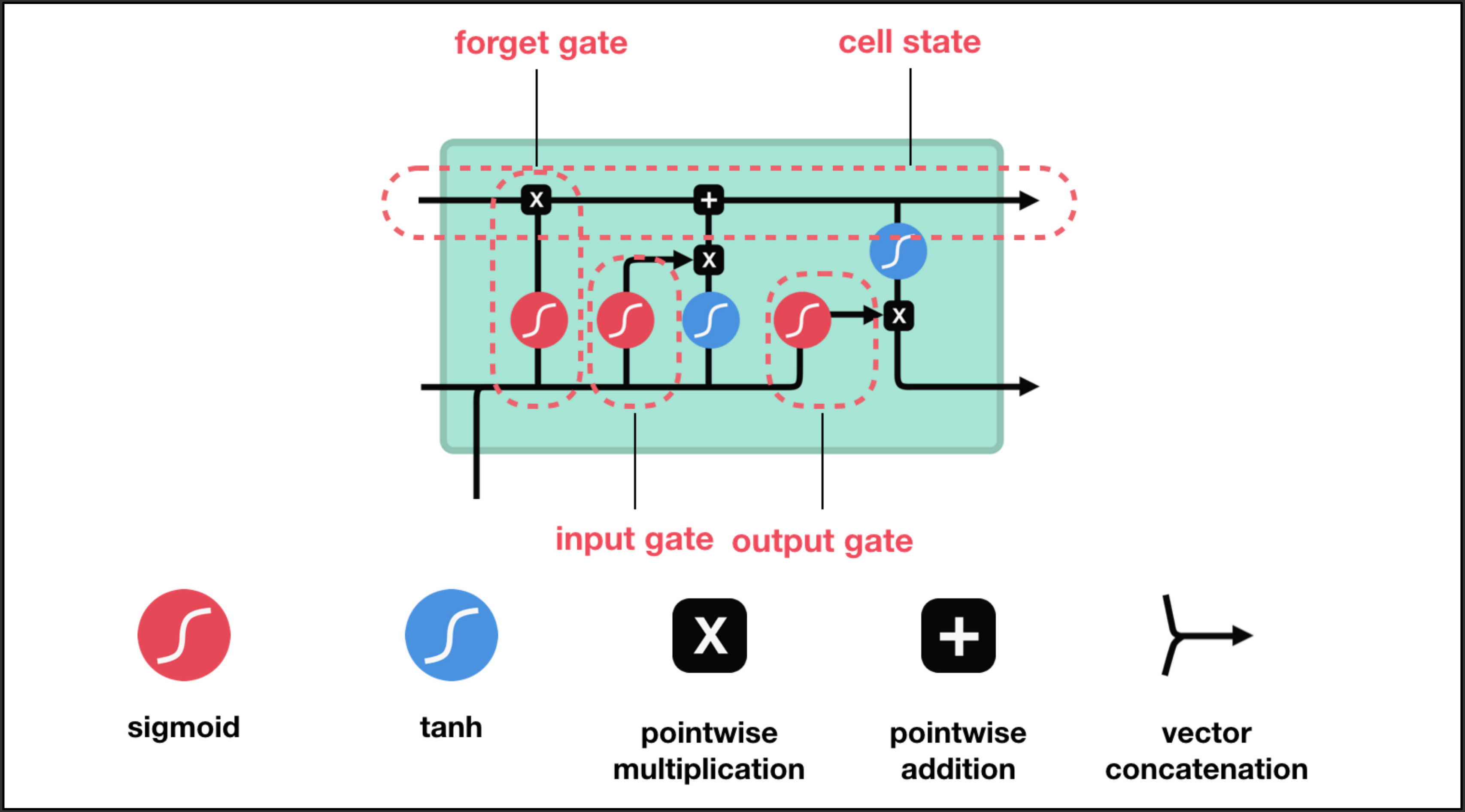
- vanilla RNN
  - vanishing/exploding gradient problem



(Nguyen, 2018)

- Long short-term memory (LSTM) (Schmidhuber, 1997)
  - minimizes vanishing/exploding gradient problem

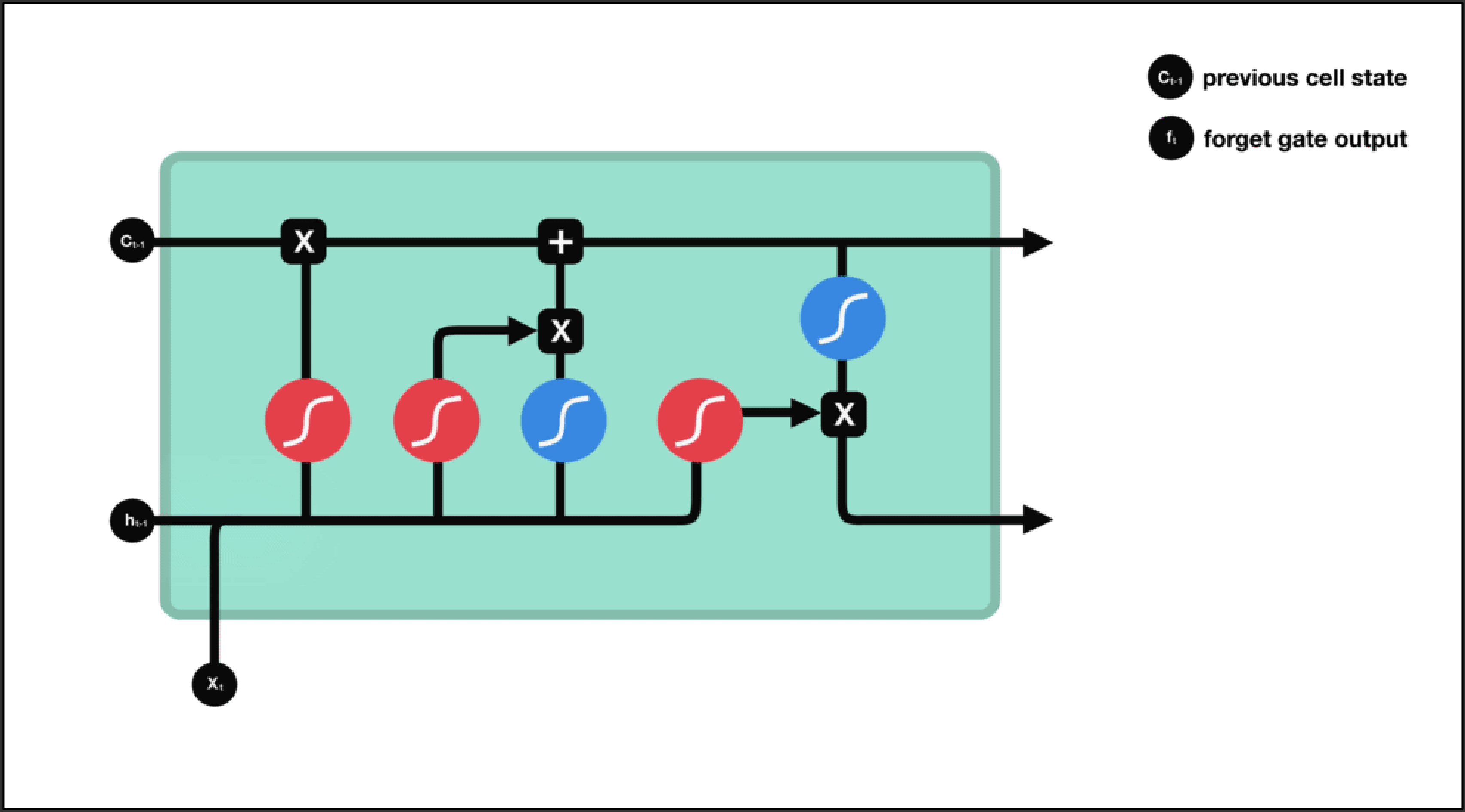
REMINDER: LSTM



(Nguyen, 2018)

REMINDER: LSTM

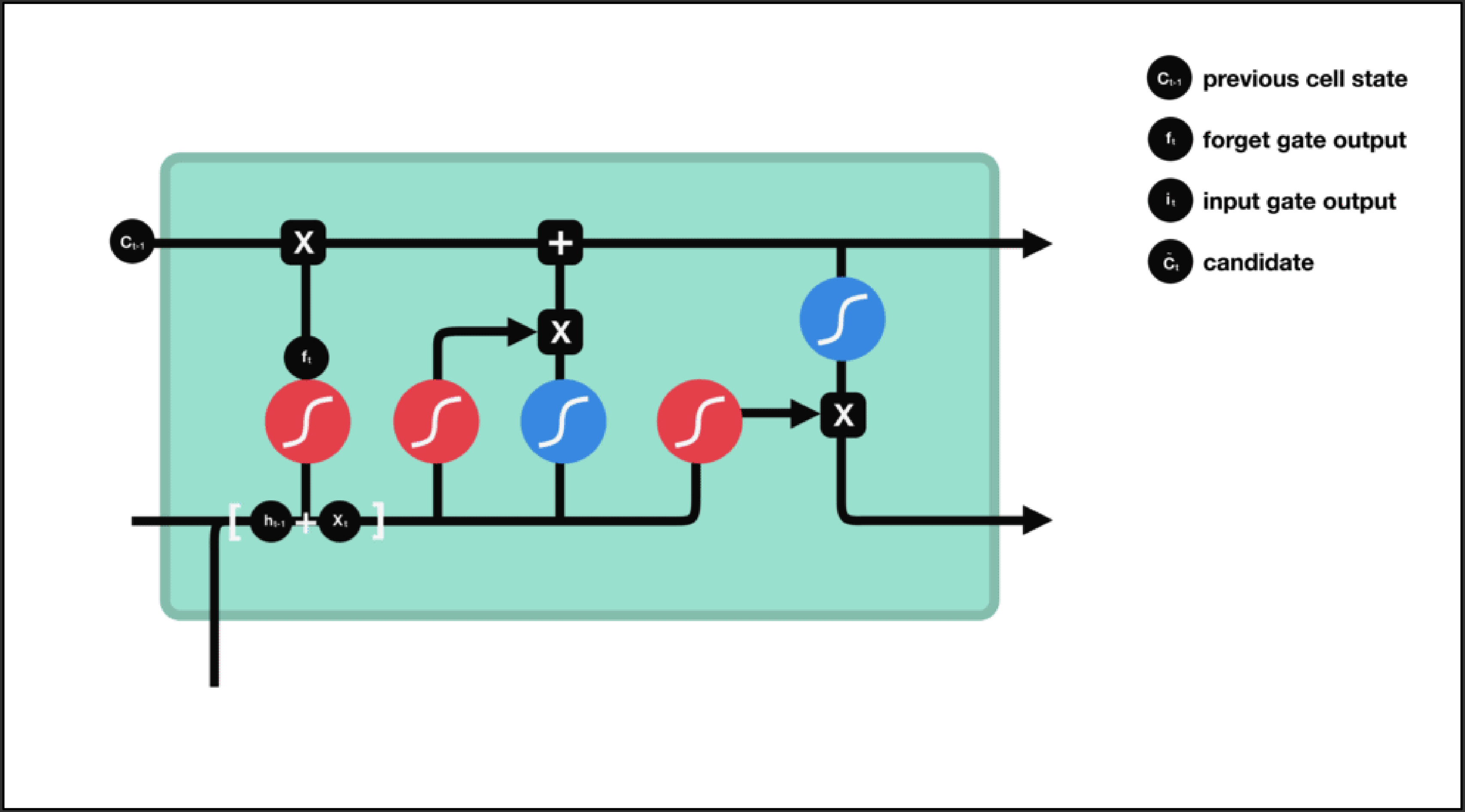
- Forget gate (what is relevant to keep from prior steps)



(Nguyen, 2018)

REMINDER: LSTM

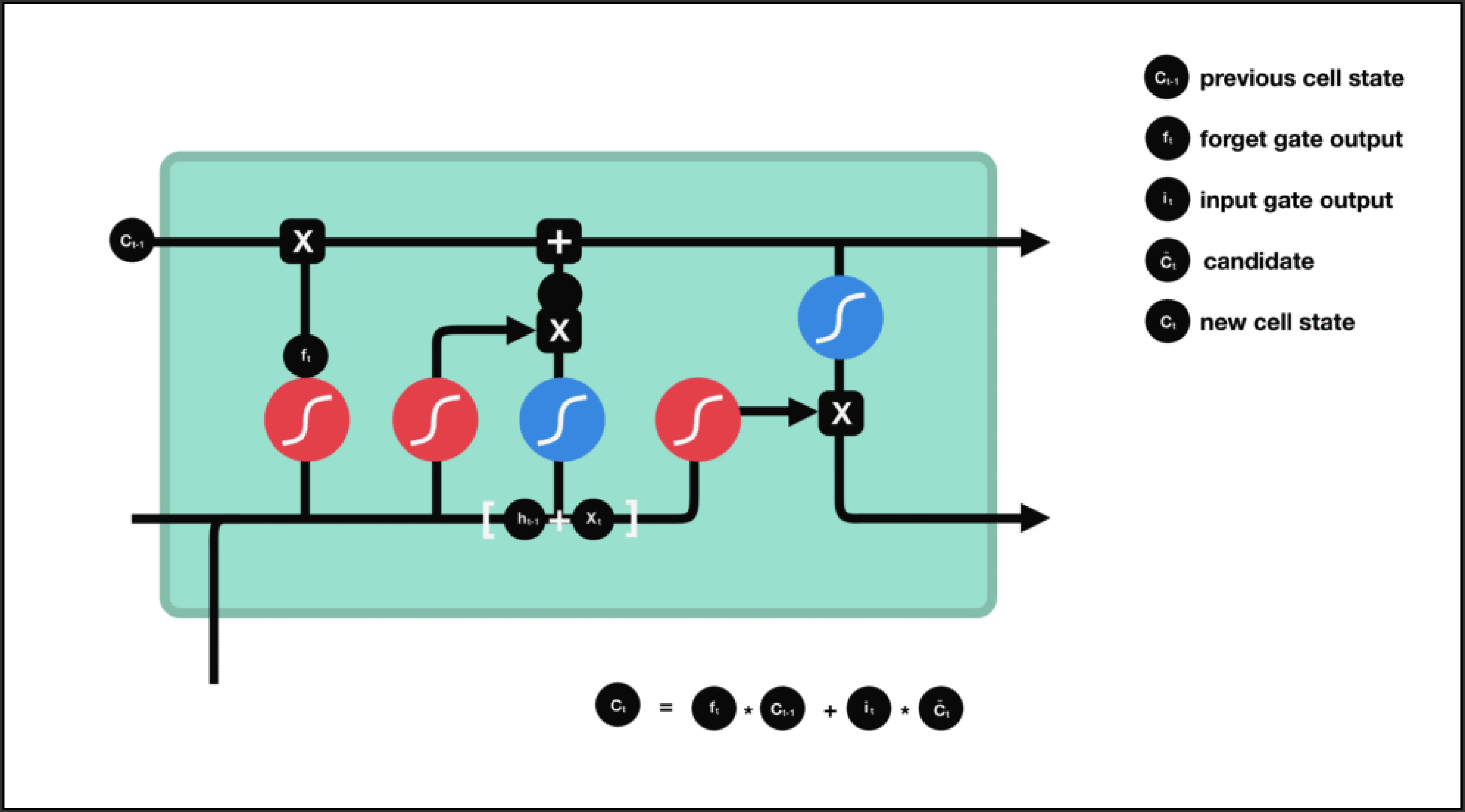
- Input Gate (what is relevant to add from the current step)



(Nguyen, 2018)

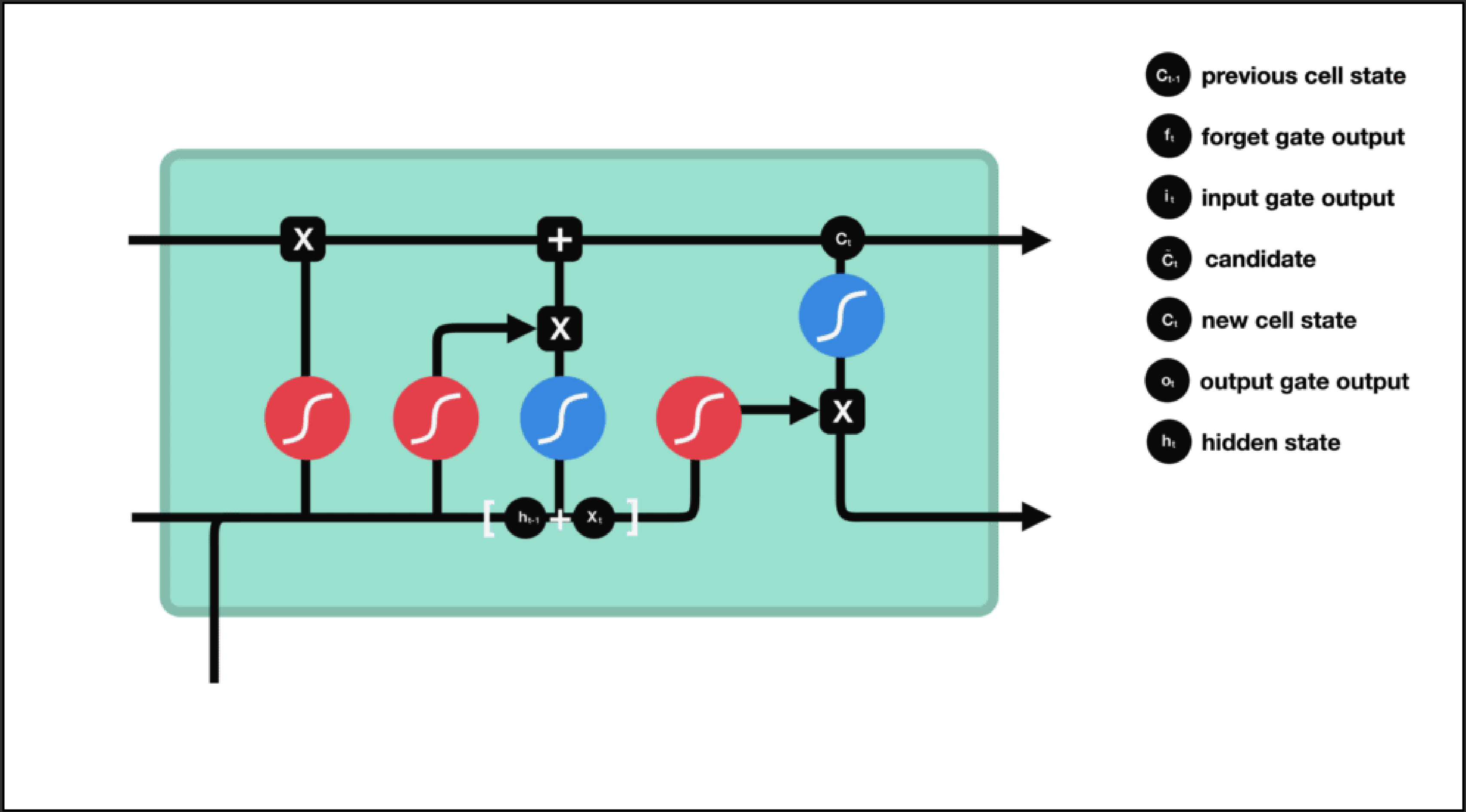
REMINDER: LSTM

- Cell State (transfers relative information, “memory”)



REMINDER: LSTM

- Output Gate (determines what the next hidden state should be)

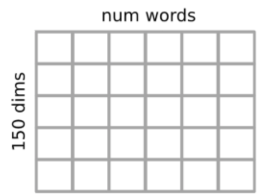


(Nguyen, 2018)

NEURAL SUMMARIZATION MODEL

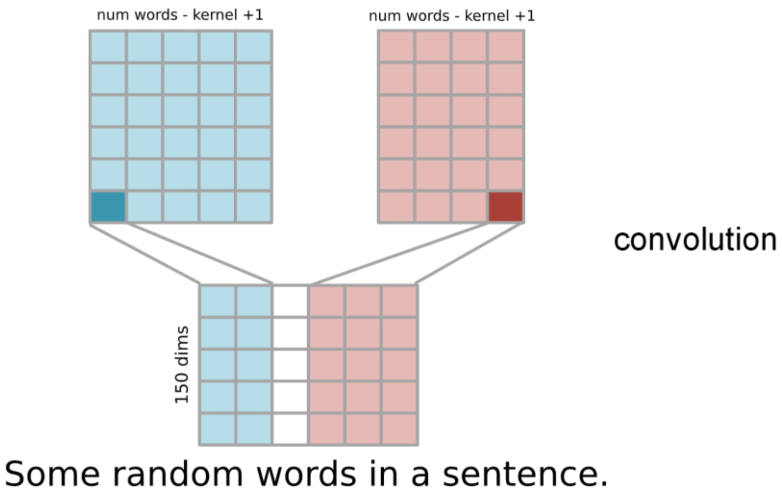
- Document Reader
- Convolutional Sentence Encoder
  - multiple kernels with different widths $\{1, 2, 3, 4, 5, 6, 7\}$
  - max-over-time pooling operation
  - summed to get the final sentence representation

CONV. SENTENCE ENCODER

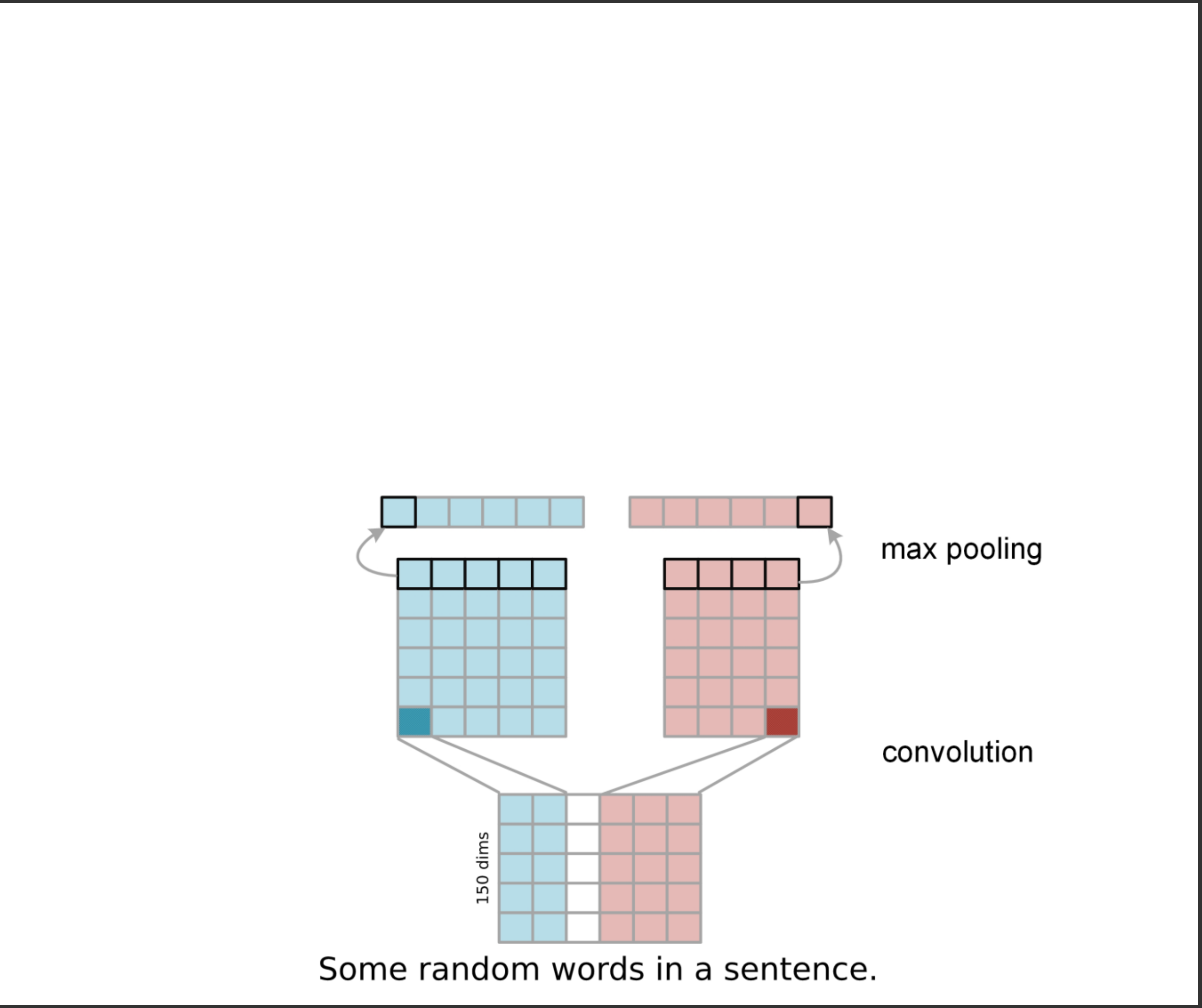


Some random words in a sentence.

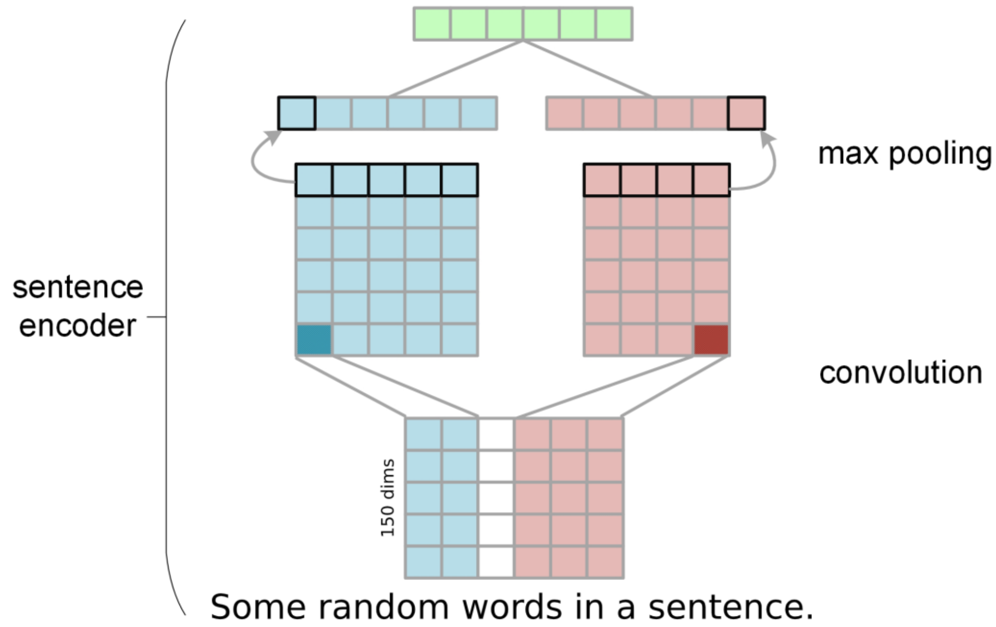
CONV. SENTENCE ENCODER



CONV. SENTENCE ENCODER



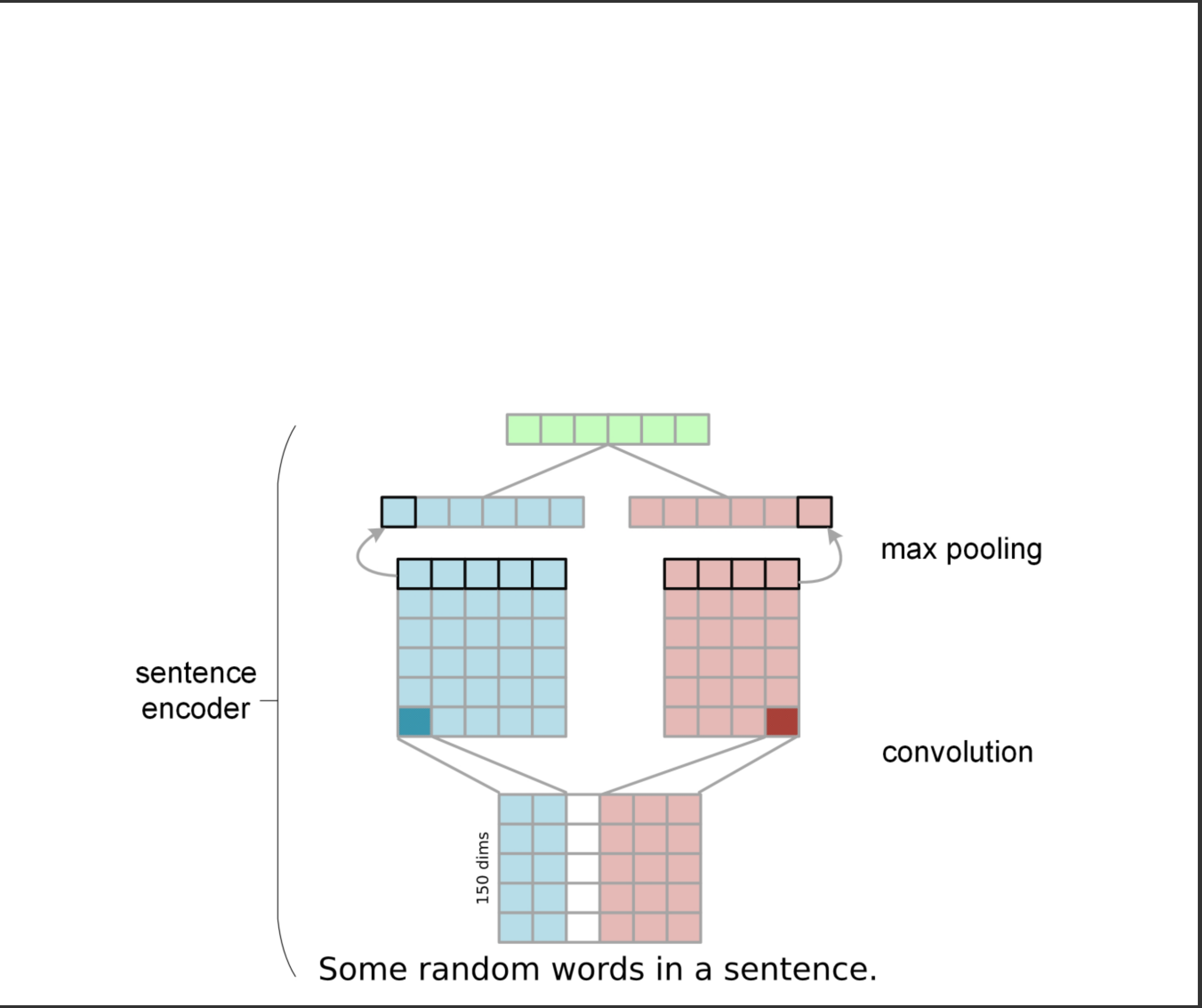
CONV. SENTENCE ENCODER



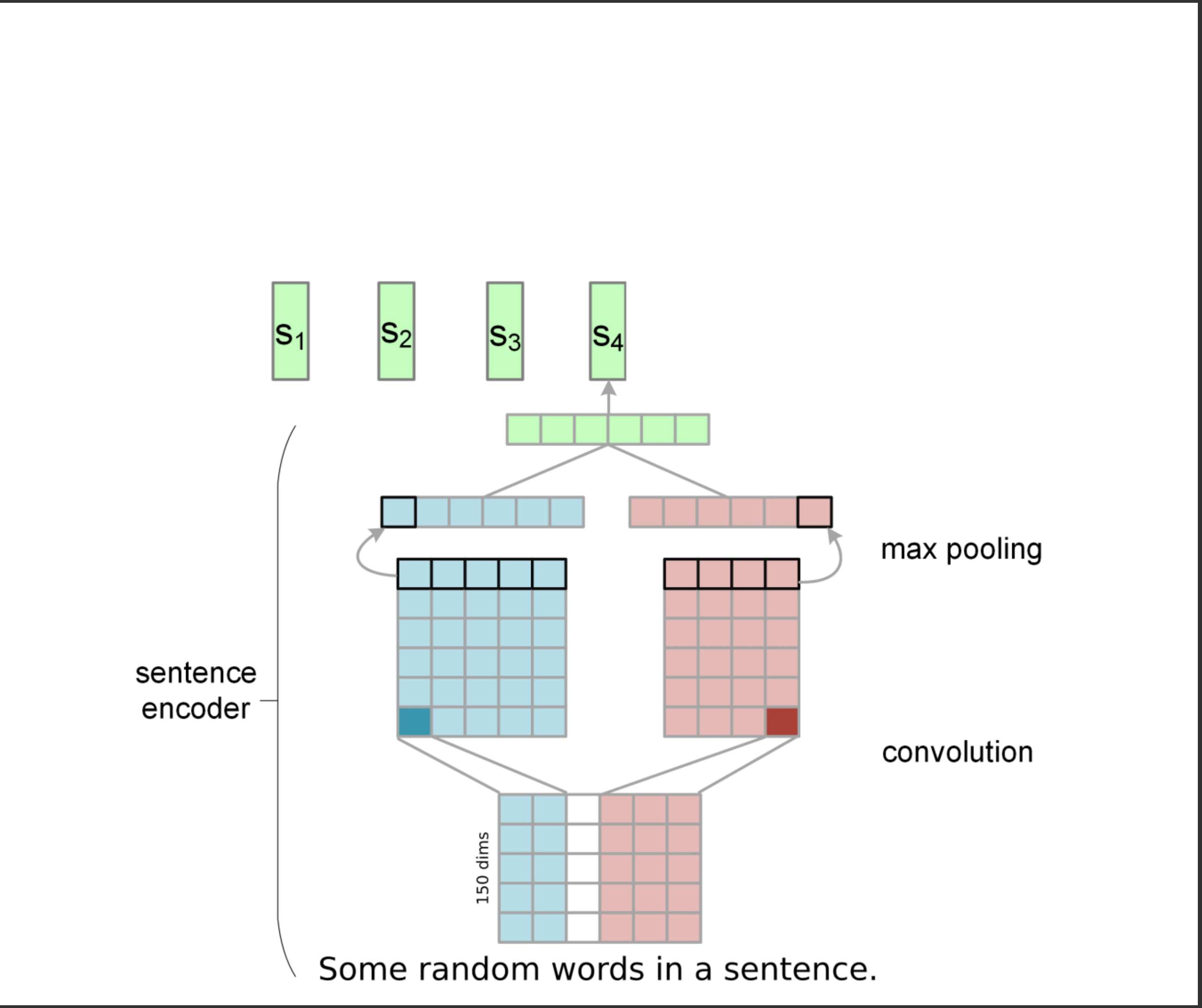
NEURAL SUMMARIZATION MODEL

- Document Reader
- Recurrent Document Encoder
  - representations for documents using LSTMs
    - ameliorating the vanishing gradient problem when training long sequences
  - sequence of sentence vectors into a document vector

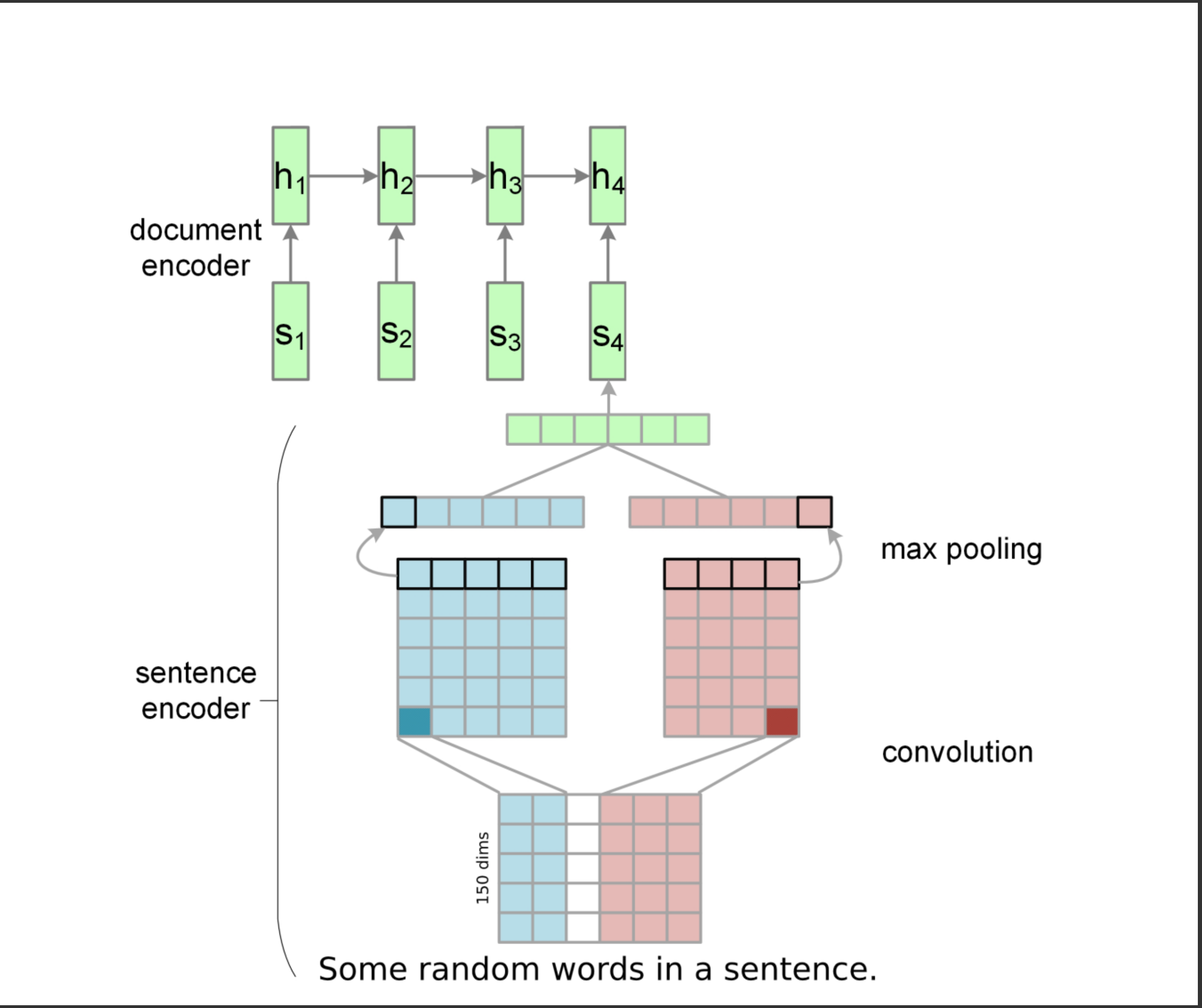
RECURRENT DOCUMENT ENCODER



RECURRENT DOCUMENT ENCODER



RECURRENT DOCUMENT ENCODER



SENTENCE EXTRACTOR

- recurrent neural network labels sentences sequentially
  - “attention” applied directly to extract sentences
  - labeling decision is made with both
    - encoded document at timestep t
    - previously labeled sentences $t - 1$

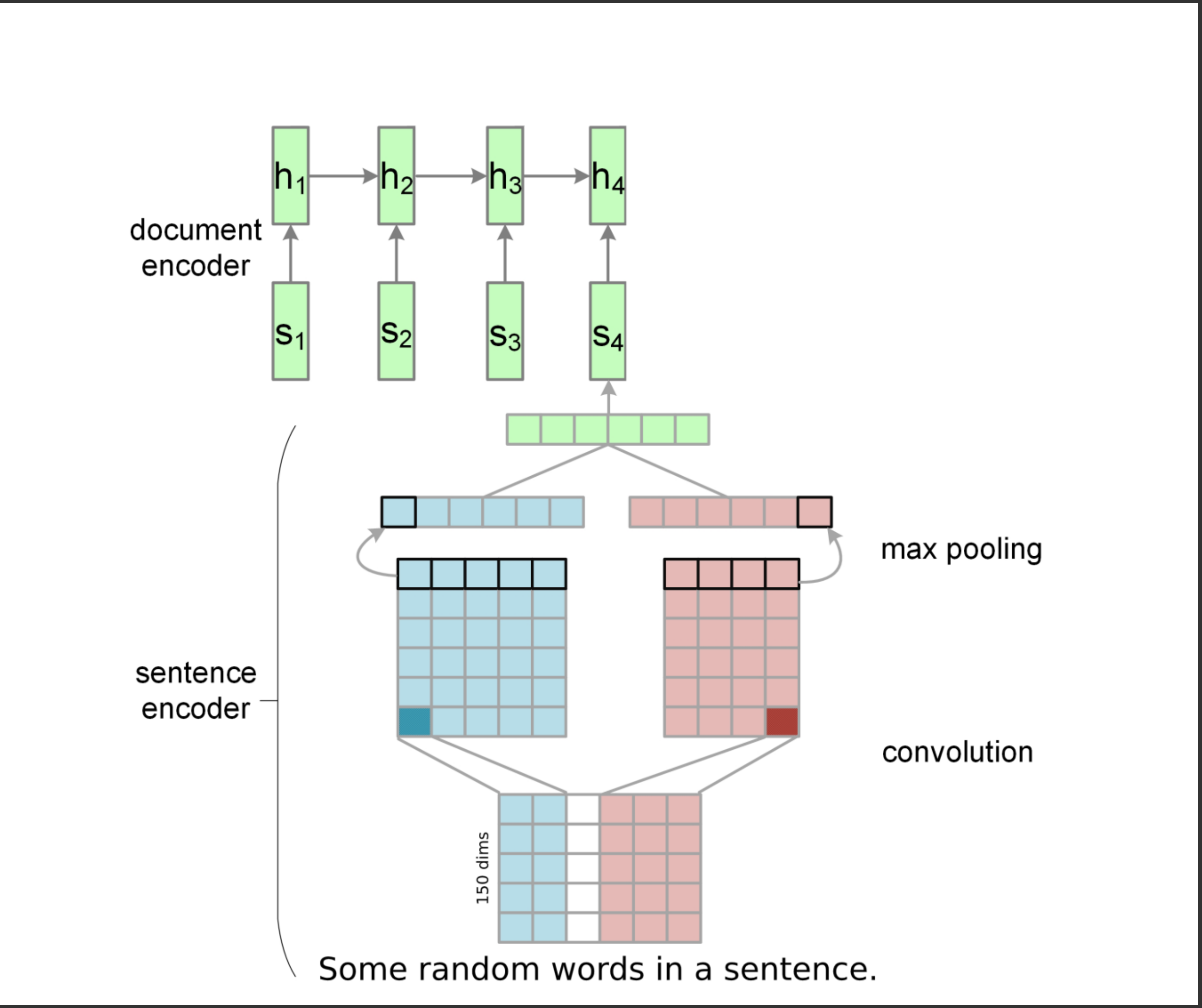
SENTENCE EXTRACTOR

$$\bar{\mathbf{h}}_t = \mathrm{LSTM}(p_{t-1}\mathbf{s}_{t-1}, \bar{\mathbf{h}}_{t-1})$$
$$p(y_L(t) = 1|D) = \sigma(\mathrm{MLP}(\bar{\mathbf{h}}_t : \mathbf{h}_t))$$

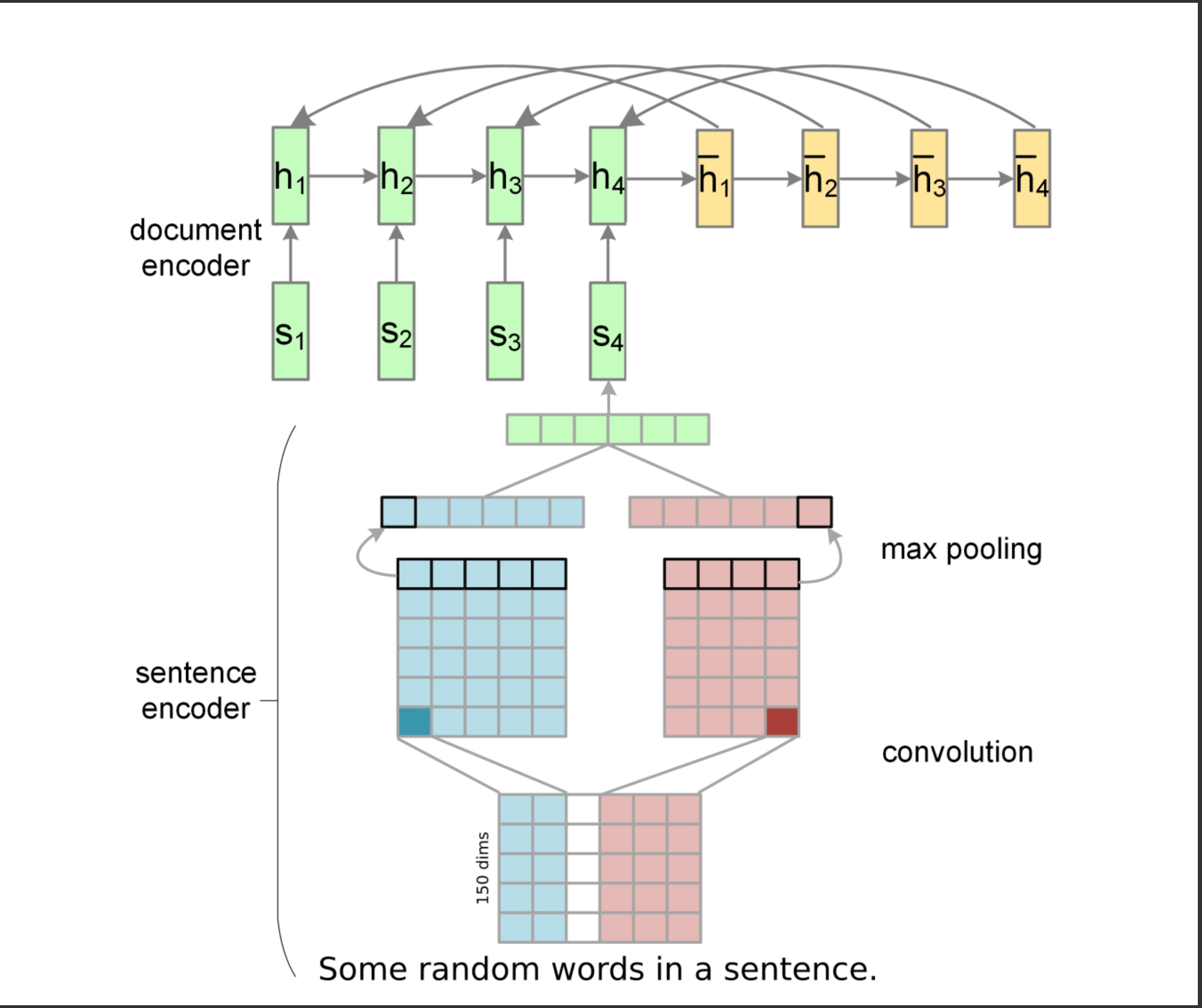
- $\bar{h}$ extractor hidden state
- p_t degree to which the last cell believes the previous sentence should be extracted and memorized
  - curriculum learning strategy

- h encoder hidden state
- MLP multi layer neural network

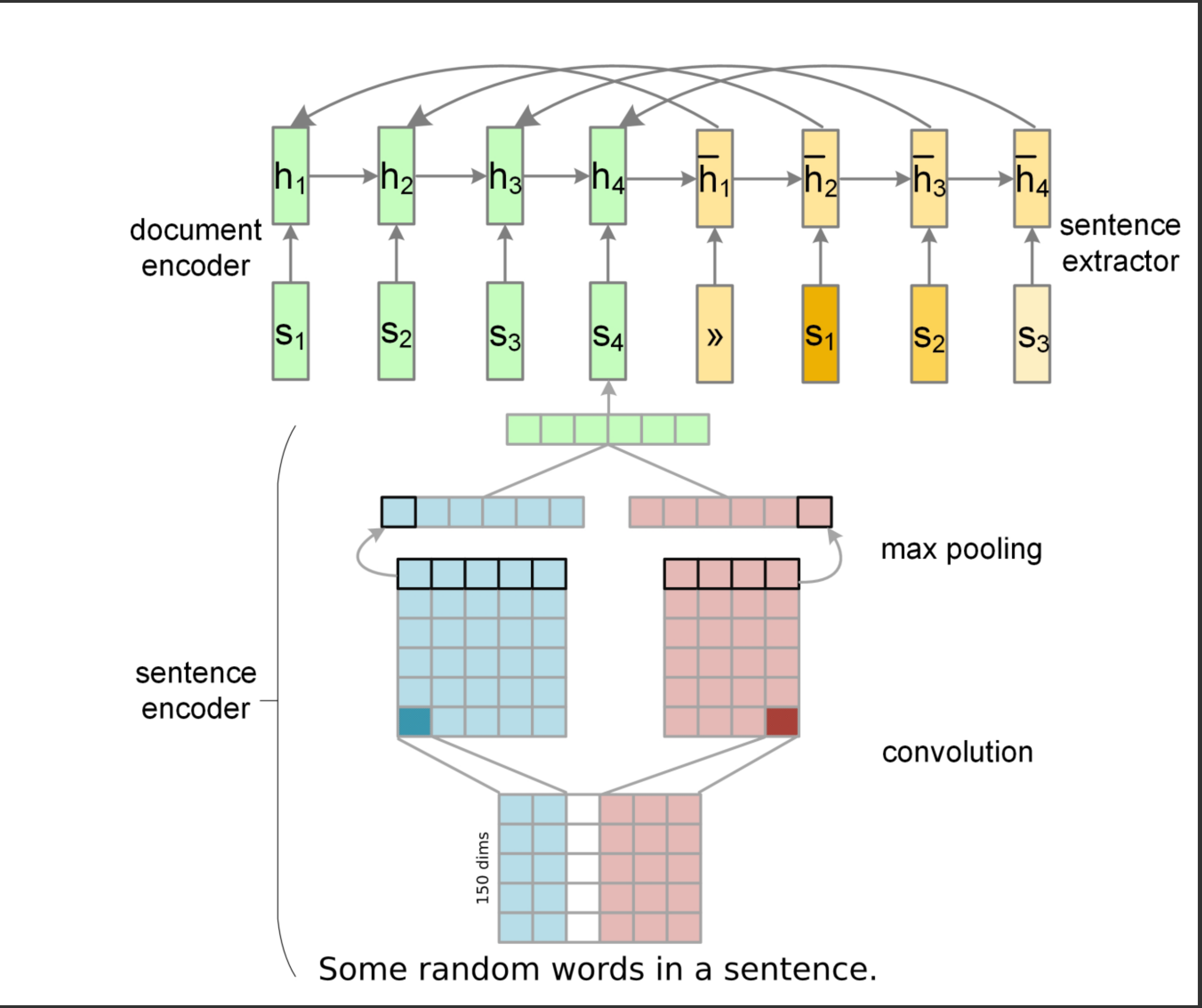
SENTENCE EXTRACTOR



SENTENCE EXTRACTOR



SENTENCE EXTRACTOR



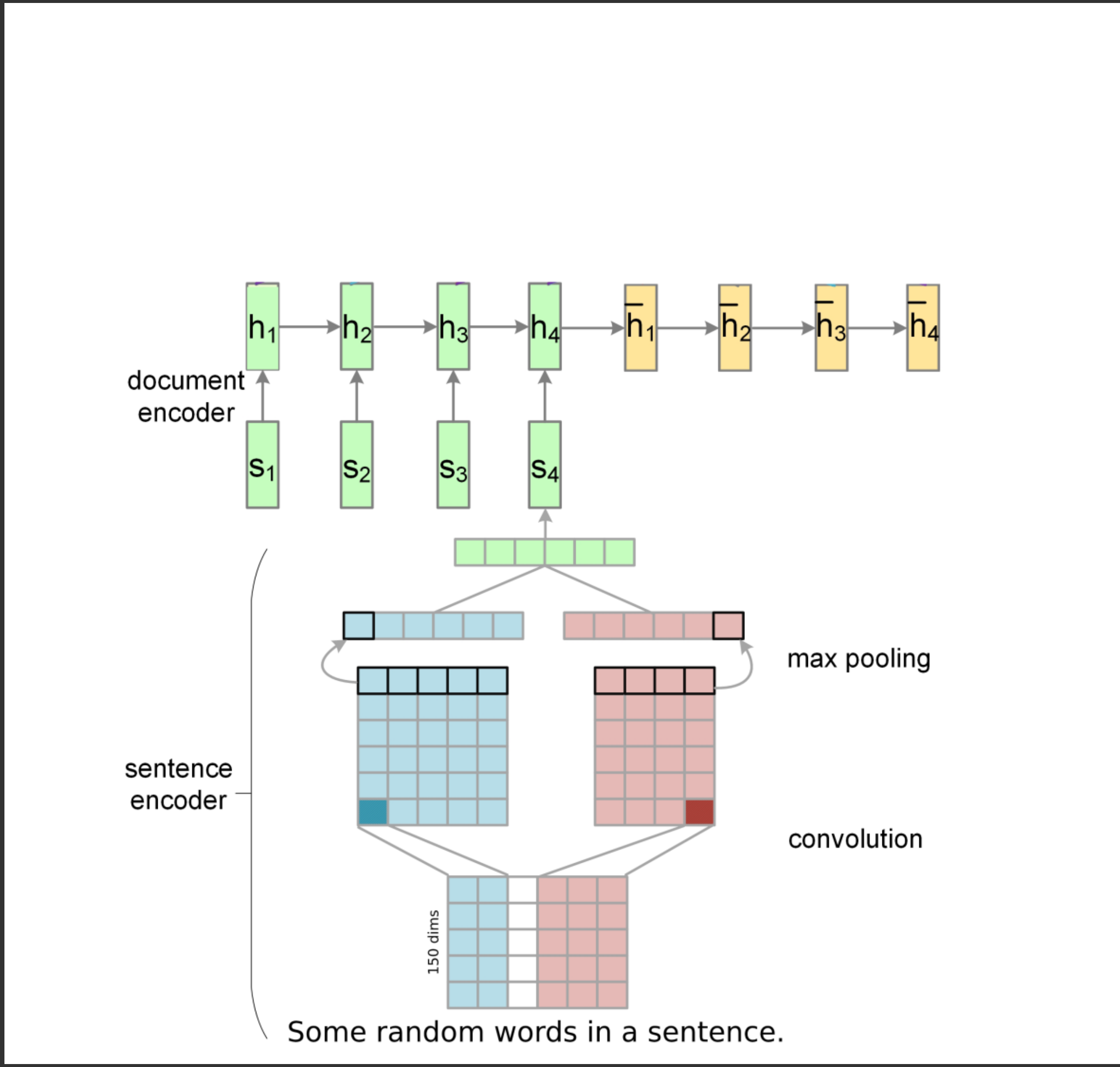
WORD EXTRACTOR

- Can be seen as a generation task
  - words must be selected
  - sentences rendered fluently and grammatically correct

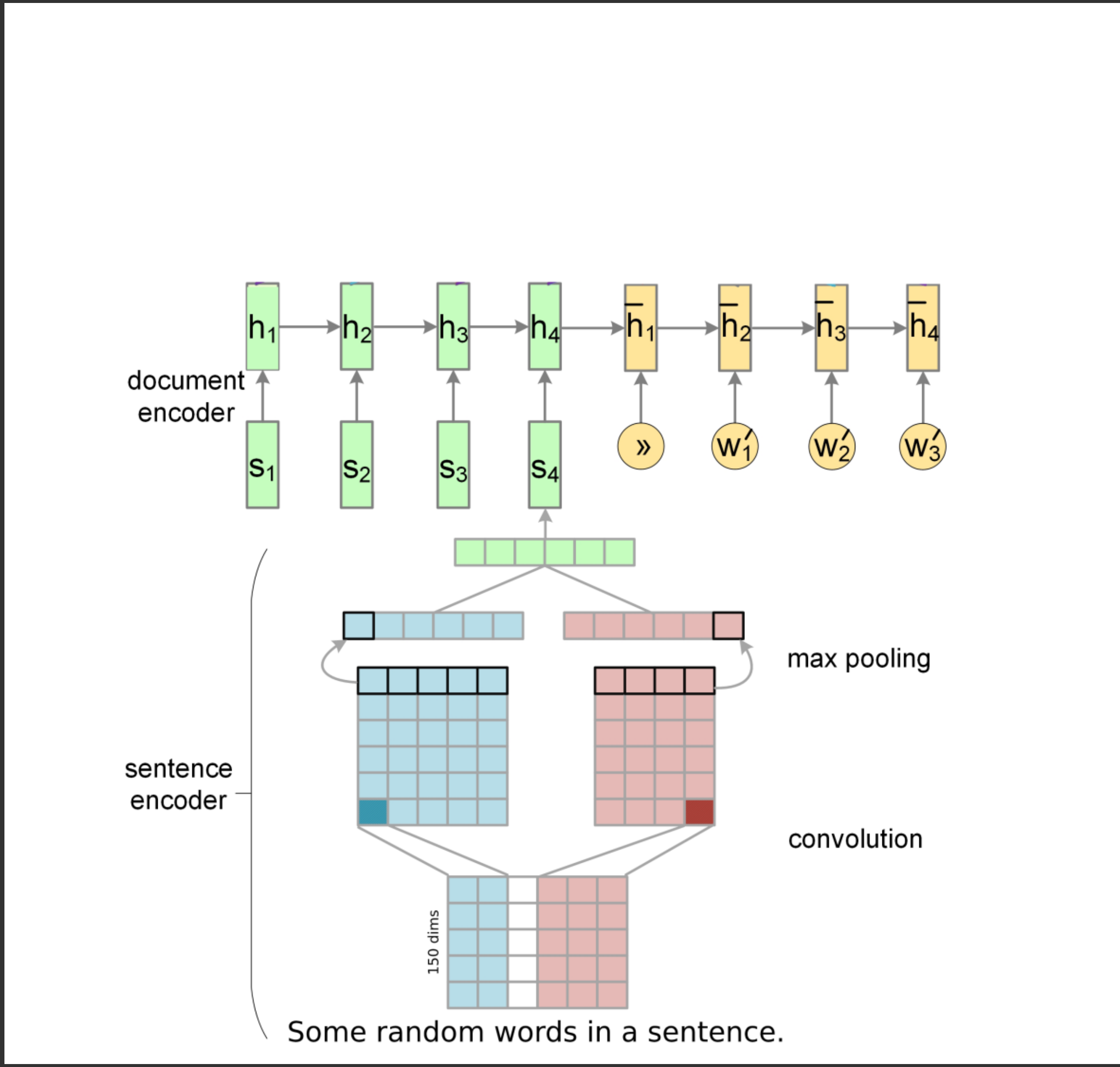
- hierarchical attention architecture:
  - decoder softly attends each document sentence
  - subsequently attends each word in the document and computes the probability of the next word to be included in the summary

- n-gram features collected from the document to rerank candidate summaries obtained via beam decoding
  - log-linear reranker (Och, 2003)

WORD EXTRACTOR

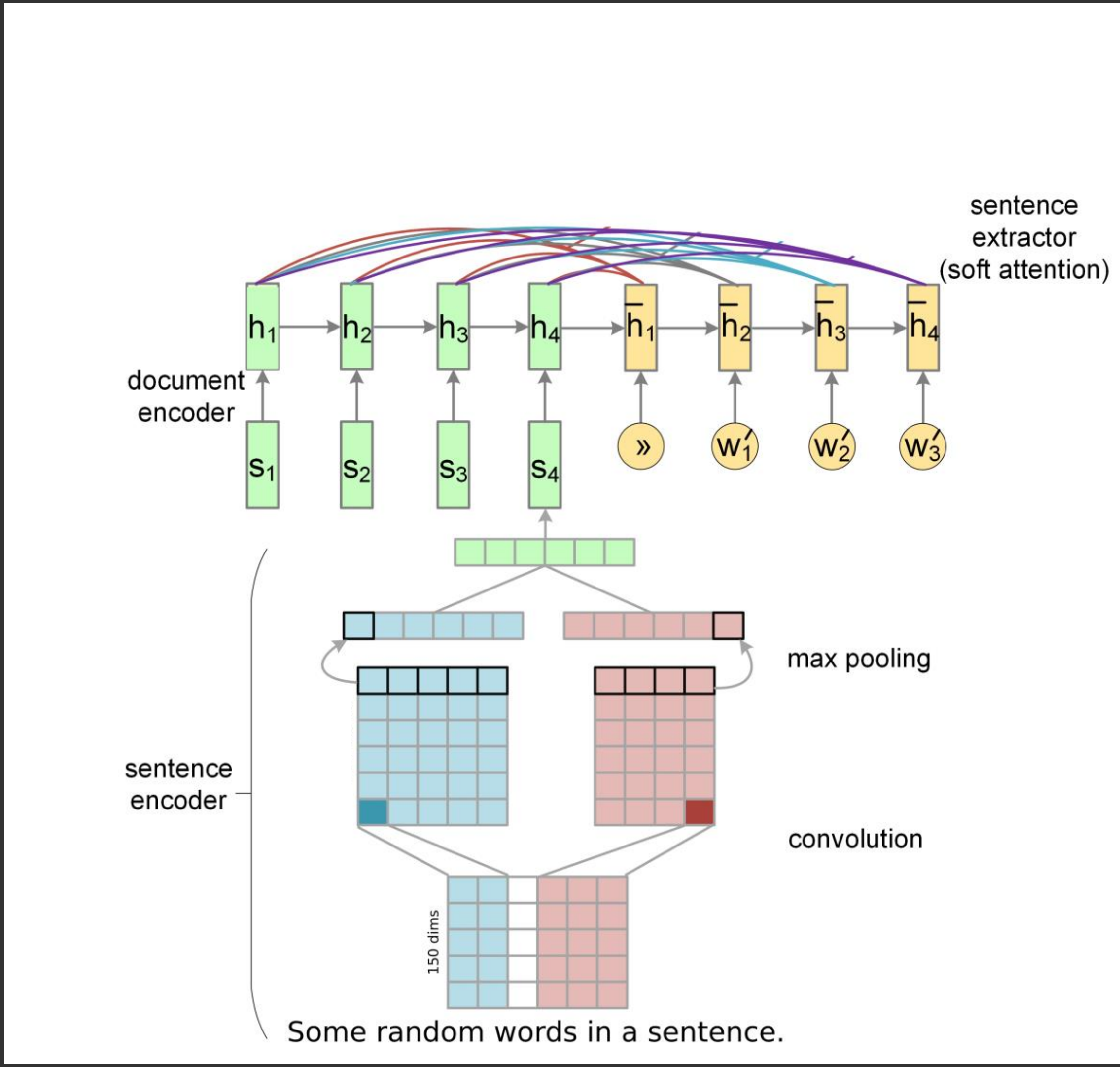


WORD EXTRACTOR



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

WORD EXTRACTOR

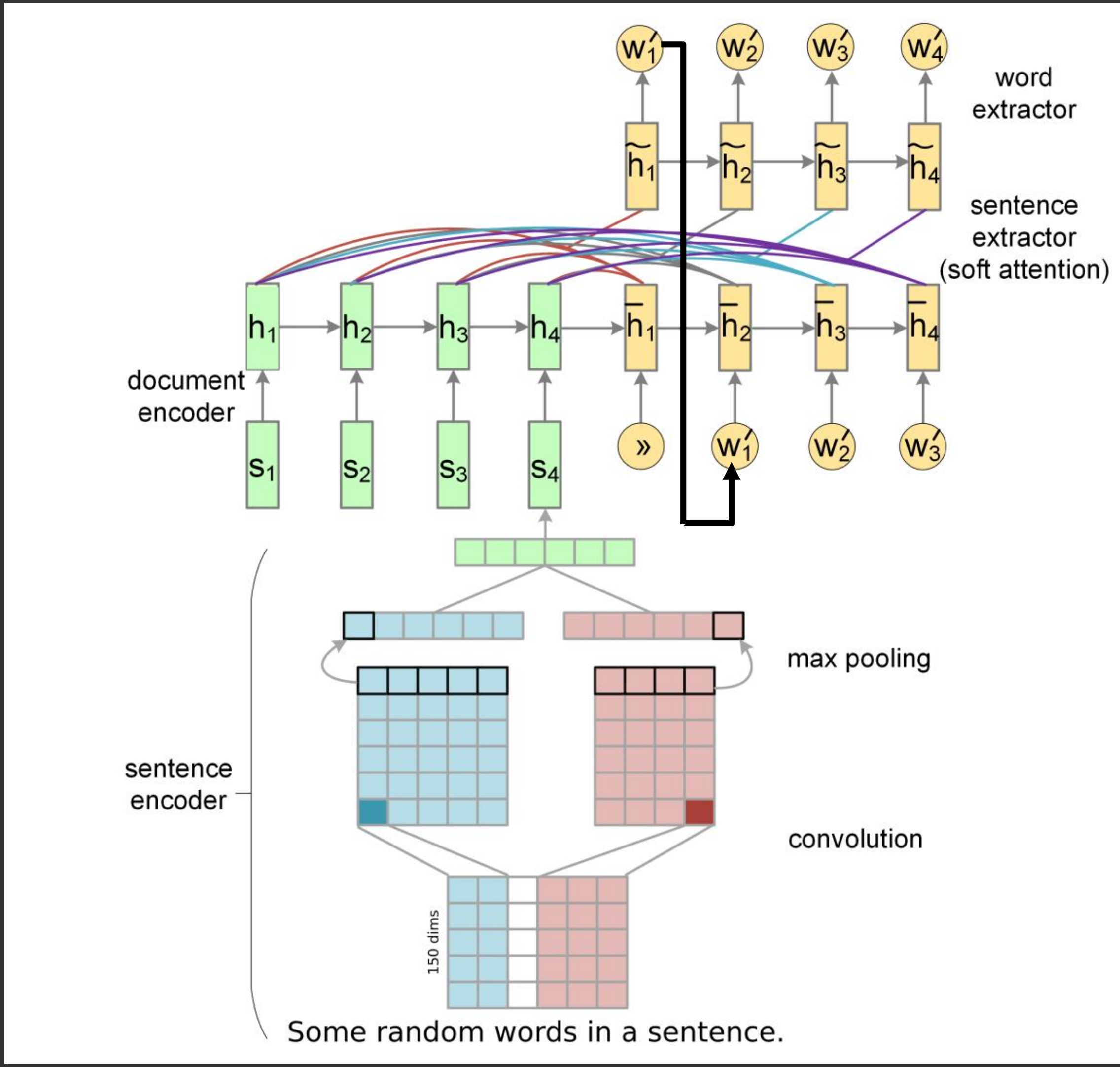


$$\bar{\mathbf{h}}_t = \mathrm{LSTM}(\mathbf{w}'_{t-1}, \bar{\mathbf{h}}_{t-1})$$

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

$$b_j^t = \mathrm{softmax}(a_j^t)$$

WORD EXTRACTOR



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

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

$$b_j^t = \mathrm{softmax}(a_j^t)$$

$$\tilde{\mathbf{h}}_t = \sum_{j=1}^{m} b_j^t \mathbf{h}_j$$

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

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

IMPLEMENTATION DETAILS

- input documents are padded to the same length
- size of word (= 150), sentence (=300), and document (= 750) embeddings
  - word vectors 150 dimensional word2vec pre-trained on Google 1-billion word benchmark (Chelba et al., 2014)
- convolutional kernel sizes {1, 2, 3, 4, 5, 6, 7}
- dropout with probability 0.5 on
  - the LSTM input-to-hidden layers
  - the scoring layer

IMPLEMENTATION DETAILS

- Sentence Extractor
  - number of sentences being selected (three sentences)
    - reranking the positively labeled sentences with the probability scores
    - obtained from the softmax layer (rather than the label itself)

- Word Extractor
  - negative sampling
    - vocabulary of different documents trimmed to the same length

RESULTS

- ROUGE (Lin and Hovy, 2003)
  - ROUGE -1,2 (unigram and bigram overlap)
    - assessing informativeness
  - ROUGE -L (longest common subsequence)
    - assessing fluency

- human judgments for ranking 20 randomly sampled DUC 2002 test documents on formativeness and fluency
  - sentence-based extraction
  - word-based extraction
  - neural abstractive system (Rush et al.,2015)
  - lead baseline (first three sentences)
  - phrase-based ILP model (Woodsend and Lapata, 2010)
  - human authored summary

  - TGRAPH and URANK (DUC 2002 only)

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

RESULTS

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

	1st	2nd	3rd	4th	5th	6th	mean
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-ABS	0.00	0.01	0.05	0.16	0.23	0.54	5.24
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
Human	0.27	0.23	0.29	0.17	0.03	0.01	**2.51**

CONCLUSIONS

- data-driven summarization framework based on an encoder-extractor architecture.
- interesting comparison to human summaries
- Word Extractor has interesting architecture with a reranker rather than just looking at the combined likelihood
  - reranker not described
- for conv layer, rather stack small kernels than using big ones
  - 7x7 <==> 3x(3x3) but less weights + more nonelinearity
- rules for data labeling as well as dataset was not published
- "attention" for sentence extractor only takes one sentence of the document into account

THANK YOU

SOURCES

- Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In Proceedings of the 2015 EMNLP, pages 379–389, Lisbon, Portugal.
- Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In Proceedings of the 29th Annual ACM SIGIR, pages 573–580, Washington, Seattle.
- Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram cooccurrence statistics. In Proceedings of HLT NAACL, pages 71–78, Edmonton, Canada.
- Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. arXiv preprint arXiv:1312.3005.
- Conroy and O'Leary. 2001. Text summarization via hidden Markov models. In Proceedings of the 34th Annual ACL SIGIR, pages 406–407, New Oleans, Louisiana.
- Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. 2004. Mead-a platform for multidocument multilingual text summarization. Technical report, Columbia University Academic Commons.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Proceedings of ICLR 2015, San Diego, California.
- Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In Stan Szpakowicz Marie-Francine Moens, editor, Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, pages 104–111, Barcelona, Spain.

SOURCES

- Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In Proceedings ofthe 41st ACL, pages 160–167, Sapporo, Japan.
- Güneş Erkan and Dragomir R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In Proceedings of the 2004 EMNLP, pages 365–371, Barcelona, Spain.
- Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems 27, pages 3104–3112. Curran Associates, Inc.
- Julian Kupiec, Jan O. Pedersen, and Francine Chen. 1995. A trainable document summarizer. In Proceedings of the 18th Annual International ACM SIGIR, pages 68–73, Seattle, Washington.
- Karl Moritz Hermann, Tomas Kocisky, 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 28, pages 1684–1692. Curran Associates, Inc.

- Karl Moritz Hermann, Tomas Kocisky, 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 28, pages 1684–1692. Curran Associates, Inc.

- Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In Proceedings of the 48th ACL, pages 565–574, Uppsala, Sweden.
- Michael Nguyen. 2018 .Illustrated Guide to LSTM’s and GRU’s: A step by step explanation
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.
- Vincent Dumoulin, Francesco Visin. 2016. A guide to convolution arithmetic for deep learning

DISCUSSION

- Why do Cheng et al. call the concatenation of encoder and decoder state "attention"?

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

- Nallapati et al. argue, Cheng et al. needed higher annotation costs, because they used manually created labels. But did they not instead created training sets via rules and heuristics. Isn't that at least a semi-automatic approach?
  - rules take into account
    - the position of the sentence
    - the unigram and bigram overlap between sentences and highlights
    - the number of entities appearing in the highlight and in the sentence
  - "We adjusted the weights of the rules on 9,000 documents with manual sentence labels created by Woodsend and Lapata (2010)."

DISCUSSION

- It sounds like their word extraction model is abstractive but with the vocabulary fixed to words in the document.
  - "conditional language model with a vocabulary constraint"
    - assigns probabilities to a sequence of words given some context
    - constraint is vocabulary of the document
      - can also be extended to include a small set of commonly-used (high-frequency) words
- Do you think that the combined architecture of CNN+LSTM performs a lot better than considering a single paragraph vector and sentence vector concept based on word2vec/fasttecxt models?
  - Yes, idea introduced in by Kim (2015) "Character-Aware Neural Language Models"
    - better results for languages with rich morphology
  - For Encoding important for long sentences as initial information might get lost otherwise
  - LSTM allows model to know what sentence have been extracted before to not extract a sentence with redundant information

THANK YOU