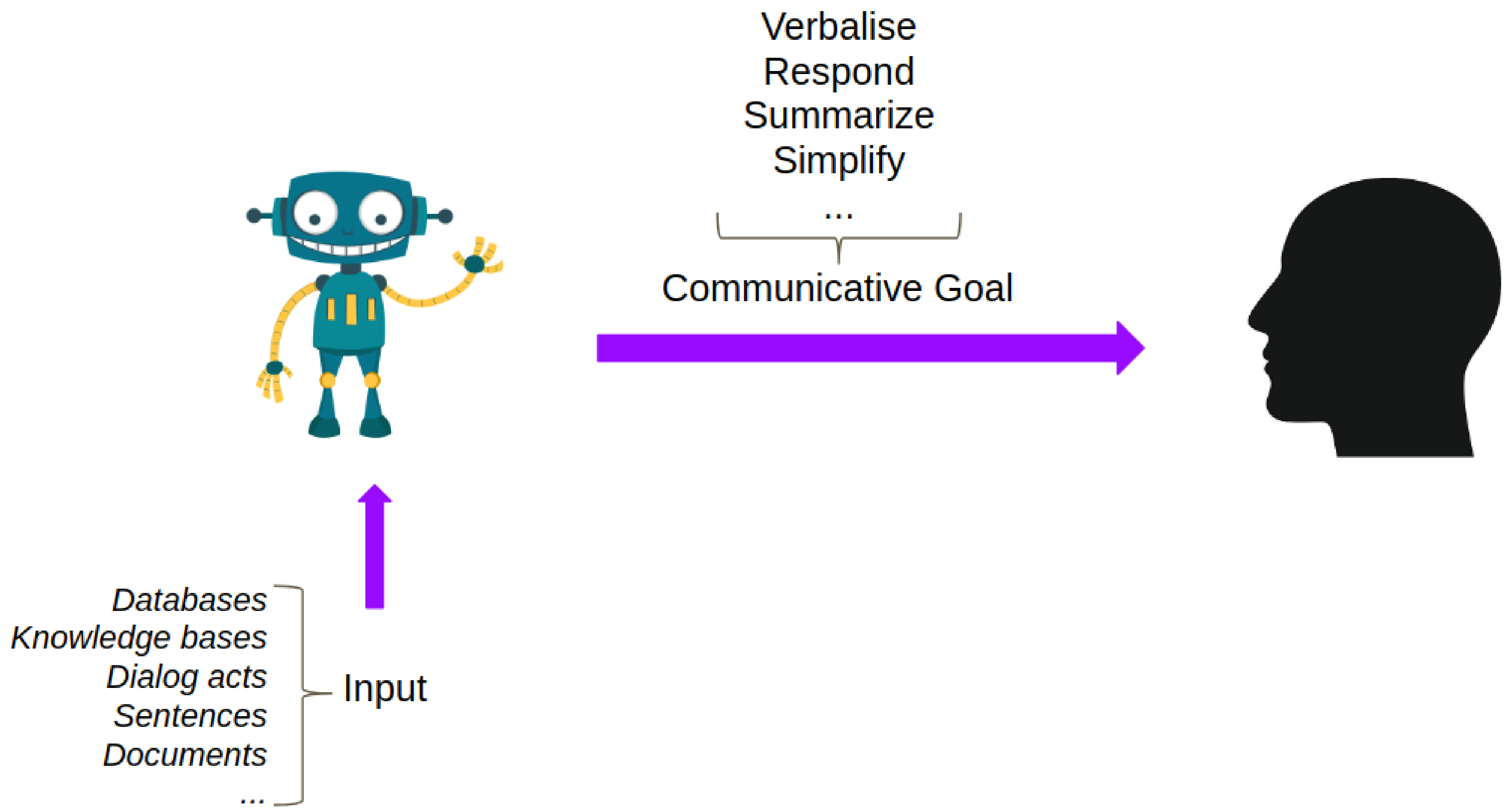
UE903 EC1: Application to Text

Natural Language Generation (NLG)

Neural NLG

Claire Gardent

NLG: Many Inputs, Many Goals

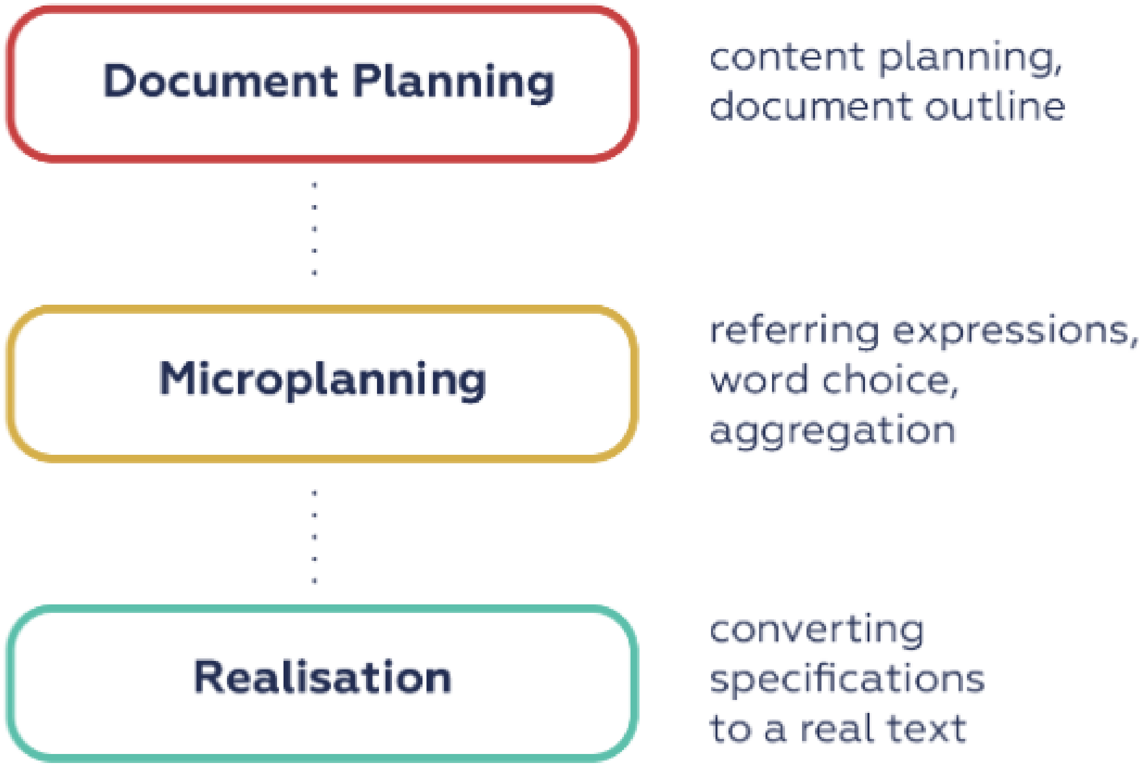


Pre-Neural NLG

Reminder

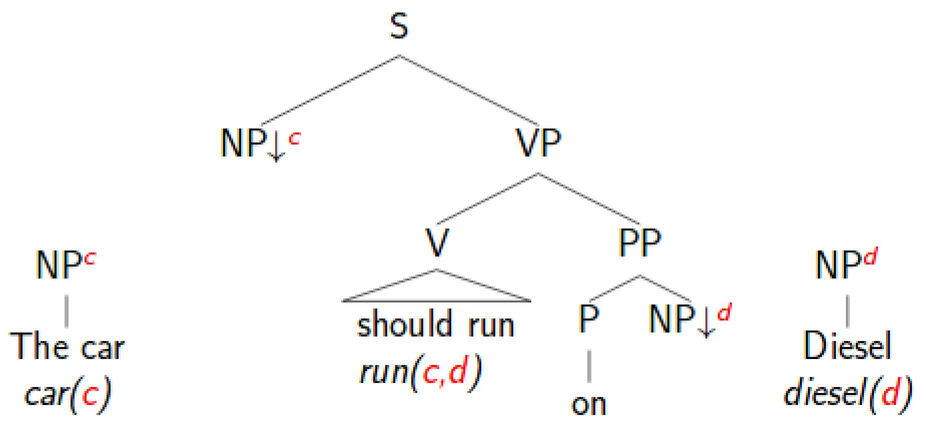
Generating from Data

The NLG Pipeline


Document Planning
content planning, document outline
Microplanning
referring expressions, word choice, aggregation
Realisation
converting specifications to a real text

Generating from Meaning Representations

Grammar

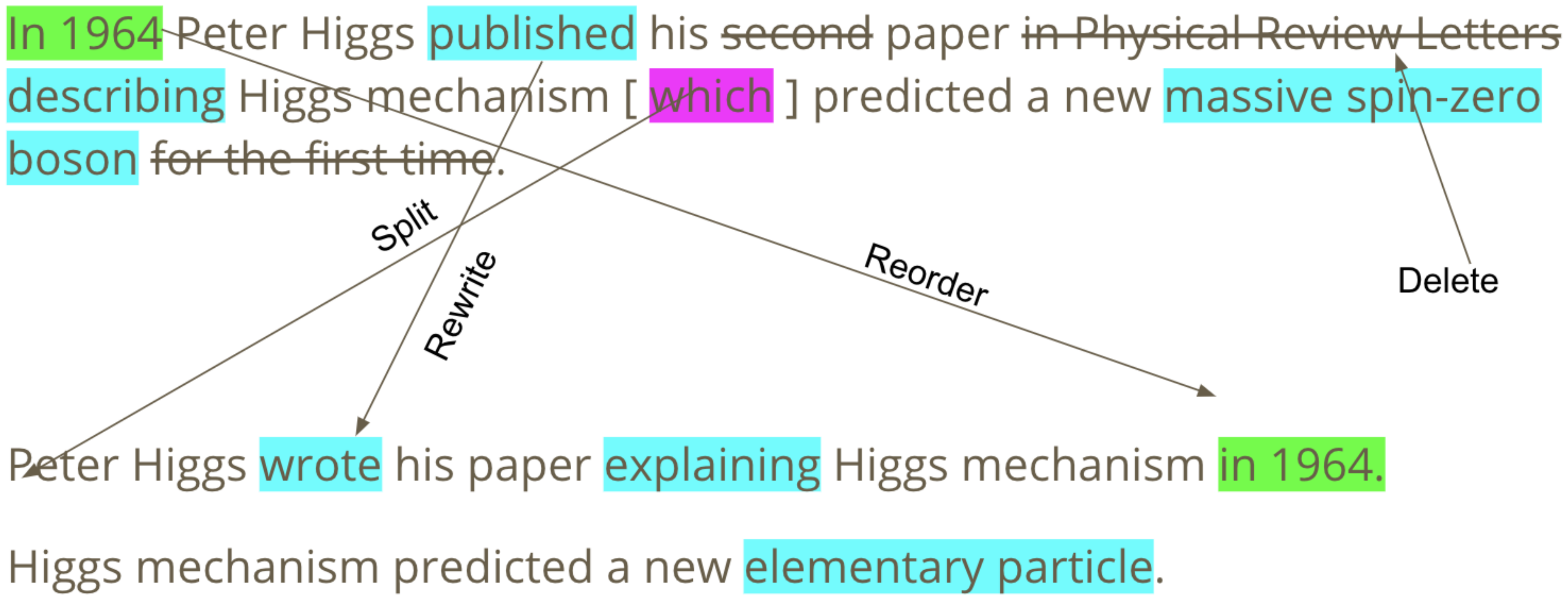


car(c), run(c,d), diesel(d)

Statistical modules

- Language models
  To choose between comparable intermediate results (the black cat/the cat black)
- Hypertaggers
  To prune the initial search space
- Rankers
  To determine the best output

Generating from Text



In 1964 Peter Higgs published his ~~second~~ paper ~~in Physical Review Letters~~ describing Higgs mechanism [which] predicted a new massive spin-zero boson ~~for the first time~~.

Peter Higgs wrote his paper explaining Higgs mechanism in 1964.

Higgs mechanism predicted a new elementary particle.

Lecture Plan

The Neural Approach to NLG

The Encoder-Decoder Framework

- With a Recurrent Encoder

Better Decoding

- Attention, Copy, Coverage

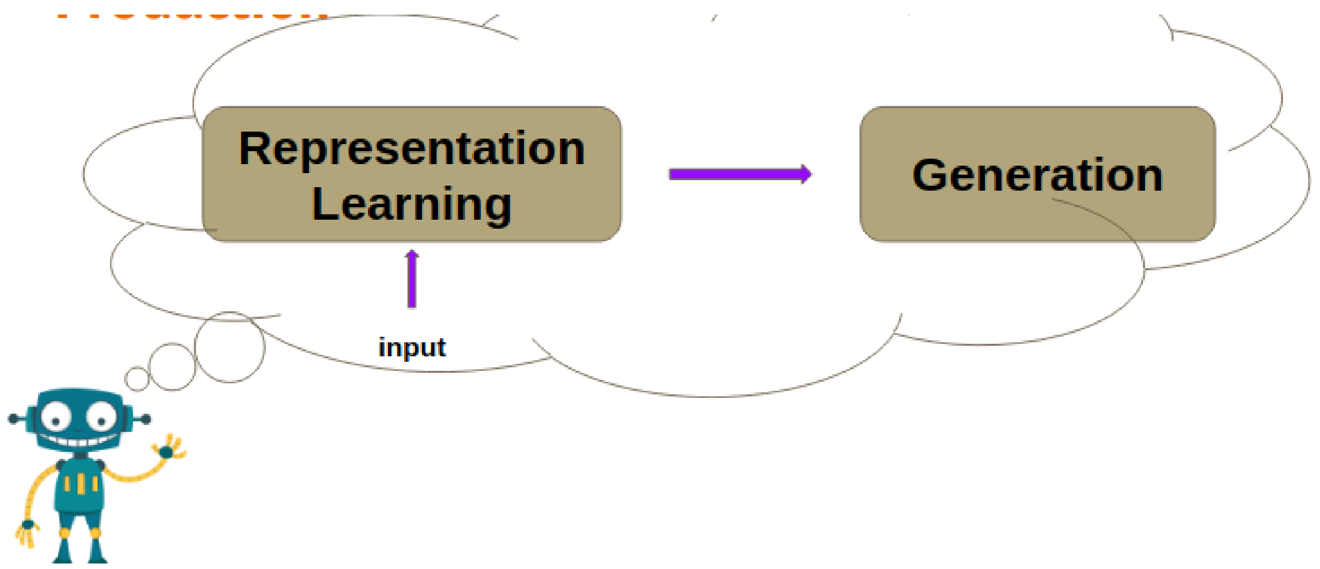
Encoding

- Recurrent Neural Network (RNN)
- Convolutional Neural Network (CNN)
- Graph-Based
- Hierarchical

Further Topics

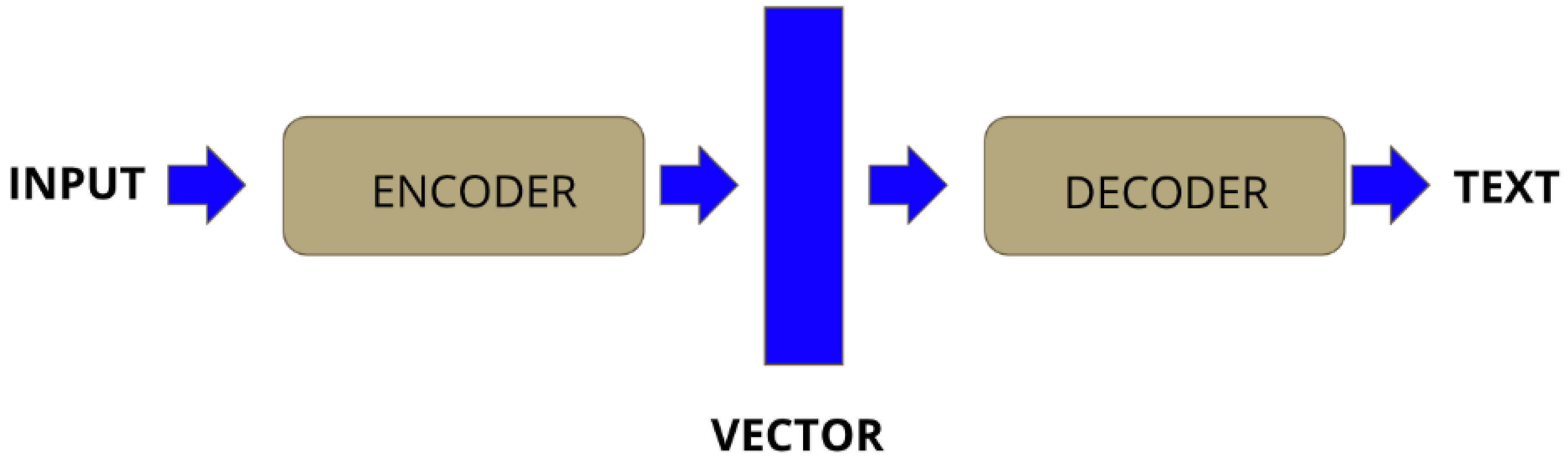
Neural Text Production

A single Encoder-Decoder framework for all text production tasks



- End to end: No sub-modules
- All types of input (data, text, meaning representation) are encoded into a numerical representation

The Encoder-Decoder Framework



The Encoder-Decoder Framework

Encoder

- Builds a representation for the input
- Converts the input to a real valued vector
- Commonly used encoders:
  - Recurrent: RNN, LSTM, GRU
  - Convolutional
  - Graph
  - Tranformer

Decoder

- A Recurrent network
- Generates text one word at a time
- Conditioned on input

The Encoder-Decoder Framework

Training and Learning

Training Data

- Parallel data
- (INPUT, OUTPUT)
- INPUT = text, meaning representation or data
- OUTPUT = text

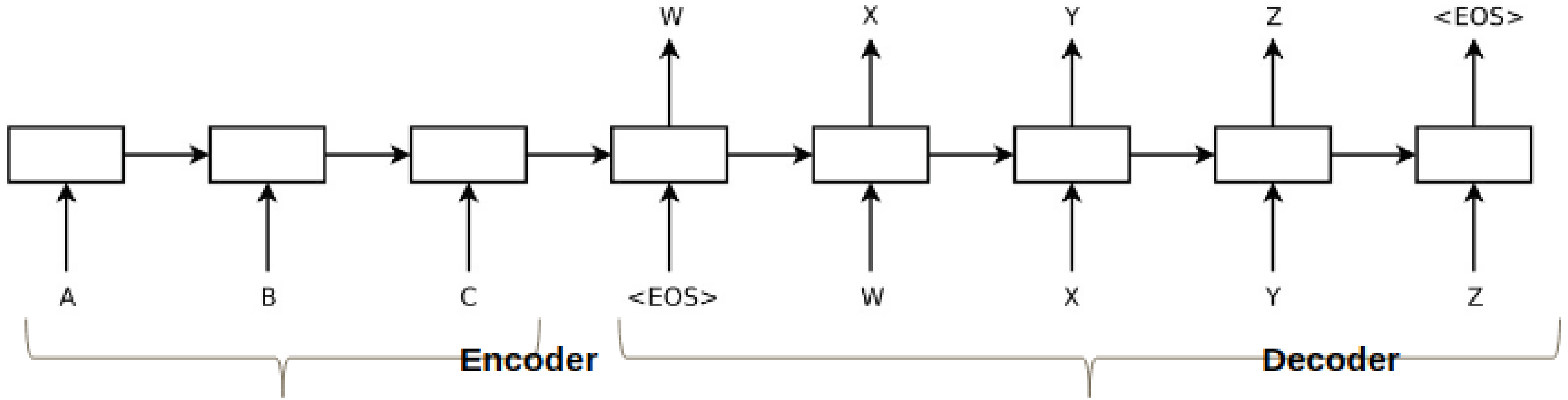
Learning

- Initialise weights with random values
- For each training example, the network makes a prediction
- This prediction is compared with the reference
- A loss is computed
- Backpropagation is used to minimise the loss and adjust the weights

Encoder-Decoder using a Recurrent Encoder

Encoder-Decoder Model

With a Recurrent Encoder

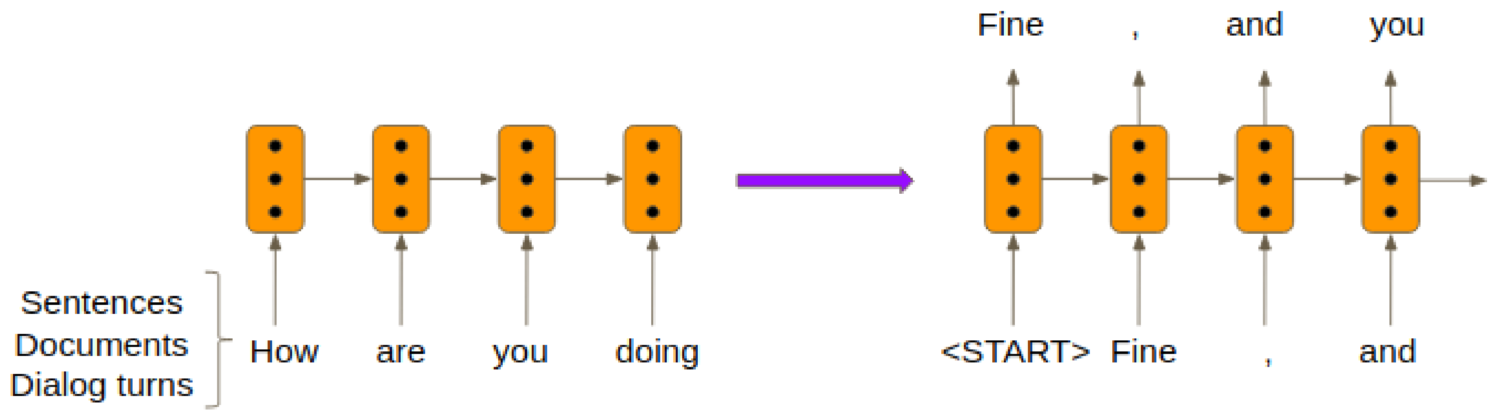


- The encoder processes each input token sequentially (one after the other)
- The input representation is generally taken to be the vector resulting from processing the last token in the input
- This input representation is a real-valued vector "representing" the whole input]

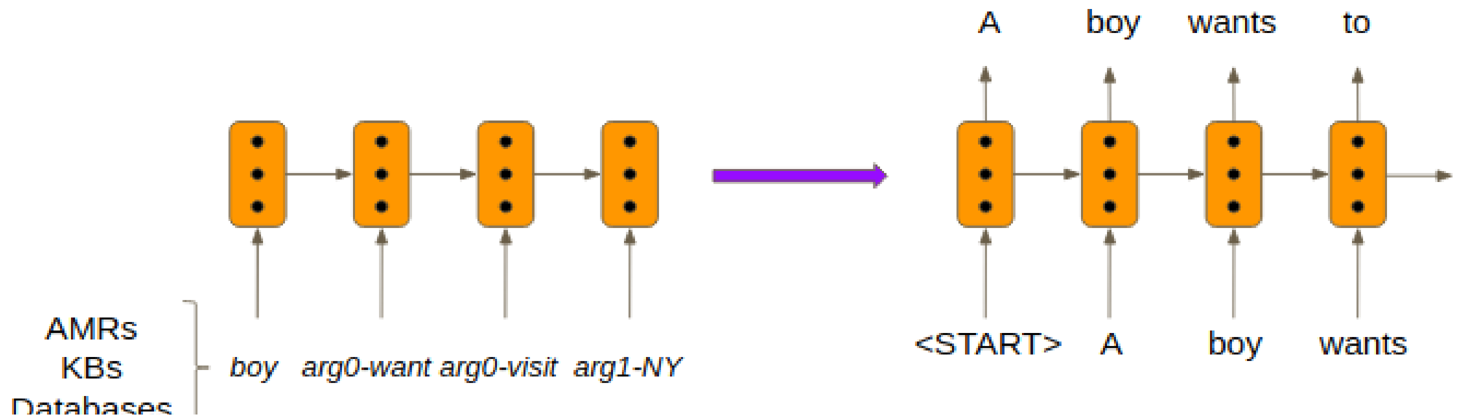
Recurrent Encoder

The input to NLG (text but also data and MRs) can be encoded using a recurrent network

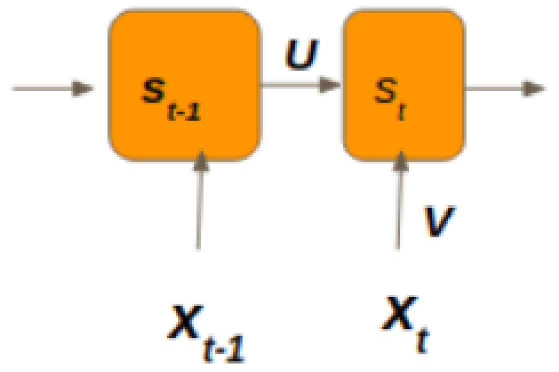
- Text



- Data or meaning representations (needs to be linearised first)



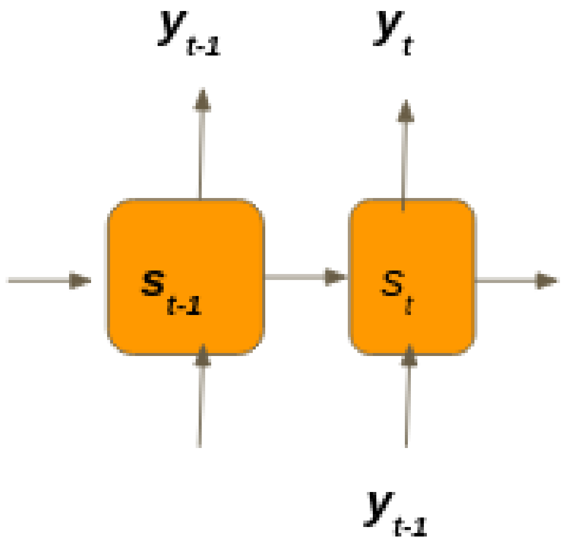
Encoding the Input using an RNN



$$s_t = tanh(U * s_{t-1} + V * x_t)$$

- x_i are vectors representing the input tokens (words, data or MR tokens)
- At each step, the encoder produces a new vector s_t (state) which represents the content of the preceding string of tokens
- The last state represents the meaning of the whole input
- U and V are the parameters learned during training
- tanh is a non linear function

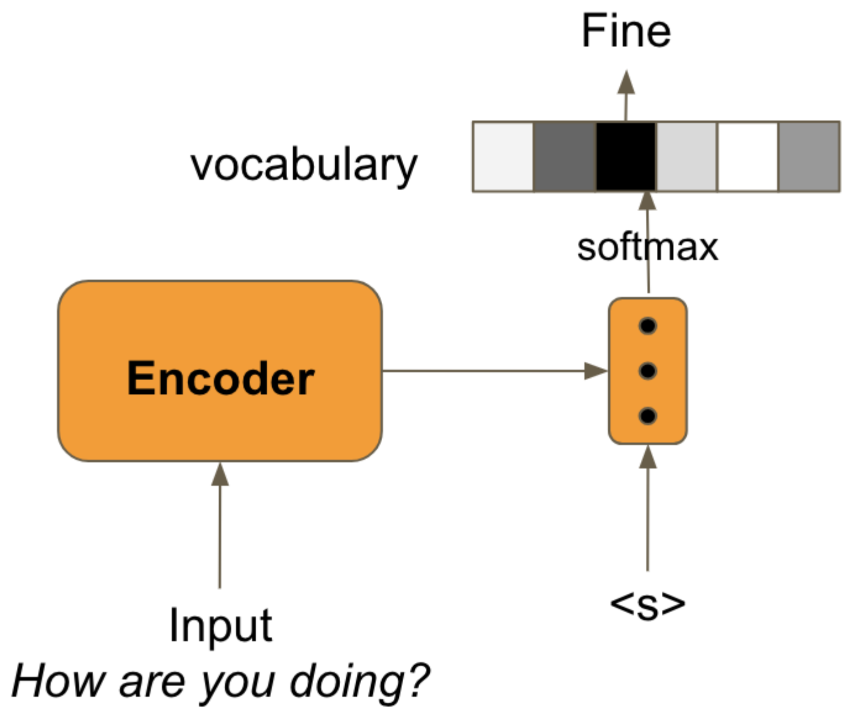
Decoding Words using an RNN



$$s_t = tanh(U * s_{t-1} + V * y_{t-1})$$
$$y_t = softmax(W * s_{t-1})$$

- y_t is the word predicted at time t
- s_t is the network state at time t
- Each new state is computed taking into account the previous state s_{t-1} and the last predicted word y_{t-1}.
- The softmax function turns a vectors of scores into a probability distribution
- At each time step t, the output/predicted token y_t is sampled from this probability distribution

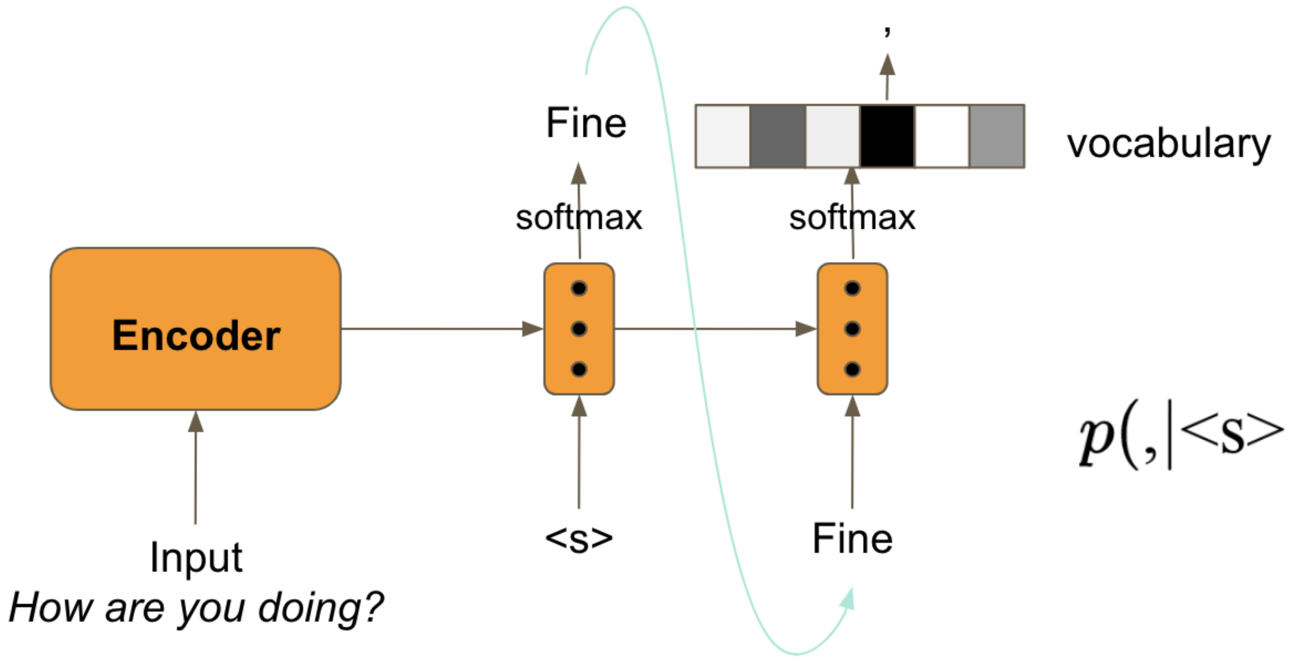
Generating Text using an RNN



$$p(\text{Fine}|\text{<s>}, \text{How are you doing?})$$

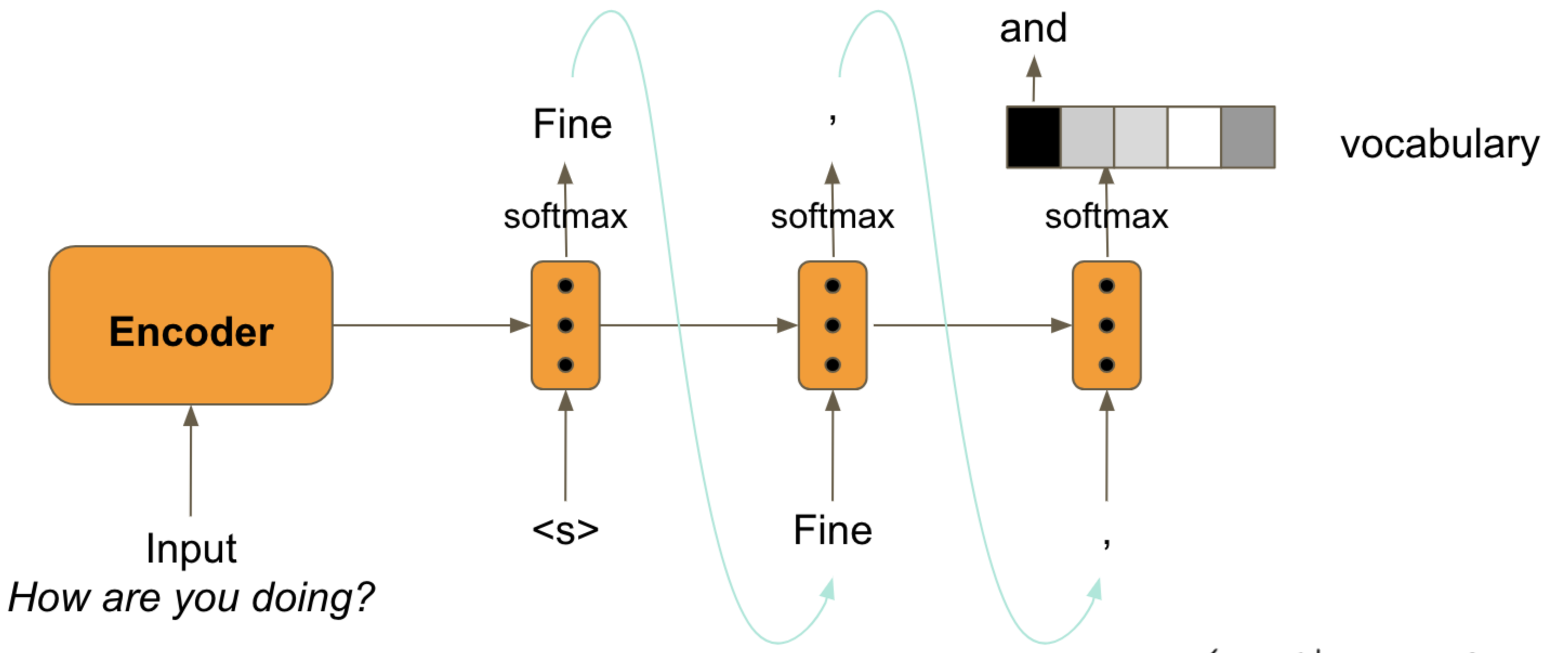
Conditional Generation

Generating Text using an RNN



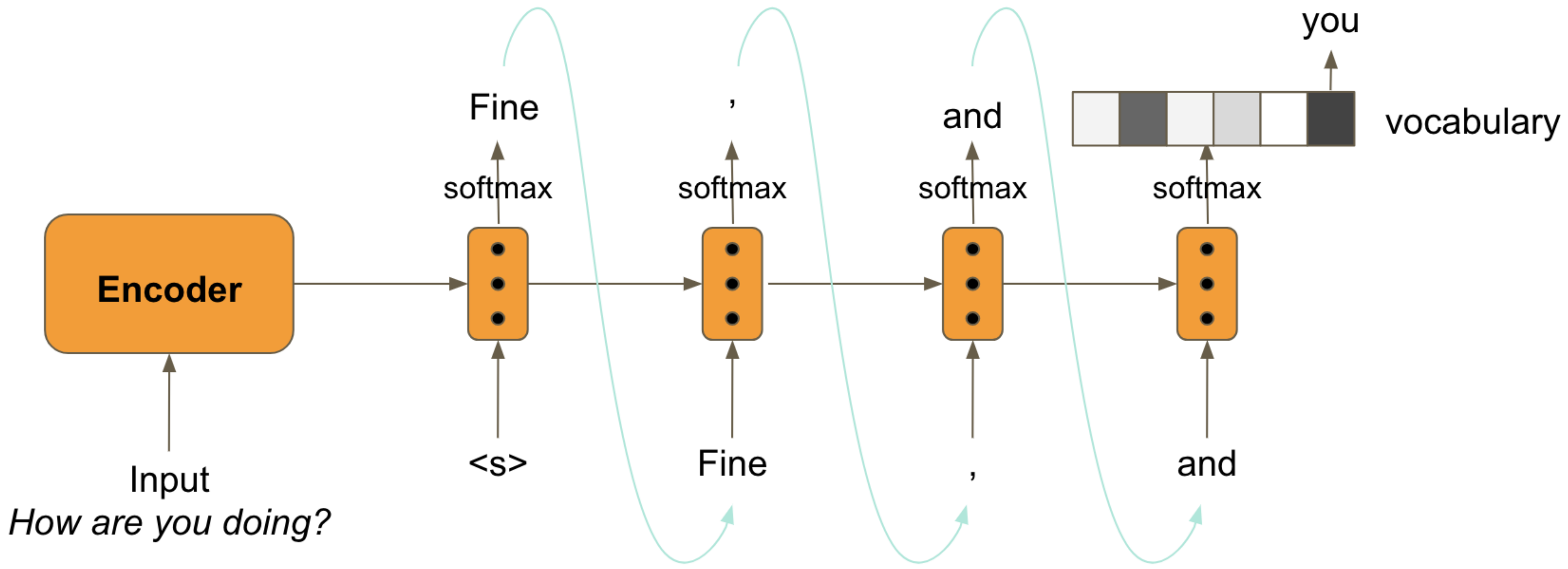
$$p(,|\text{<s> Fine, How are you doing?})$$

Generating Text using an RNN

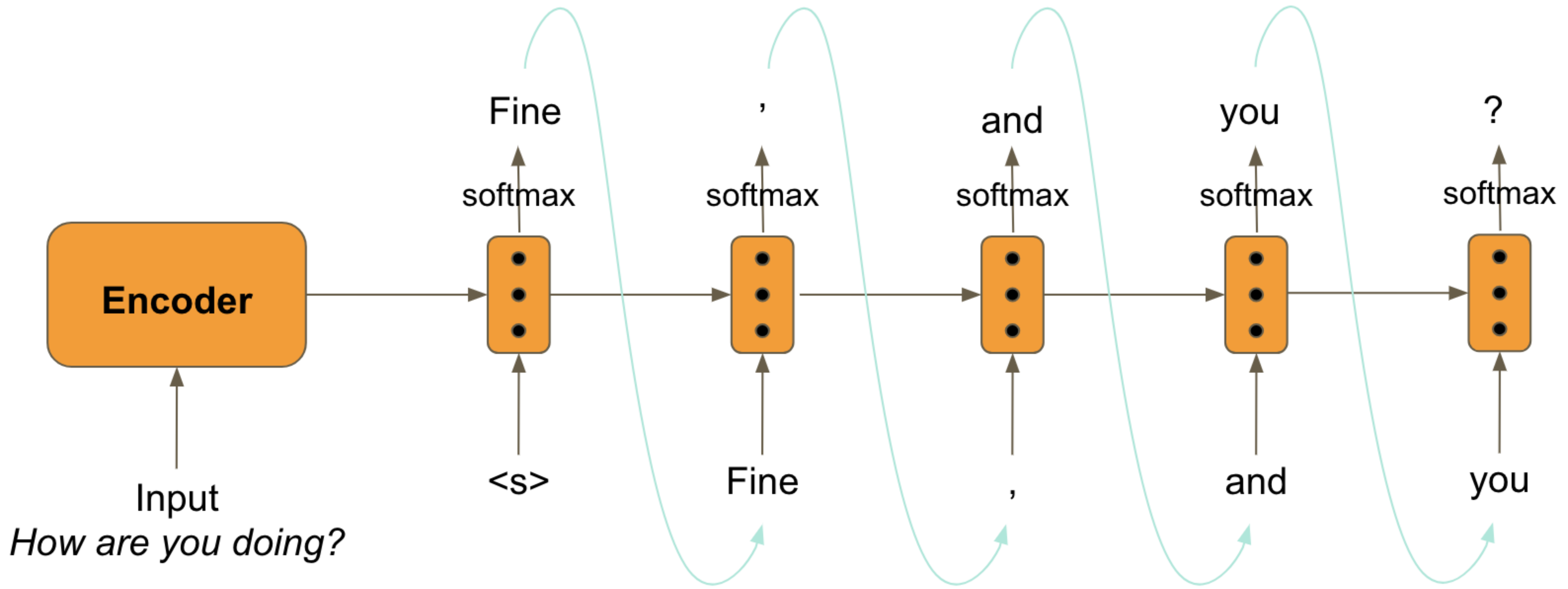


$$p(\text{and} | \text{<s> Fine,; How are you doing?})$$

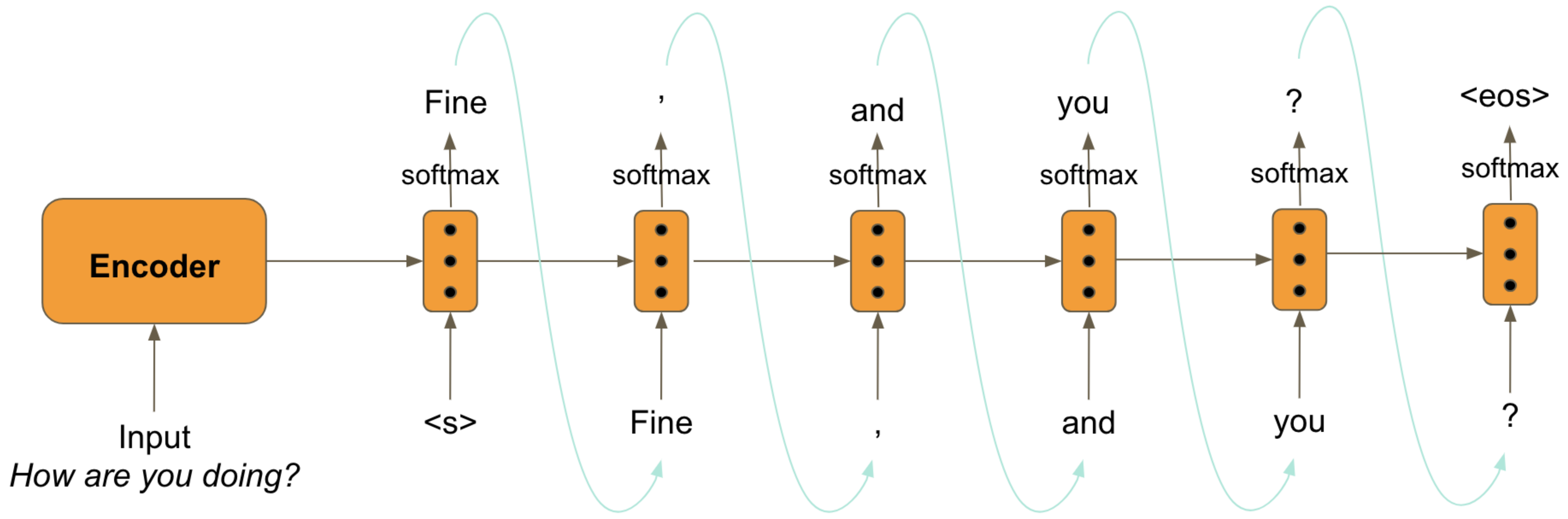
Generating Text using an RNN



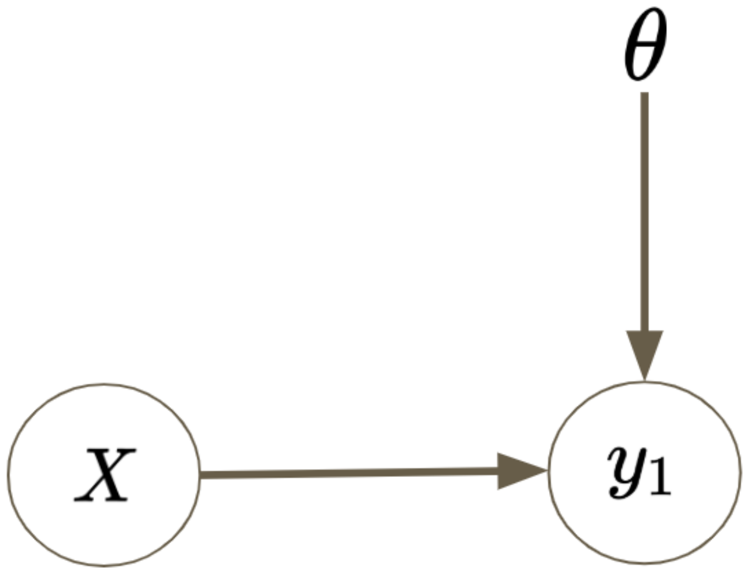
Generating Text using an RNN



Generating Text using an RNN



RNN = Conditional Language Model without Markov Assumption



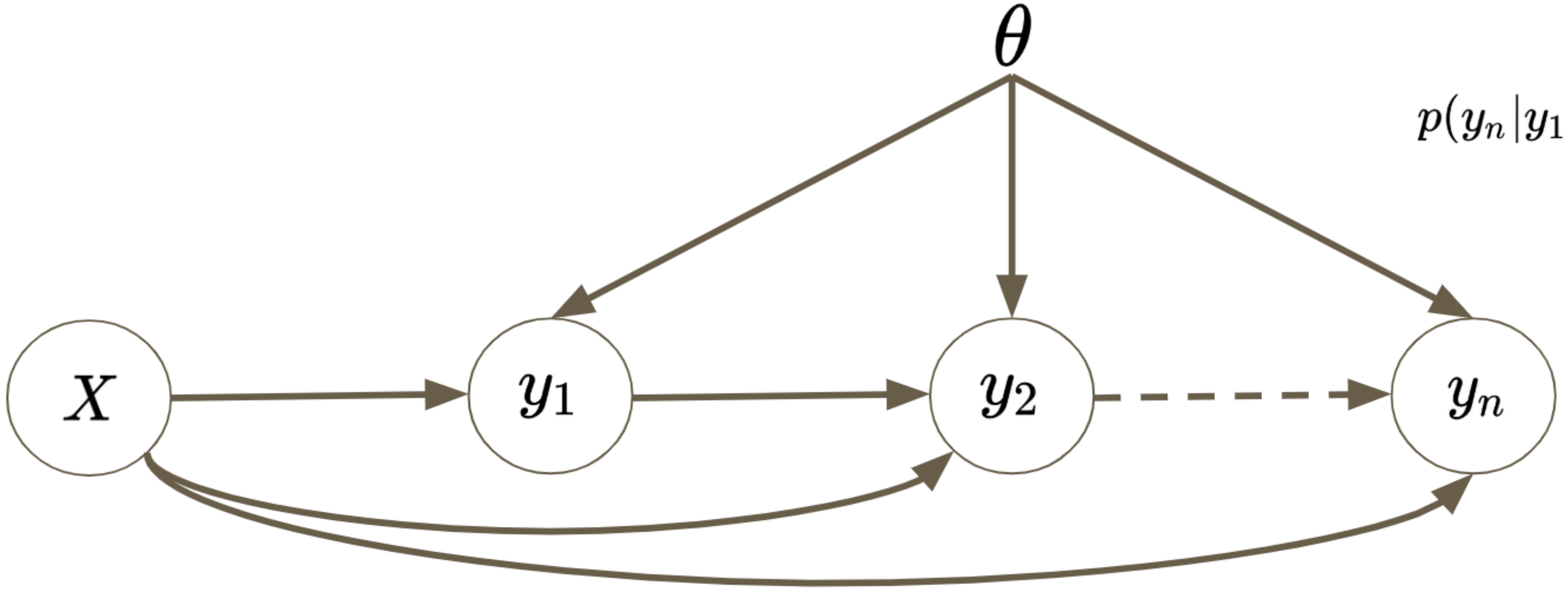
$p(y_1 | X; \theta)$

RNN = Conditional Language Model without Markov Assumption

$$\begin{tikzcd}
 & & & \theta \arrow[dl] \arrow[dr] & \\
X \arrow[rr] \arrow[rrrr, bend right] & & y_1 \arrow[rr] & & y_2
\end{tikzcd}$$

$p(y_2 | y_1, X; \theta)$

RNN = Conditional Language Model without Markov Assumption



$p(y_n | y_1, \ldots, y_{n-1}, X; \theta)$

Better Decoding

Decoding

Some Problems with Neural Generation

ACCURACY

- The output text sometimes contains information not present in the input.

REPETITIONS

- The output text sometimes contains repetitions

COVERAGE

- The output text sometimes does not cover all the input

RARE OR UNKNOWN WORDS

Decoding

Three Ways to Improve Decoding

Attention

- To improve accuracy

Copy

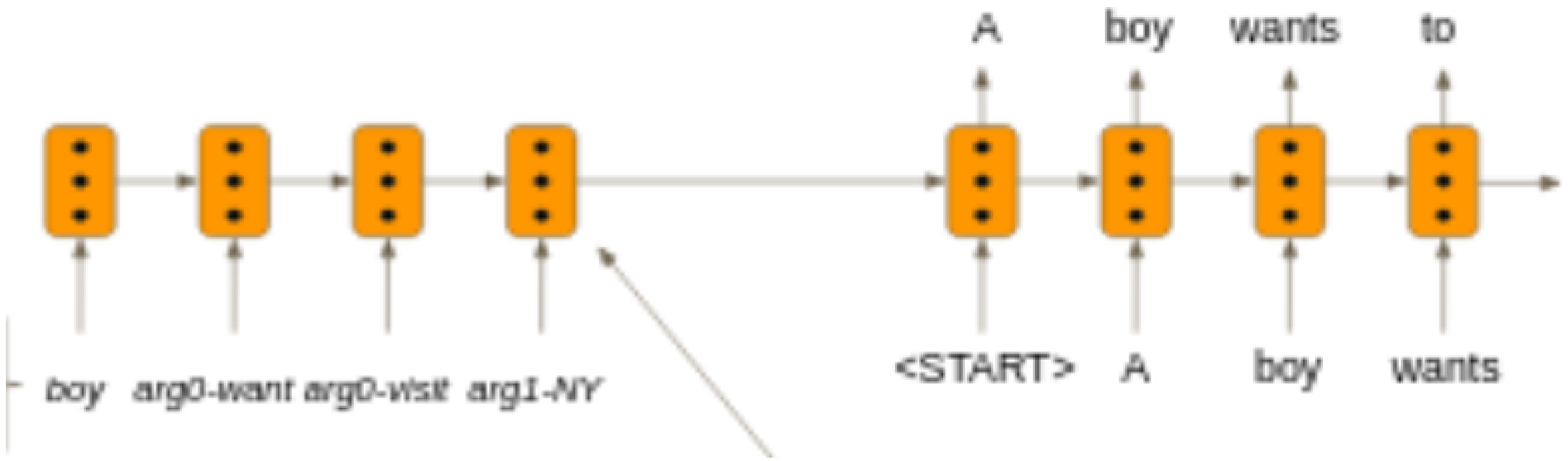
- To copy from the input
- To handle rare or unknown words

Coverage

- To help cover all and only the input
- To avoid repetitions

Attention

Standard RNN Decoding



- The input is compressed into a fixed-length vector
- Performance decreases with the length of the input [Sutskever et al. 2014]]

Decoding with Attention

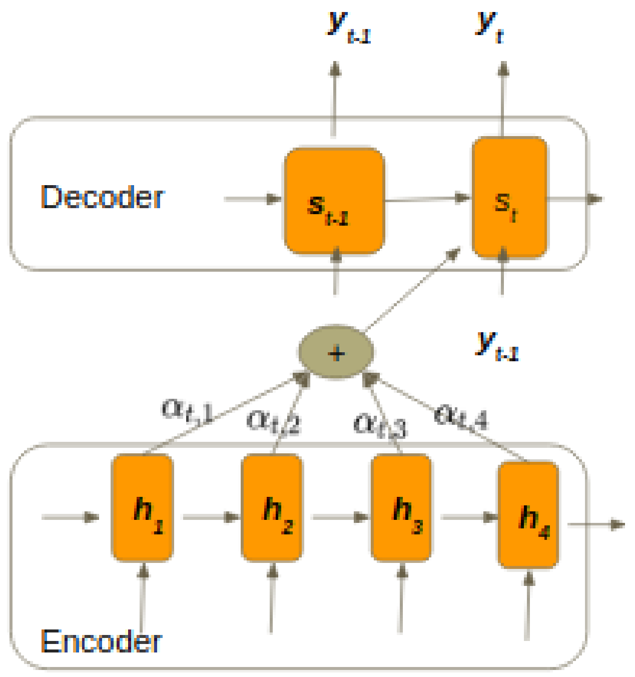
Input

- the previous state s_{t-1}
- the previously generated token y_{t-1} and
- a context vector c_t

Context vector

- depends on the previous state and therefore changes at each step
- indicates which part of the input is most relevant to the decoding step

RNN with Attention



α can be viewed as a probability distribution over the source words

- A score is computed between each input token encoder state and the current state

$$\alpha_{t,j} = v^t anh(W_h \times h_j + W_s \times s_t + b)$$

- The context vector is the weighted sum of the encoder states

$$c_t = \sum_{j=1}^{T_X} \alpha_{t,j}.h_j$$

- The new state is computed taking into account this context vector.

$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$

- The next predicted token is sampled from the new target vocabulary distribution

$$P_{vocab} = softmax(W * s_t)$$

Copy

Copy

Motivation

- To copy from the input (E.g., in Text Summarisation applications)
- To handle rare or unknown words

Method

- At each time step, the model decides whether to copy from the input or to generate from the target vocabulary.

- Two probabilities for each token
  - P_{target}: the probability to sample it from the target vocabulary i.e., to generate that word.
  - P_{source}: the probability to sample it from the source vocabulary i.e., to copy that word.

Copy

P_{gen}

A soft switch to choose between generating a word from the vocabulary by sampling from P_{vocab}, or copying a word from the input sequence by sampling from the attention distribution α_t.

$$P_{gen} = \sigma(W_c \times c_t + W_s \times s_t + W_y \times y_{t-1})$$

Final Probability Distribution

(over source and target vocabulary)

$$P(w) = p_{gen} * P_{target}(w) + (1 - p_{gen}) * P_{source}$$

- $P(w)$, probability of generating word w
- P_{source}, probability of copying word from the source
  $= \sum_{i:w=w_i} \alpha_{t,i}$
  $= 0$ if w is not in the input
- P_{target}, probability of generating word
  = 0 if w is not in the target vocabulary

Coverage

Coverage

Problem

- Neural models tend to omit or repeat information from the input

Solution

(Tu et al. 2017)

- Coverage: cumulative attention, what has been attended to so far
- Use coverage as extra input to attention mechanism
- Loss: Penalises attending to input that has already been covered

Coverage

Coverage in Summarisation

- A coverage vector k_t captures how much attention each input words has received

$$k_t = \sum_{t=0}^{t-1} \alpha_t$$

- The attention mechanism takes coverage into account

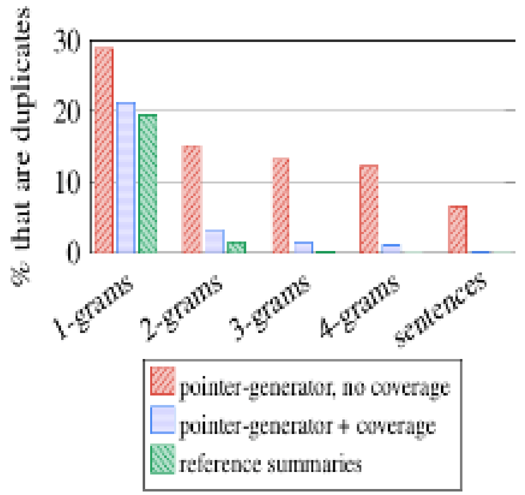
$$\alpha_{t,j} = v^t anh(W_h \times h_j + W_s \times s_t + b)$$

- The loss penalizes repeatedly attending to the same location

$$loss_t = -logP(w_t) + \lambda \sum_j min(\alpha_{t,j}, k_{t,j})$$

Coverage

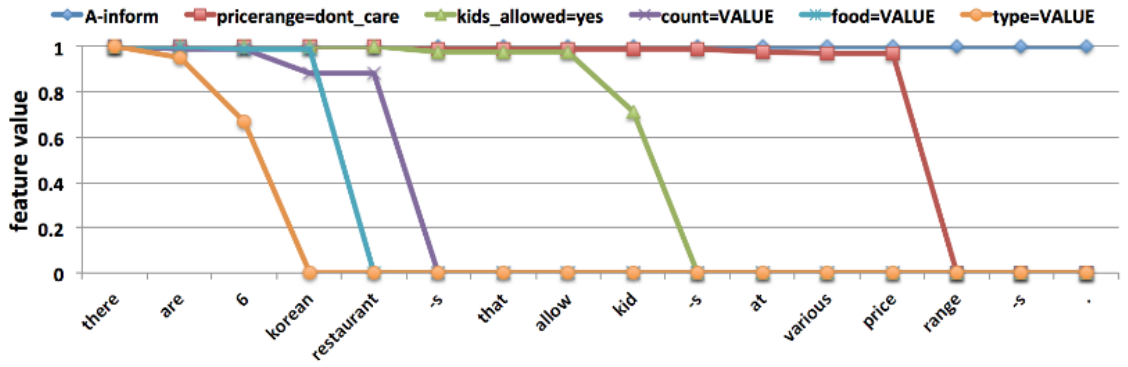
Impact of Coverage on Duplicate N-Grams



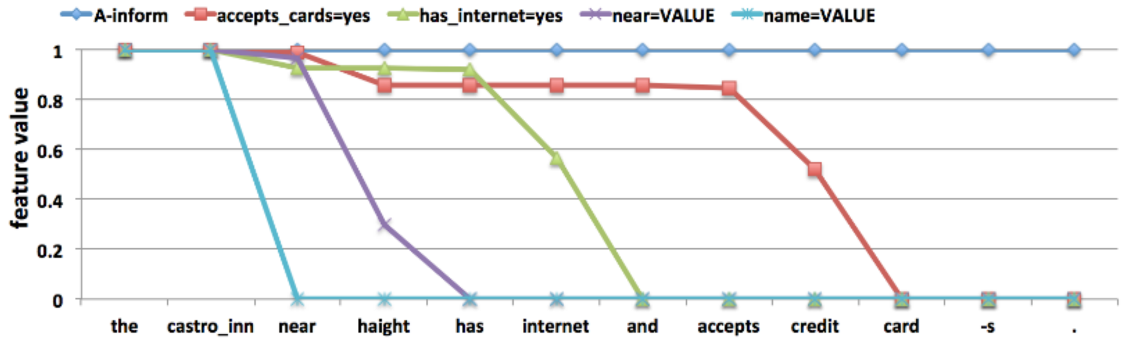
- Coverage successfully eliminates repetitions.
- The proportion of duplicate n-grams is similar in the reference summaries and in the summaries produced by the model with coverage.

Coverage in Text Production

- Dialog: SC-LSTM (Wen et al. 2015)



(a) An example realisation from SF restaurant domain



(b) An example realisation from SF hotel domain

Figure 3: Examples showing how the SC-LSTM controls the DA features flowing into the network via its learned semantic gates. Despite errors due to sparse training data for some slots, each gate generally learned to detect words and phrases describing a particular slot-value pair.

Copy and Coverage in Summarisation

Session 6: 03/10

Neural Text-to-Text Generation

SABDENOV Aidos

Abigail See, PeterJ. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073--1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. [DOI | http]

Neural sequence-to-sequence models have provided a viable new approach for abstractive text summarization (meaning they are not restricted to simply selecting and rearranging passages from the original text). However, these models have two shortcomings: they are liable to reproduce factual details inaccurately, and they tend to repeat themselves. In this work we propose a novel architecture that augments the standard sequence-to-sequence attentional model in two orthogonal ways. First, we use a hybrid pointer-generator network that can copy words from the source text via pointing, which aids accurate reproduction of information, while retaining the ability to produce novel words through the generator. Second, we use coverage to keep track of what has been summarized, which discourages repetition. We apply our model to the CNN / Daily Mail summarization task, outperforming the current abstractive state-of-the-art by at least 2 ROUGE points.

Rare Words

Rare Words

Three methods

- Copying
- Delexicalisation
- Character-Based Network
  - smaller vocabulary
  - unknown words handled by copying characters

Rare Words

Delexicalisation

- Slot values occurring in training utterances are replaced with a placeholder token representing the slot
- At generation time, these placeholders are then copied over from the input specification to form the final output

inform(restaurant name = Au Midi , neighborhood= midtown , cuisine = french)

Au Midi is in Midtown and serves French food .

inform(restaurant name = **restaurant name**, neighborhood= **neighborhood**, cuisine = **cuisine**)

restaurant name is in **neighborhood** and serves **cuisine** food.

Session 4: 26/09, 10am-12pm

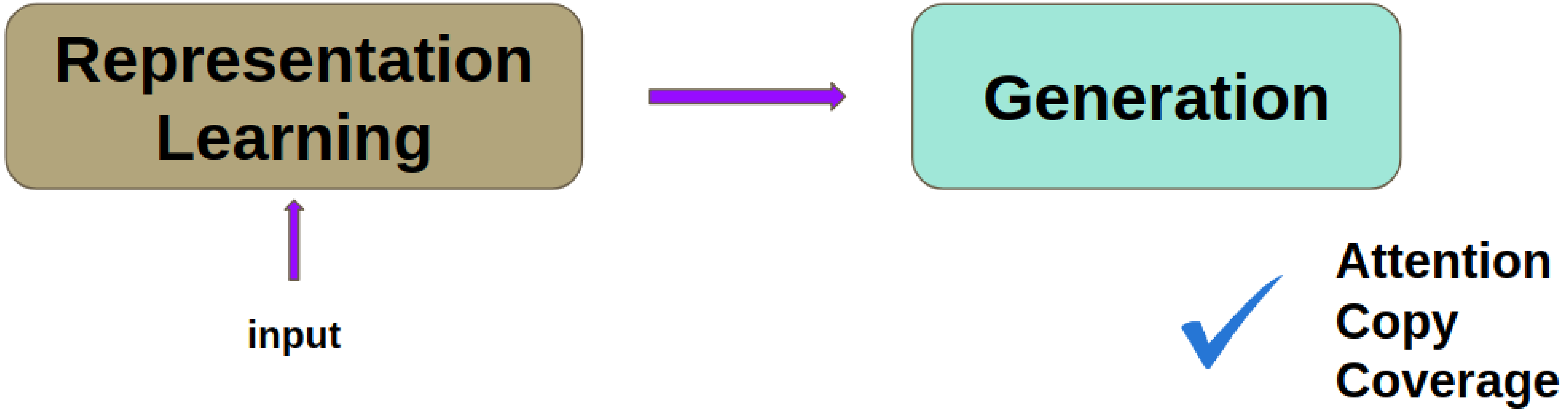
Pre-Neural Text to-Text Generation

LEAVITT Phyllicia

Glorianna Jagfeld, Sabrina Jenne, and NgocThang Vu. Sequence-to-sequence models for data-to-text natural language generation: Word-vs. character-based processing and output diversity. *arXiv preprint arXiv:1810.04864*, 2018. [http]

We present a comparison of word-based and character-based sequence-to-sequence models for data-to-text natural language generation, which generate natural language descriptions for structured inputs. On the datasets of two recent generation challenges, our models achieve comparable or better automatic evaluation results than the best challenge submissions. Subsequent detailed statistical and human analyses shed light on the differences between the two input representations and the diversity of the generated texts. In a controlled experiment with synthetic training data generated from templates, we demonstrate the ability of neural models to learn novel combinations of the templates and thereby generalize beyond the linguistic structures they were trained on.

Better Decoding



Better Encoding ?

Better Encoding

Text

- Long Distance Dependencies: LSTM, GRU, bi-LSTM
- Salient Text Fragments: Convolutional sentence encoders
- Text Structure: Hierarchical encoders

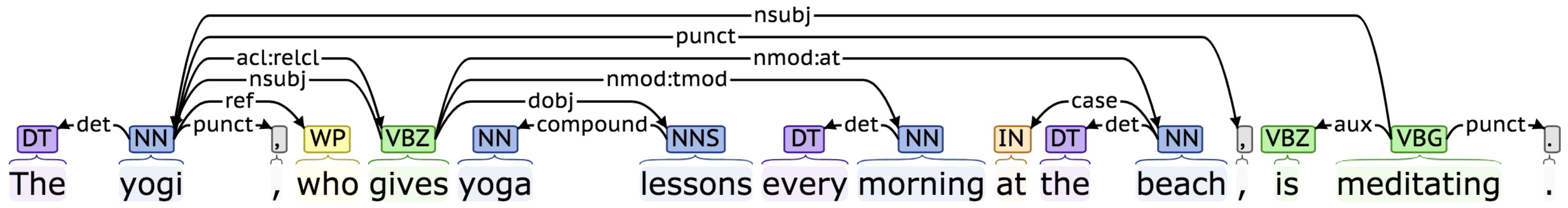
Data and MR structure

- Encoding Graphs

LSTM, GRU, bi-LSTM

Two Problems with RNNs

Long Distance Dependencies



- In practice, RNNs cannot handle long range dependency because of the Vanishing and Exploding Gradients issue (Bengio et al. 1994)
- LSTMs and GRUs are alternative RNNs which helps with this

Directionality

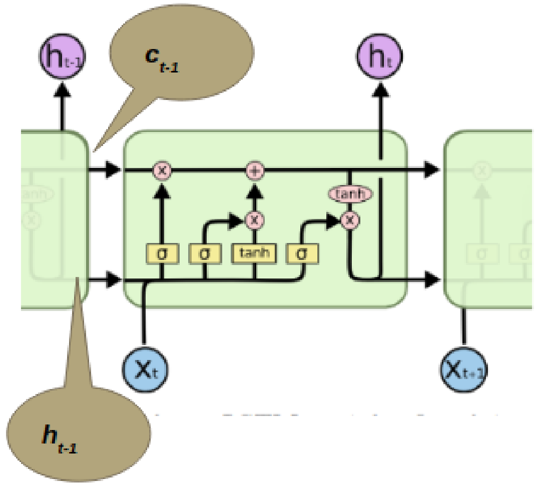
- RNNs proceed left-to-right through the sequence
- The right context of a token is often helpful to capture its meaning
- bi-directional RNNs permit taking into account both left- and right-contexts.

RNN and Long Distance Dependencies

LSTM

Long Short Term Memory Network

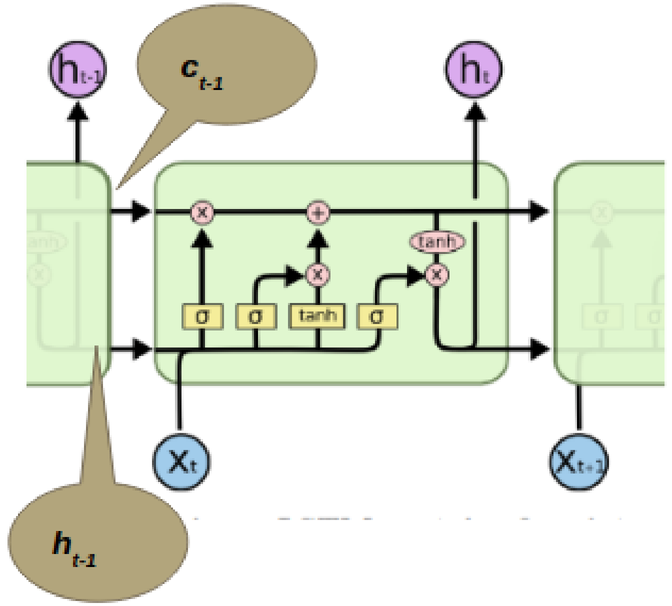
RNN with Memory



	$\tilde{c}_t = tanh(W_c * [h_{t-1}, x_t])$
update	$u_t = \sigma(W_t * [h_{t-1}, x_t])$
forget	$f_t = \sigma(W_f * [h_{t-1}, x_t])$
output	$o_t = \sigma(W_o * [h_{t-1}, x_t])$
	$c_t = u_t * \tilde{c}_t + f_t * c_{t-1}$
	$h_t = o_t * tanh(c_t)$

- Uses a cell and three gates
- The cell keeps track of what is kept, forgotten and updated
- Gates are a way to optionally let information through.
- The gates sigmoid layer outputs numbers between 0 and 1, describing how much of each component should be let through (0 = “forget”, 1 = “keep”).

LSTM



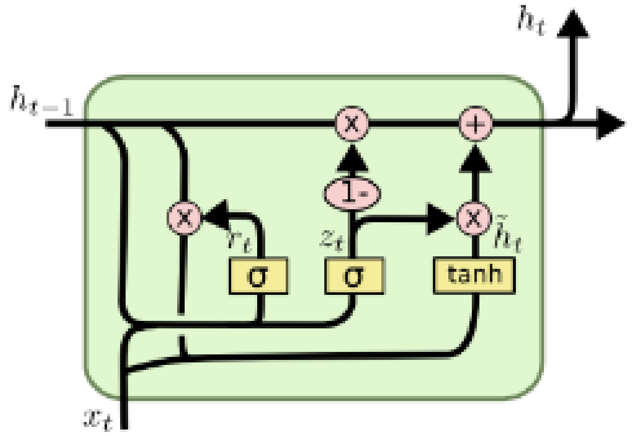
$$\begin{aligned}
\tilde{c}_t &= tanh(W_c * [h_{t-1}, x_t]) \\
\text{update} \quad u_t &= \sigma(W_t * [h_{t-1}, x_t]) \\
\text{forget} \quad f_t &= \sigma(W_f * [h_{t-1}, x_t]) \\
\text{output} \quad o_t &= \sigma(W_o * [h_{t-1}, x_t]) \\
c_t &= u_t * \tilde{c}_t + f_t * c_{t-1} \\
h_t &= o_t * tanh(c_t)
\end{aligned}$$

- The forget gate looks at the preceding state and current input token and decides which values to keep/forget in the preceding cell state C_{t-1}
- The tanh layer creates a vector of new candidate values that could be added to the state.
- The u_t gate decides which values to update in this candidate state.
- The new candidate state and cell are added
- The output gate is applied to the result to create the output state

See Colah's Blog for a detailed explanation.

GRU

A simpler way to decide what to forget and what to memorize



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

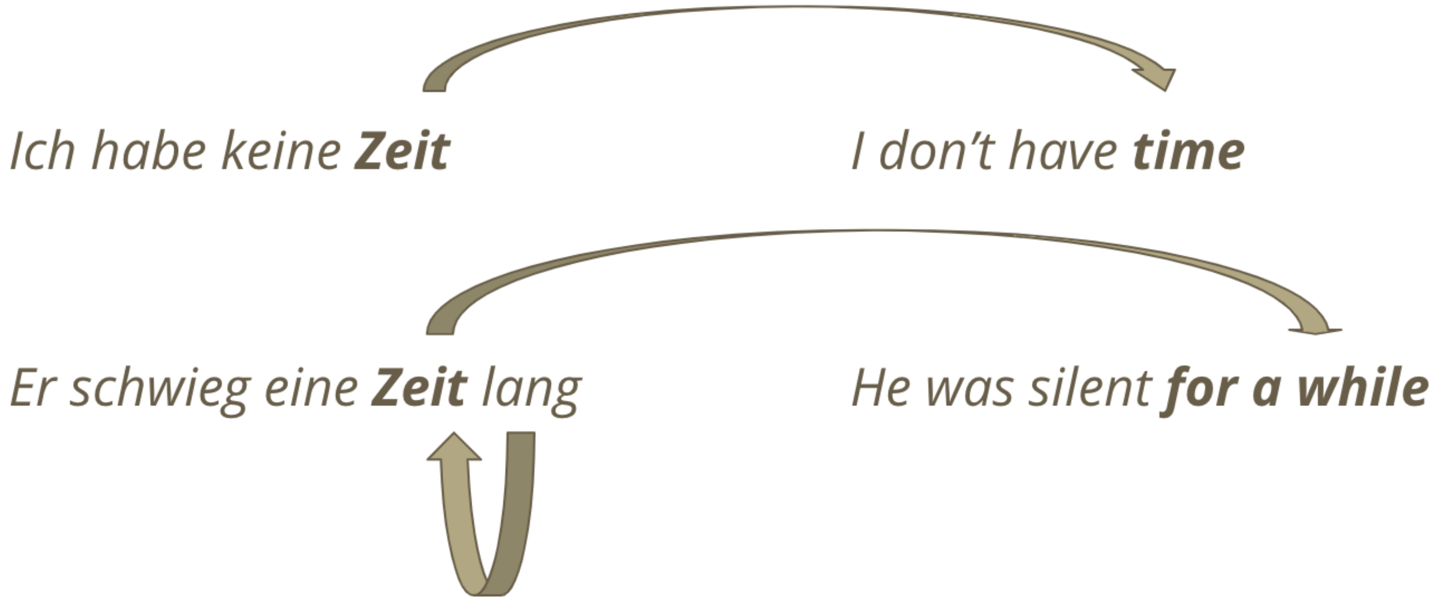
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- Creates a candidate state using a reset gate (r_t)
- Applies an update gate (z_t) to that candidate state
- Applies the corresponding opposite gate ($1 - z_t$) to the previous state
- Add the updated candidate state and previous state to create the output state

RNNs and Directionality

Bi-LSTM

Both left and right context matter



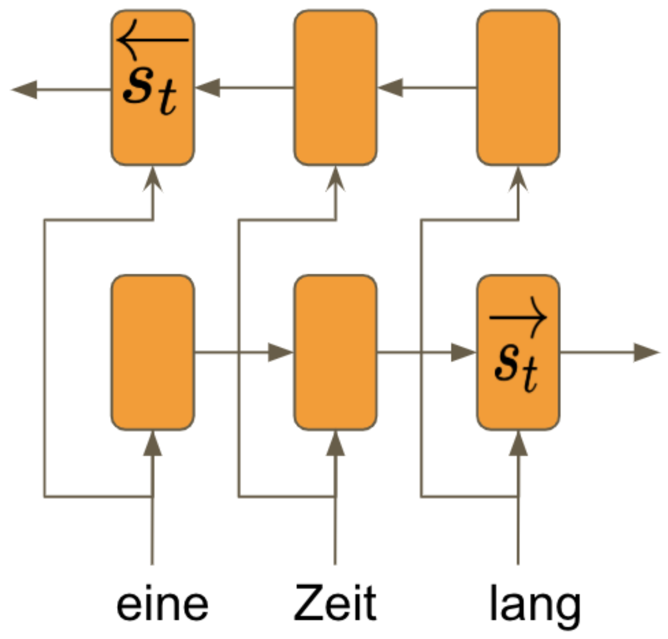
*Ich habe keine **Zeit***

*I don't have **time***

*Er schwieg eine **Zeit** lang*

*He was silent **for a while***

Bidirectional RNN



Input representation

$$[\overrightarrow{s_t}\,,\,\overleftarrow{s_t}]$$

- Forward RNN encodes left context
- Backward RNN encodes right context
- Forward and backward states are concatenated

Convolutional Neural Networks (CNN)

CNN

History and Motivations

Massively used in Computer Vision. Also used in NLP.

Efficient at identifying properties from the input that correlate well with the output

- Caption Generation: CNN learns to align objects in images to words in caption

Less computationally expensive then RNN

- The computations involved (linear computation of the convolutional layer followed by a non linearity) are much lighter than the cell computation involved in LSTMs.
- There is no temporal dependencies between filters, so they can be applied concurrently

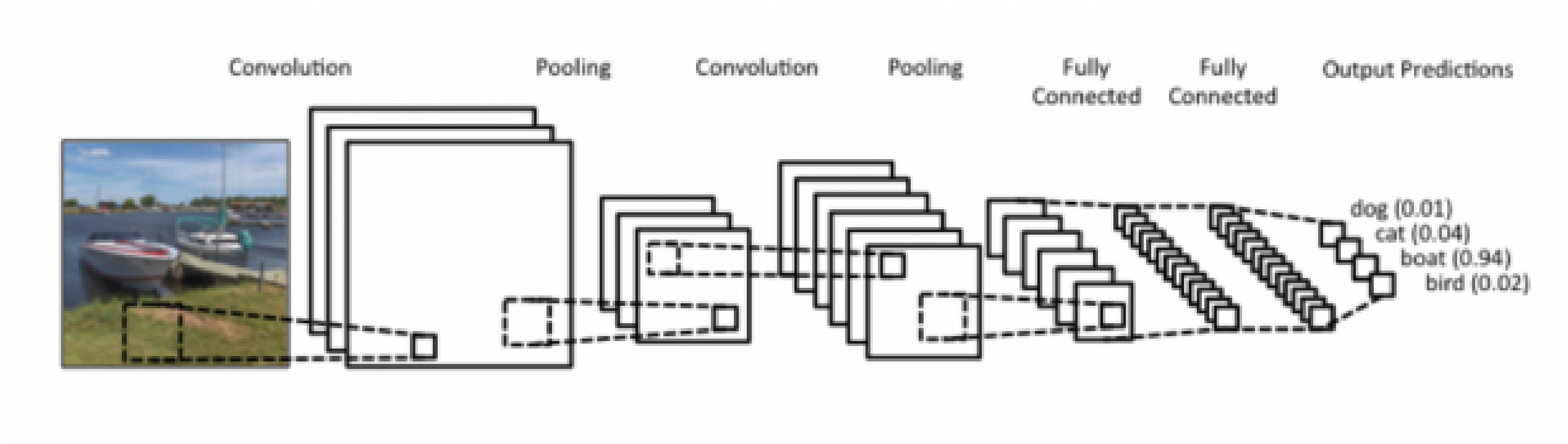
Can also capture long range dependencies by hierarchically increasing the receptive field.

CNN

Stack of Convolutional, ReLU and Pool layers

Slides (convolves) one or more filters over the input

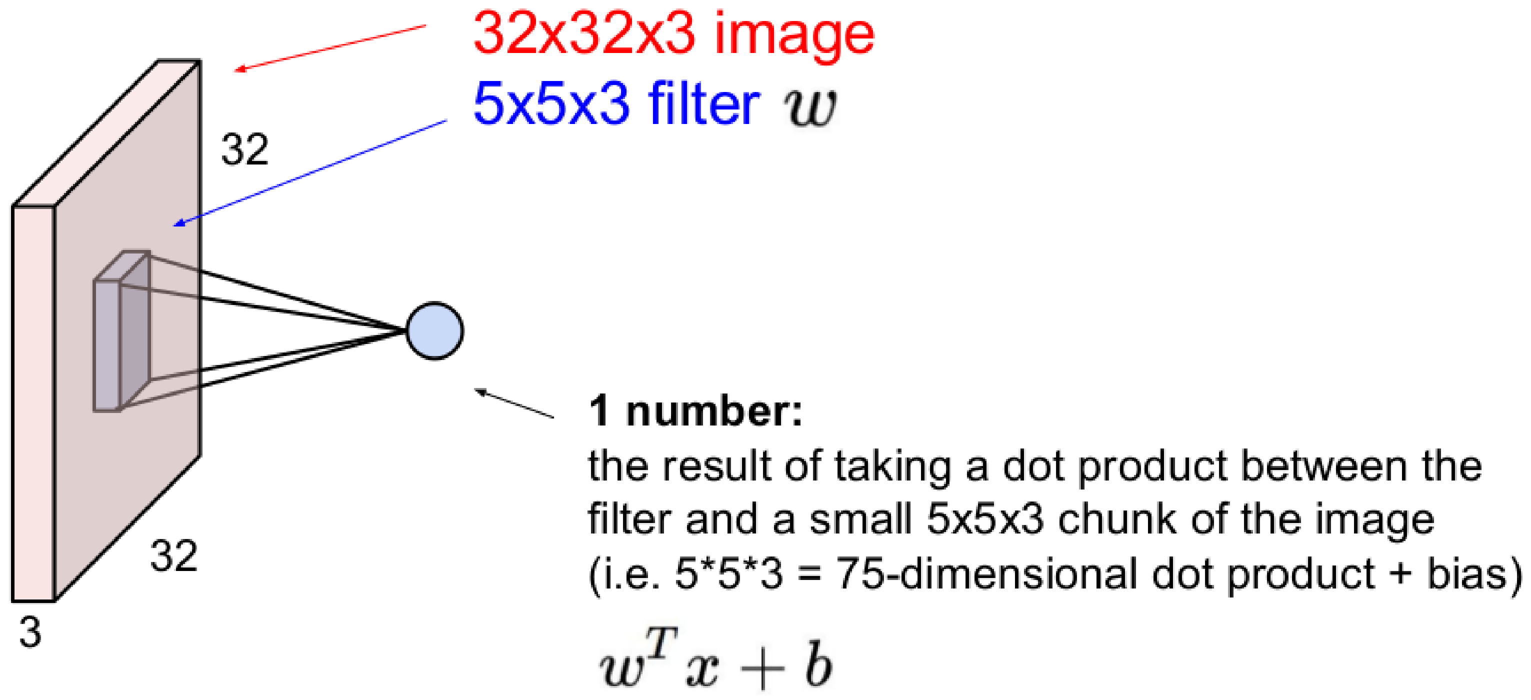
Applies non-linearity and max-pooling operations



CNN

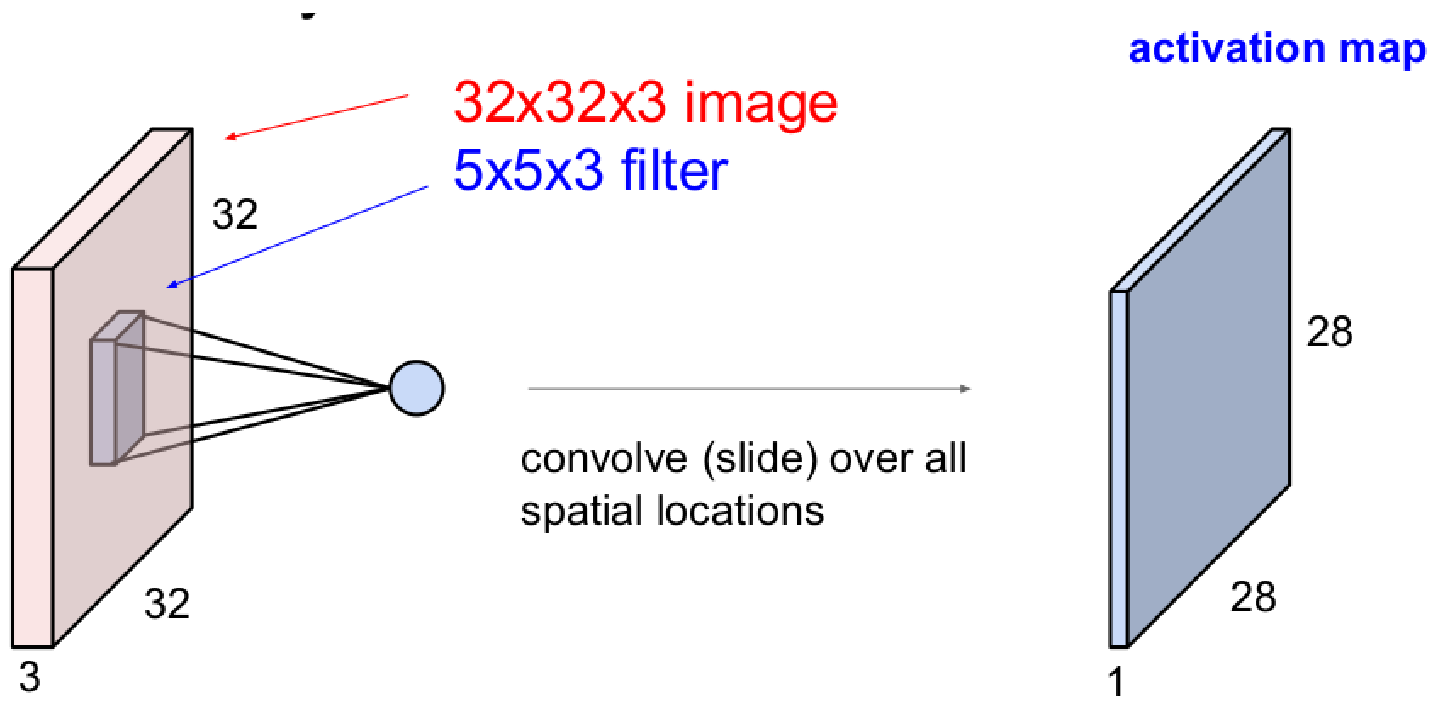
Convolution Layer

The filter is applied to a small chunk of the input and returns a real number (the activation value) for that input chunk


32x32x3 image
5x5x3 filter w
32
32
3
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)
$w^T x + b$

CNN

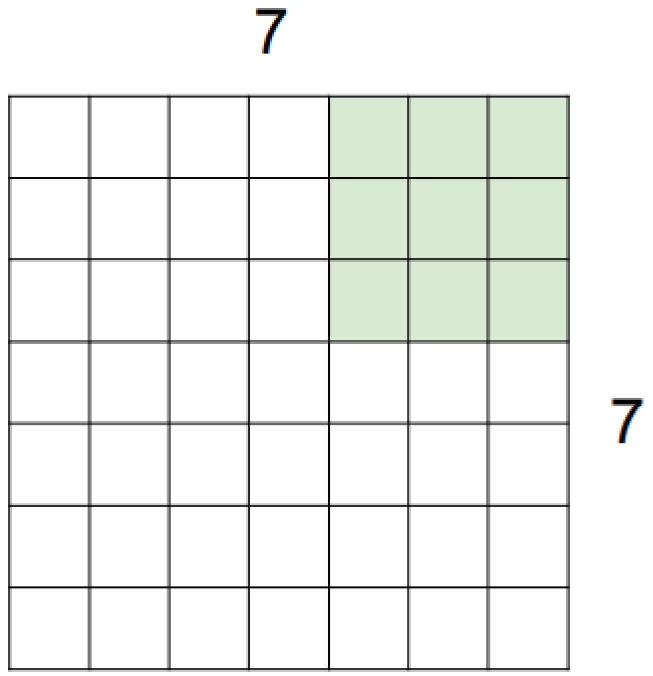
Activation Map



- The filter is applied over all spatial locations in the input
- There can be several filters of various size

CNN

Stride

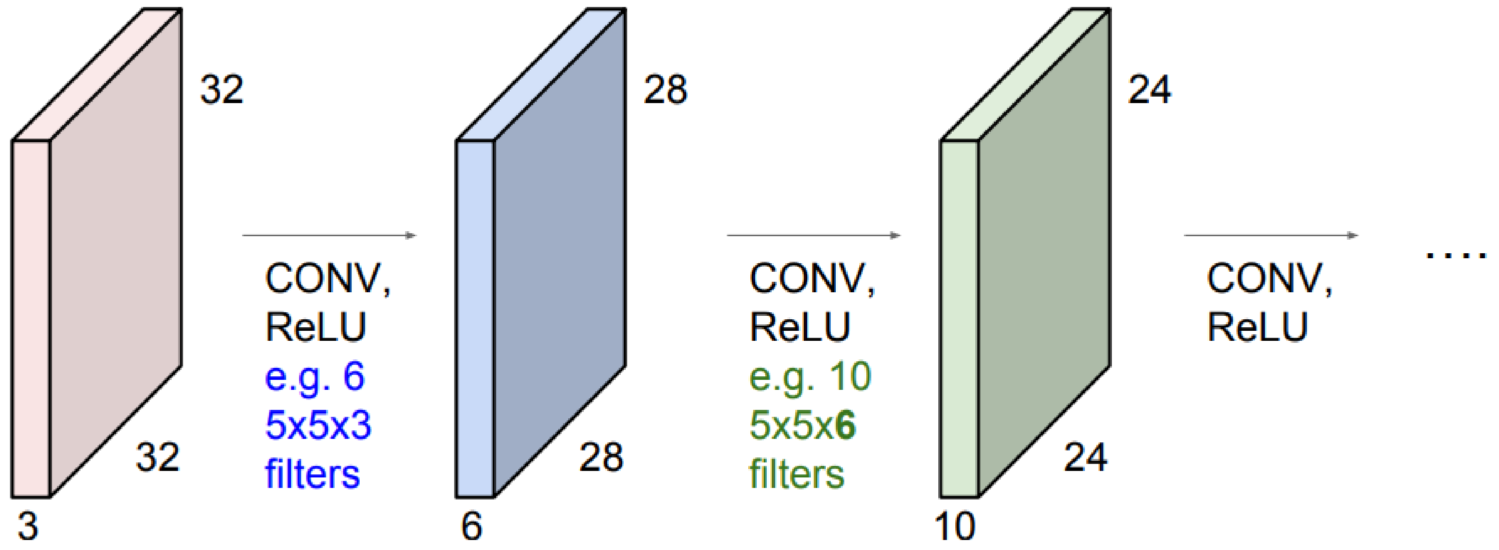


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

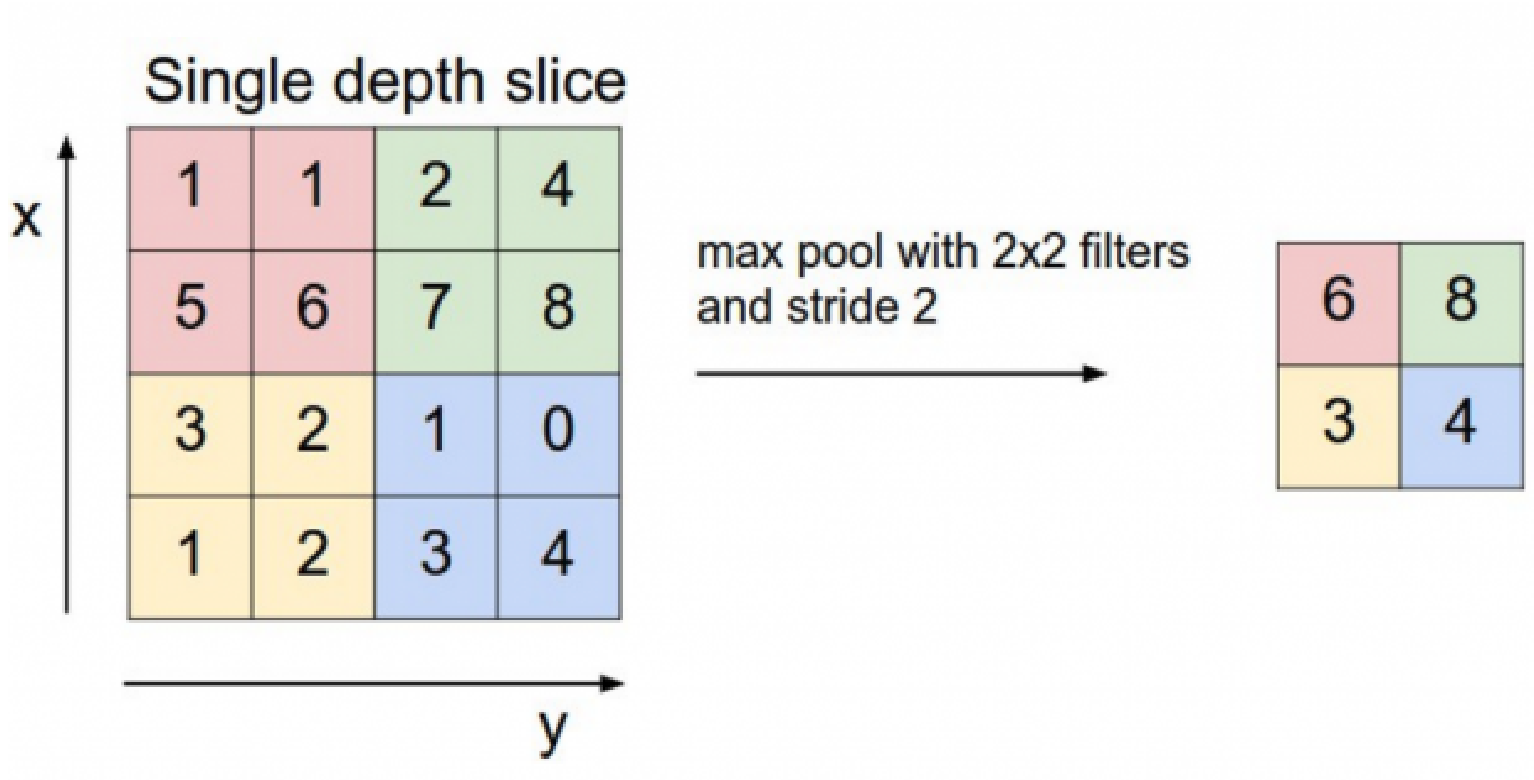
- The stride determines how the filter is applied

CNN

Convolutions alternate with Non Linear and Pooling layers



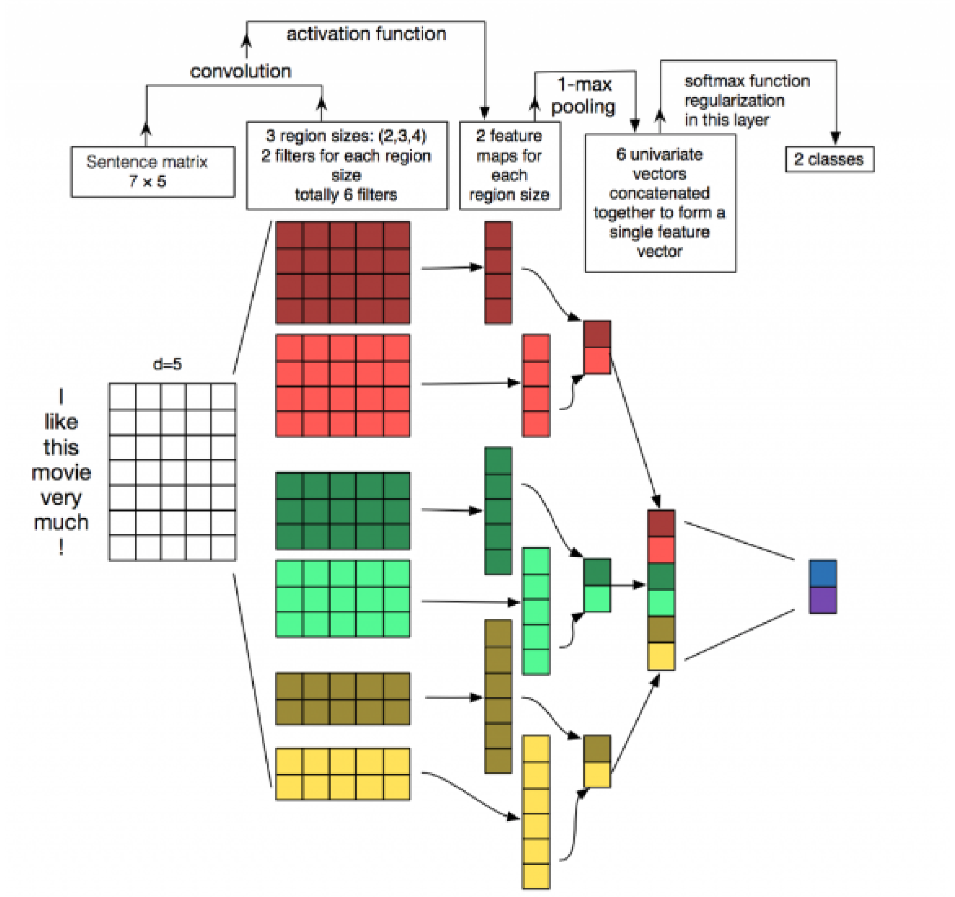
CNN



- Pooling is applied after the convolutional layers
- Pooling layers subsample their input.
- Usually applies a max operation to the result of each filter.
- Can be applied to the whole matrix or over a window.
- The picture shows max pooling for a 2×2 window
- in NLP we typically apply pooling over the complete output, yielding just a

CNN

Applying CNN to Text


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Session 6: 03/10, 9-12

Neural MR- and Text-to-Text Generation

YANG Ruoxiao (Lisa)

Shashi Narayan, Shay B. Cohen and Mirella Lapata. Ranking Sentences for Extractive Summarization with Reinforcement Learning In *Proceedings of NAACL-HLT 2018*, page 1747-1759, New Orleans, USA, June 2018. Association for Computational Linguistics [http]

Single document summarization is the task of producing a shorter version of a document while preserving its principal information content. In this paper we conceptualize extractive summarization as a sentence ranking task and propose a novel training algorithm which globally optimizes the ROUGE evaluation metric through a reinforcement learning objective. We use our algorithm to train a neural summarization model on the CNN and DailyMail datasets and demonstrate experimentally that it outperforms state-of-the-art extractive and abstractive systems when evaluated automatically and by humans.

Encoding Text Structure

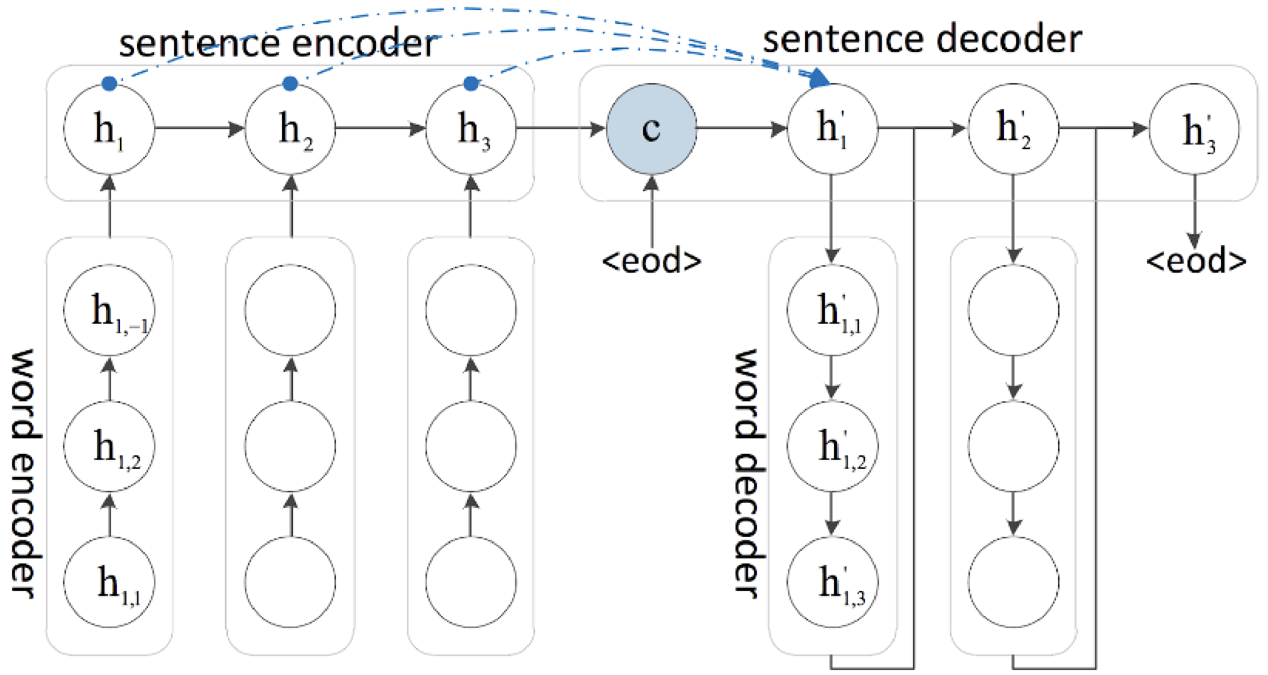
Encoding Text Structure

Abstractive and extractive summarisation

- Hierarchical encoders
  - with recurrent sentence encoders
  - with convolutional sentence encoders
- Ensemble encoders

Text as a Sequence of Sentences

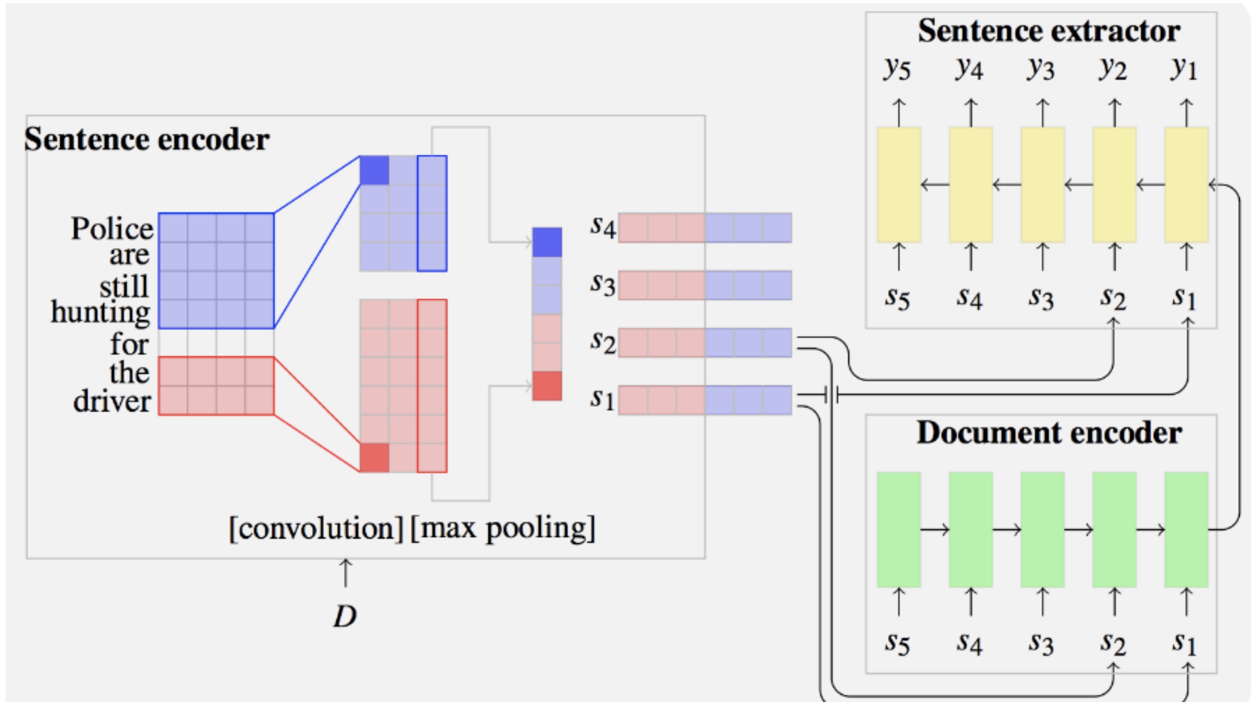
Using RNNs



Abstractive Document Summarization (Tan et al. ACL 2017)

Text as a Sequence of Sentences

Using CNNs


Sentence encoder
Police
are
still
hunting
for
the
driver
[convolution] [max pooling]
D
s_4
s_3
s_2
s_1
Sentence extractor
y_5 y_4 y_3 y_2 y_1
s_5 s_4 s_3 s_2 s_1
Document encoder
s_5 s_4 s_3 s_2 s_1

Session 6: 03/10, 9-12

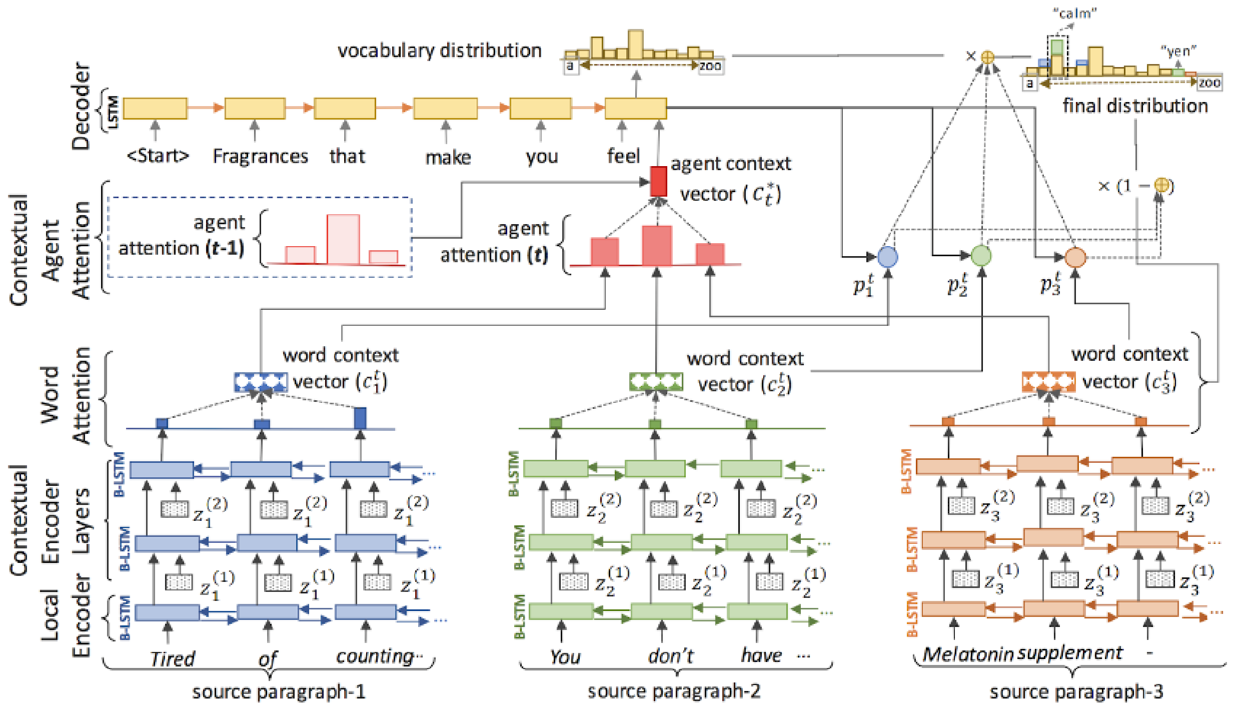
Neural MR- and Text-to-Text Generation

YANG Ruoxiao (Lisa)

Shashi Narayan, Shay B. Cohen and Mirella Lapata. Ranking Sentences for Extractive Summarization with Reinforcement Learning In *Proceedings of NAACL-HLT 2018*, page 1747-1759, New Orleans, USA, June 2018. Association for Computational Linguistics [http]

Single document summarization is the task of producing a shorter version of a document while preserving its principal information content. In this paper we conceptualize extractive summarization as a sentence ranking task and propose a novel training algorithm which globally optimizes the ROUGE evaluation metric through a reinforcement learning objective. We use our algorithm to train a neural summarization model on the CNN and DailyMail datasets and demonstrate experimentally that it outperforms state-of-the-art extractive and abstractive systems when evaluated automatically and by humans.

Text as a Sequence of Paragraphs



Abstractive Document Summarization (Celikyilmaz et al. NAACL 2018)

--- ## Encoders ###

RNNs ### CNN ### Graph Encoders]

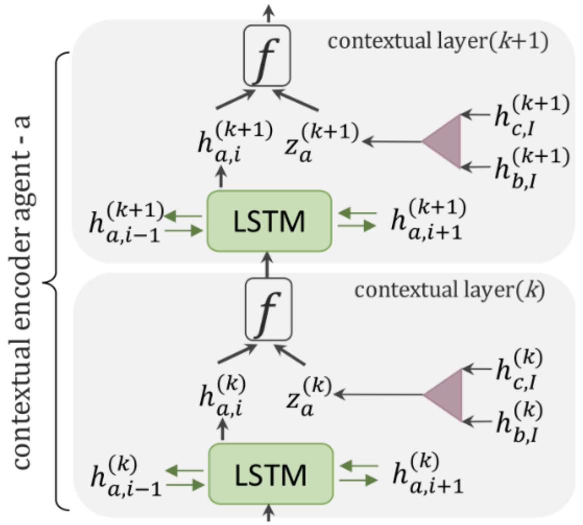
AMR

Multi-agent encoder message passing

$$\overrightarrow{h}_i^{(k+1)}, \overleftarrow{h}_i^{(k+1)} = \mathrm{bLSTM}(f(h_i^{(k)}, z^{(k)}), \overrightarrow{h}_{i-1}^{(k+1)}, \overleftarrow{h}_{i+1}^{(k+1)})$$

$$h_i^{(k+1)} = W_2[\overrightarrow{h}_i^{(k+1)}, \overleftarrow{h}_i^{(k+1)}]$$

$$z^{(k)} = \frac{1}{M-1} \sum_{m \neq a} h_{m,I}^{(k)}$$



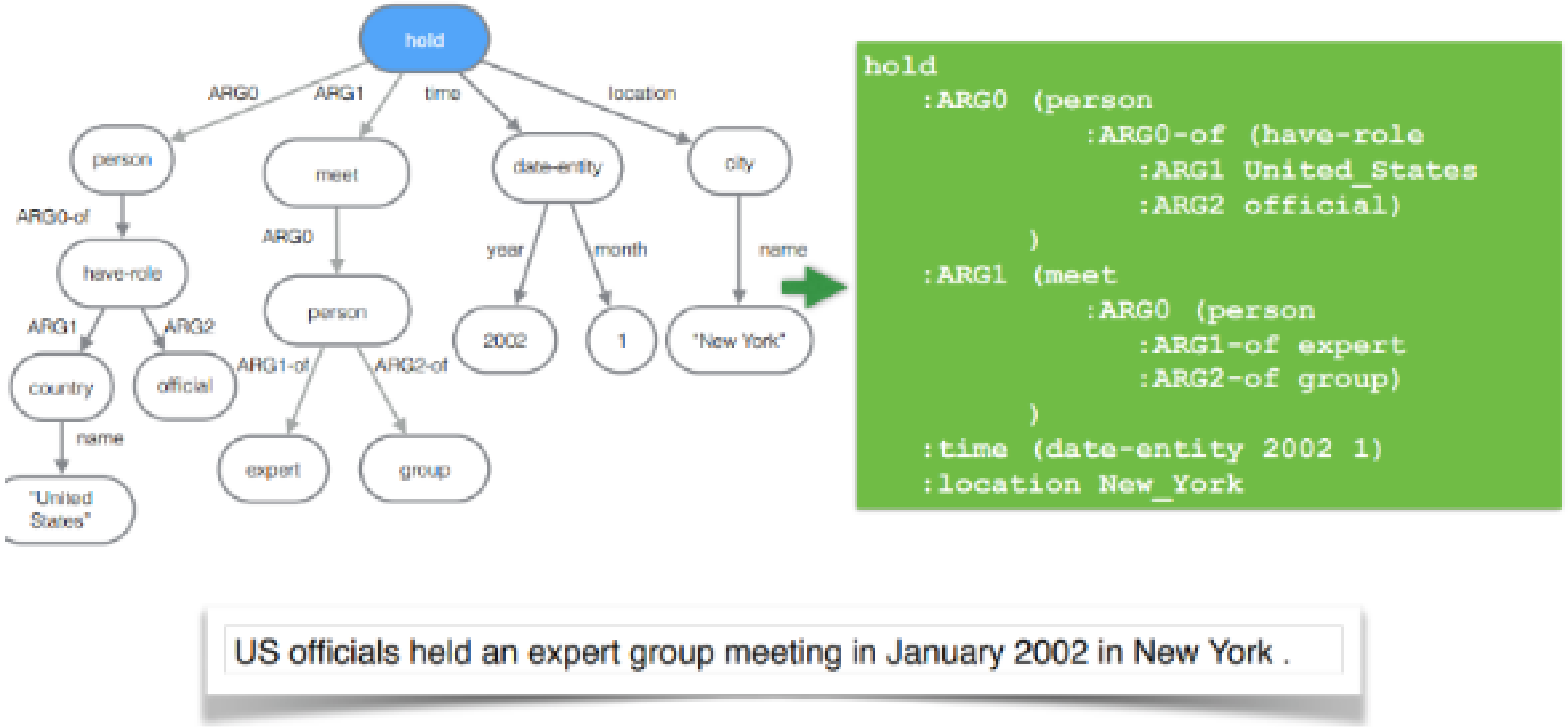
- Agents b and c transmit the last hidden state output of the current layer k as a message, which are passed through an average pool.
- The receiving agent a uses the new message z^k a as additional input to its next layer.

Graph Encoding

Graph Encoders

The Input to NLG can be a graph

E.g., Generation from AMR 2017 Challenge. The input to MR2T Generation is an Abstract Meaning Representation which can be viewed as a graph



```
hold
   :ARG0 (person
            :ARG0-of (have-role
               :ARG1 United_States
               :ARG2 official)
         )
   :ARG1 (meet
            :ARG0 (person
               :ARG1-of expert
               :ARG2-of group)
         )
   :time (date-entity 2002 1)
   :location New_York
```

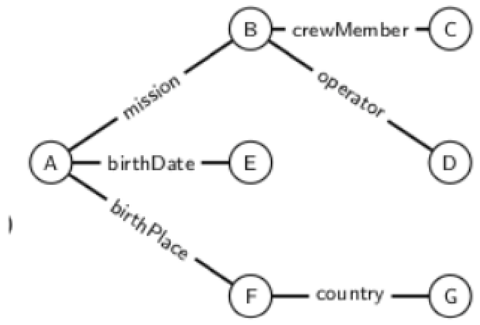
US officials held an expert group meeting in January 2002 in New York .

Graph Encoders

WebNLG Challenge 2017

- The input to D2T Generation is a set of RDF triples which can be viewed as a graph

D2T Generation (Data = RDF)



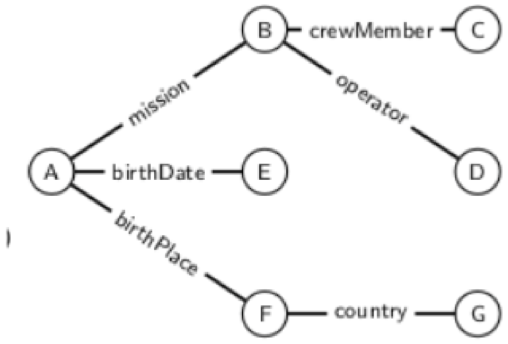
LINEARISATION

Alan_Bean **mission** Apollo_12 Apollo_12 **crewMember** Peter_Conrad Apollo_12 **operator** Nasa Alan_Bean birthDate 1932-03-15 Alan_Bean birthPlace Wheeler_Texas Wheeler_Texas country USA

Graph Structure as a Sequence

Early approaches to MR- or Data-to-Text generation encode the input structure as a sequence.

D2T Generation (Data = RDF)



LINEARISATION

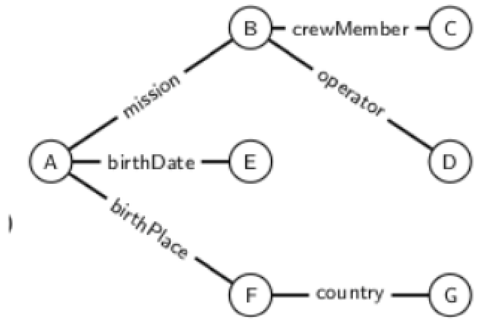
Alan_Bean **mission** Apollo_12 Apollo_12 **crewMember** Peter_Conrad Apollo_12 **operator** Nasa Alan_Bean birthDate 1932-03-15 Alan_Bean birthPlace Wheeler_Texas Wheeler_Texas country USA

Graph Encoders

Problems with Graph Linearization

- Local dependencies available in the input turned into long-range dependencies
- RNNs often have trouble modeling long-range dependencies

D2T Generation (Data = RDF)



LINEARISATION

Alan_Bean **mission** Apollo_12 Apollo_12 **crewMember** Peter_Conrad Apollo_12 **operator** Nasa Alan_Bean birthDate 1932-03-15 Alan_Bean birthPlace Wheeler_Texas Wheeler_Texas country USA

Session 5: 01/10, 10:15-12:15

Neural Data- and MR-to-Text Generation

LY Sophea

Bayu Distiawan, Jianzhong Qi, Rui Zhang, and Wei Wang. Gtr-lstm: A triple encoder for sentence generation from rdf data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1627--1637, 2018. [http]

A knowledge base is a large repository of facts that are mainly represented as RDF triples, each of which consists of a subject, a predicate (relationship), and an object. The RDF triple representation offers a simple interface for applications to access the facts. However, this representation is not in a natural language form, which is difficult for humans to understand. We address this problem by proposing a system to translate a set of RDF triples into natural sentences based on an encoder-decoder framework. To preserve as much information from RDF triples as possible, we propose a novel graph-based triple encoder. The proposed encoder encodes not only the elements of the triples but also the relationships both within a triple and between the triples. Experimental results show that the proposed encoder achieves a consistent improvement over the baseline models by up to 17.6%, 6.0%, and 16.4% in three common metrics BLEU, METEOR, and TER, respectively.

Session 6: 03/10, 9-12

Neural Data- and MR-to-Text Generation

NGO Minh Huong

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. A graph-to-sequence model for AMR-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616--1626, Melbourne, Australia, July 2018. Association for Computational Linguistics. [DOI | http]

The problem of AMR-to-text generation is to recover a text representing the same meaning as an input AMR graph. The current state-of-the-art method uses a sequence-to-sequence model, leveraging LSTM for encoding a linearized AMR structure. Although being able to model non-local semantic information, a sequence LSTM can lose information from the AMR graph structure, and thus facing challenges with large-graphs, which result in long sequences. We introduce a neural graph-to-sequence model, using a novel LSTM structure for directly encoding graph-level semantics. On a standard benchmark, our model shows superior results to existing methods in the literature.

Session 6: 03/10, 9-12

Neural Data- and MR-to-Text Generation

RAZET Guilherme

Leonardo FR Ribeiro, Claire Gardent, and Iryna Gurevych. Enhancing amr-to-text generation with dual graph representations. *arXiv preprint arXiv:1909.00352*, 2019. [.pdf]

Generating text from graph-based data, such as Abstract Meaning Representation (AMR), is a challenging task due to the inherent difficulty in how to properly encode the structure of a graph with labeled edges. To address this difficulty, we propose a novel graph-to-sequence model that encodes different but complementary perspectives of the structural information contained in the AMR graph. The model learns parallel top-down and bottom-up representations of nodes capturing contrasting views of the graph. We also investigate the use of different node message passing strategies, employing different state-of-the-art graph encoders to compute node representations based on incoming and outgoing perspectives. In our experiments, we demonstrate that the dual graph representation leads to improvements in AMR-to-text generation, achieving state-ofthe-art results on two AMR datasets.

Summary

Taking into account the structure of the input helps improve results

Graph encoding for data and MRs

Hierarchical encoding for text

Other Topics

Optimizing Key Properties of the Output

- Summarization: promoting the selection of key and correct information
- Simplification: promoting simplification
- Generation: promoting text/data alignment (semantic adequacy)

Interpretability

- Opening the black box
- Modular approaches

Other Topics

Learning

- Reinforcement learning
  Reward = BLEU, SARI, ROUGE, perplexity etc.
- Multitasking
  Exploiting other datasets/tasks

Architecture

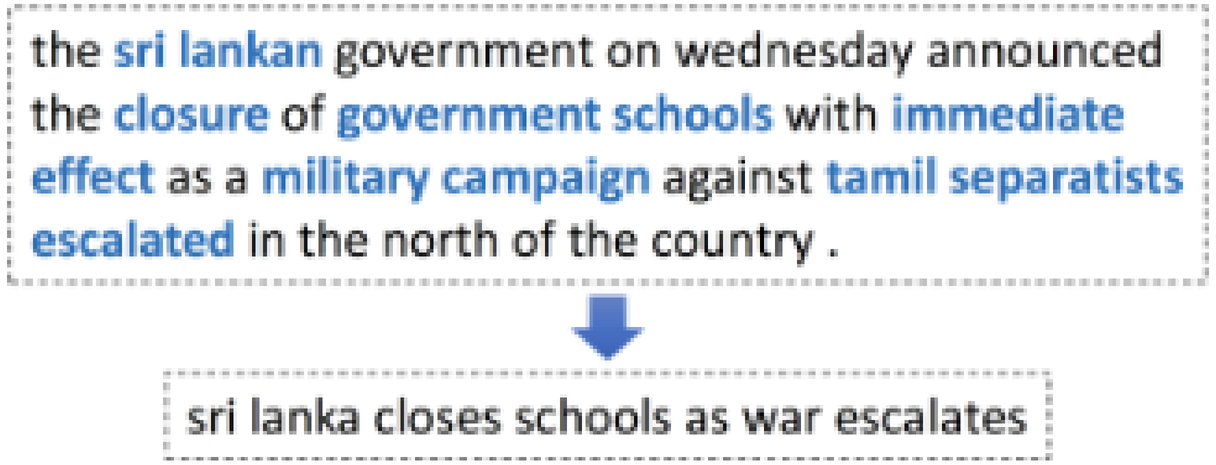
- Additional Information
  Parse trees, facts, PageRank information, etc.
- More Complex Networks
  Gates, Attention, etc.

Other Topics

Relevance

Abstractive Summarization

- A summary should express the input key information.

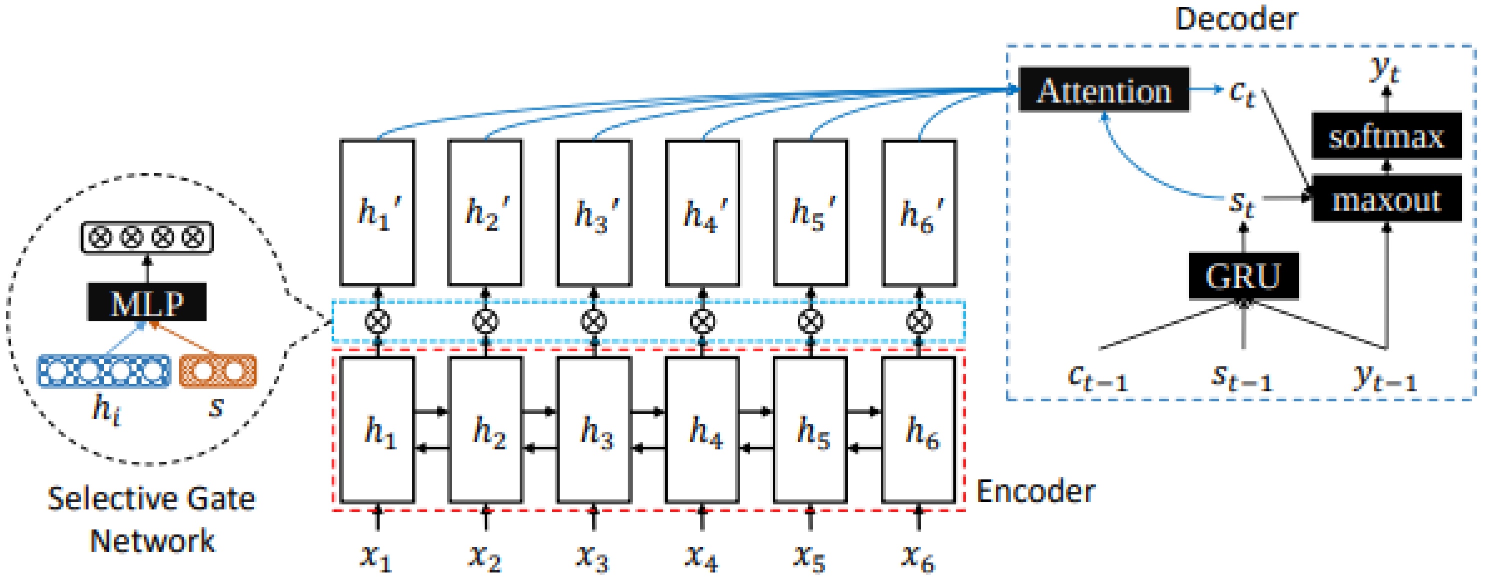

the **sri lankan** government on wednesday announced the **closure** of **government schools** with **immediate effect** as a **military campaign** against **tamil separatists escalated** in the north of the country .

sri lanka closes schools as war escalates

Other Topics

Relevance

Selective Encoding



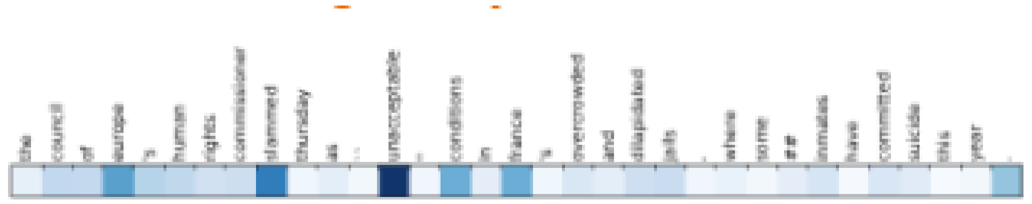
- Two LSTMs and a Selective Gate
- Intuition: use word (h_i) and sentence (s) representations to identify salient words
- Create a sentence representation tailored to highlight key information
- Decode using this tailored sentence representation

Other Topics

Relevance

Selective Encoding

- Key words have high activation values.



Input: The Council of Europe's human rights commissioner slammed thursday as "unacceptable" conditions in France's overcrowded and dilapidated jails, where some ## inmates have committed suicide this year.

Output Summary: Council of Europe slams French prisons conditions

Reference summary: Council of Europe again slams French prisons conditions

Session 6: 03/10. Selective Encoding

SRIVASTAVA Preprak

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1095--1104, Vancouver, Canada, July 2017. Association for Computational Linguistics. [DOI | http]

We propose a selective encoding model to extend the sequence-to-sequence framework for abstractive sentence summarization. It consists of a sentence encoder, a selective gate network, and an attention equipped decoder. The sentence encoder and decoder are built with recurrent neural networks. The selective gate network constructs a second level sentence representation by controlling the information flow from encoder to decoder. The second level representation is tailored for sentence summarization task, which leads to better performance. We evaluate our model on the English Gigaword, DUC 2004 and MSR abstractive sentence summarization datasets. The experimental results show that the proposed selective encoding model outperforms the state-of-the-art baseline models.

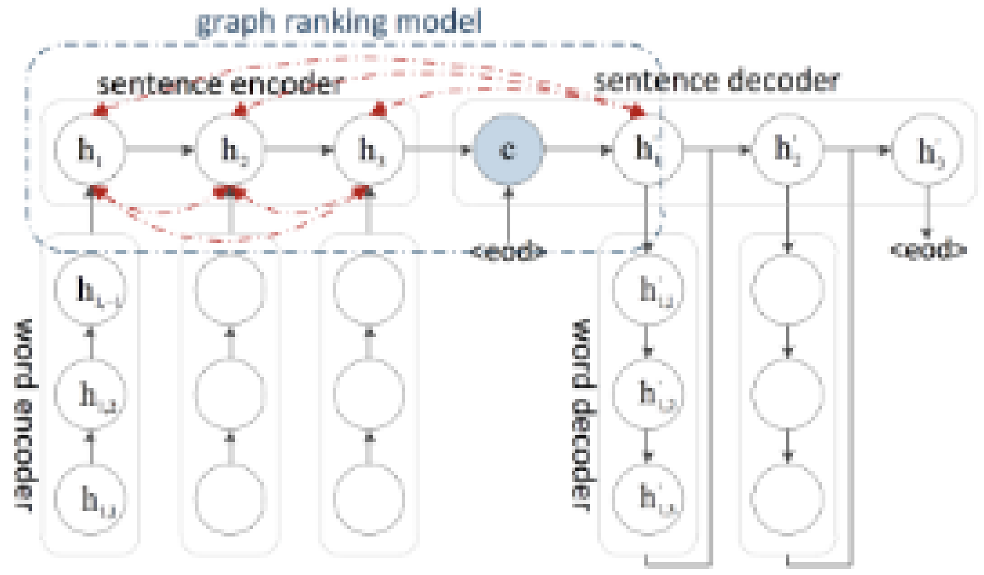
Other Topics

Relevance

Document Summarization with Modified Attention

Identifying Salient sentences with topic-sensitive PageRank

A sentence is important in a document if it is heavily linked with many important sentences.



(Tan, Wan and Xiao. ACL 2017)

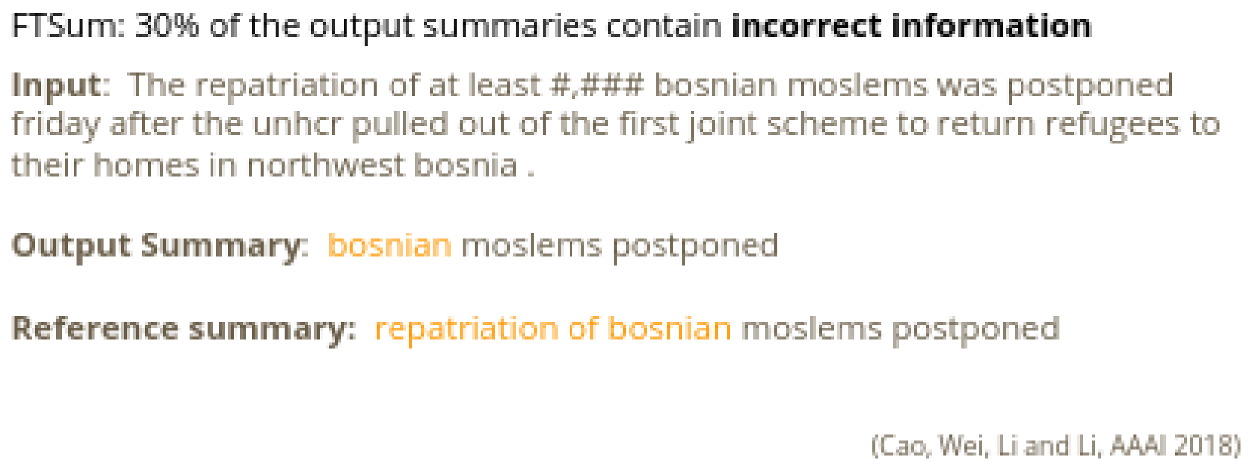
- Do not learn saliency.
- Instead use std sentence ranking algorithm (PageRank).
- Modify attention with weight from graph ranking model

Other Topic

Accuracy

Ensuring Faithfulness

Summarization approaches often generate summaries which contain incorrect information.



FTSum: 30% of the output summaries contain **incorrect information**

Input: The repatriation of at least #,### bosnian moslems was postponed friday after the unhcr pulled out of the first joint scheme to return refugees to their homes in northwest bosnia .

Output Summary: bosnian moslems postponed

Reference summary: repatriation of bosnian moslems postponed

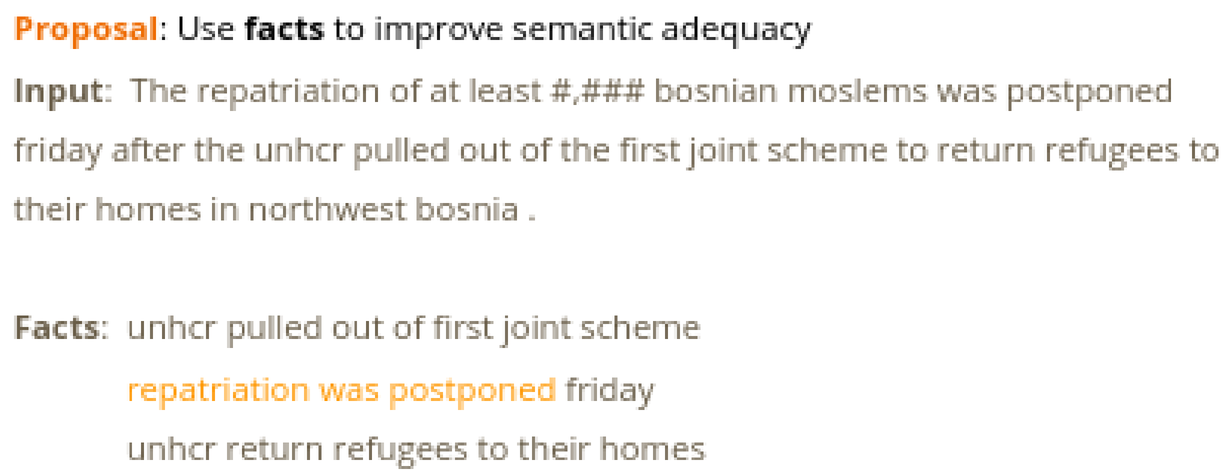
(Cao, Wei, Li and Li, AAAI 2018)

Other Topics

Accuracy

Encoding Facts

Intuition: Augment Input with Key Facts automatically extracted from the input document using IE (Information Extraction) tool.

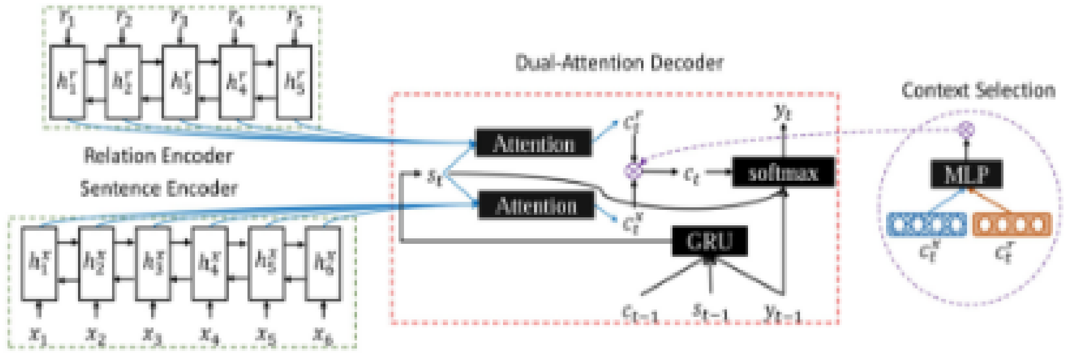


Proposal: Use **facts** to improve semantic adequacy

Input: The repatriation of at least #,### bosnian moslems was postponed friday after the unhcr pulled out of the first joint scheme to return refugees to their homes in northwest bosnia .

Facts: unhcr pulled out of first joint scheme
repatriation was postponed friday
unhcr return refugees to their homes

Faithful to the Original Model

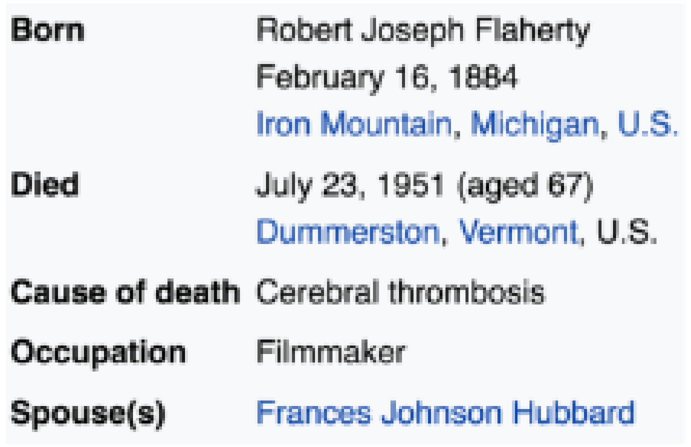


- Two encoders (document, extracted facts) each with separate attention
- The gate values are computed based on linear combinations of the two context vectors passed through a sigmoid function
- The final context vector (c_t)is built by taking the weighted sum of the two context vectors

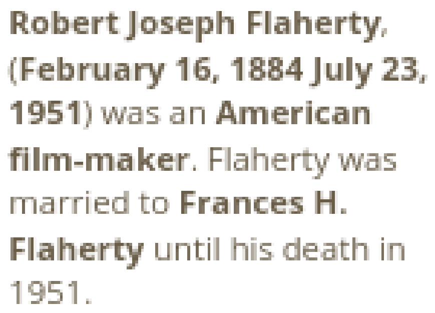
Other Topics

Selection

Generation from Loosely Aligned Data



Born	Robert Joseph Flaherty February 16, 1884 Iron Mountain, Michigan, U.S.
Died	July 23, 1951 (aged 67) Dummerston, Vermont, U.S.
Cause of death	Cerebral thrombosis
Occupation	Filmmaker
Spouse(s)	Frances Johnson Hubbard



Robert Joseph Flaherty, (**February 16, 1884 July 23, 1951**) was an **American film-maker**. Flaherty was married to **Frances H. Flaherty** until his death in 1951.

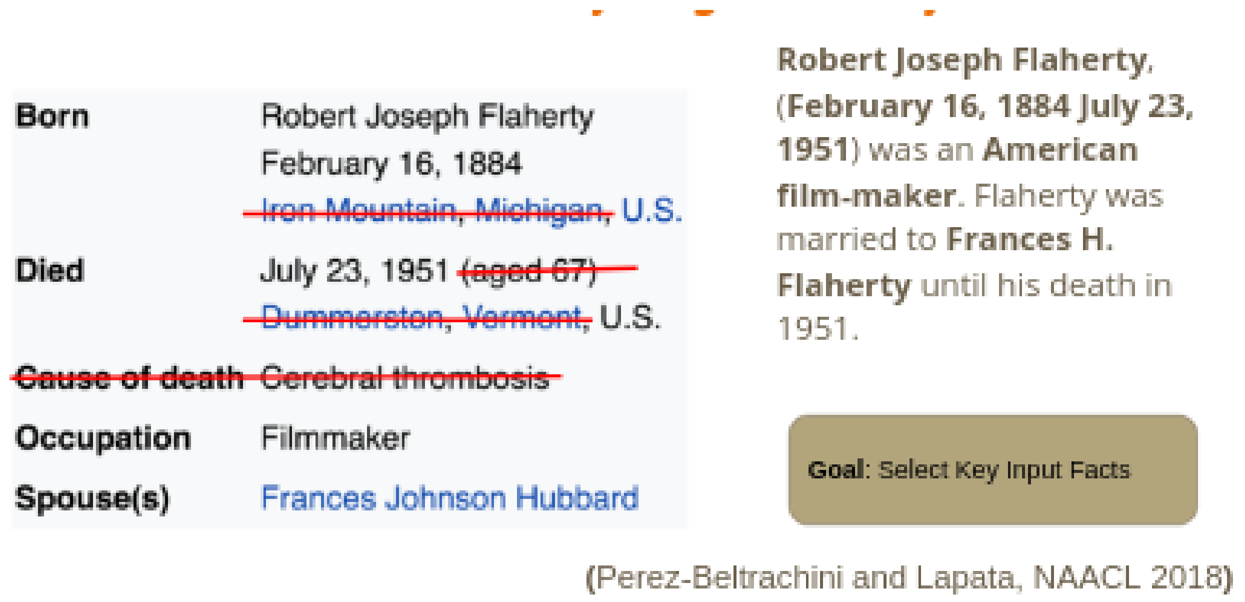
(Perez-Beltrachini and Lapata, NAACL 2018)

Other Topics

Selection

Generation from Loosely Aligned Data

Not all text information is present in the input data.



Born	Robert Joseph Flaherty February 16, 1884 ~~Iron Mountain, Michigan,~~ U.S.
Died	July 23, 1951 ~~(aged 67)~~ ~~Dummerston, Vermont,~~ U.S.
~~Cause of death~~	~~Cerebral thrombosis~~
Occupation	Filmmaker
Spouse(s)	Frances Johnson Hubbard

Robert Joseph Flaherty, (**February 16, 1884 July 23, 1951**) was an **American film-maker**. Flaherty was married to **Frances H. Flaherty** until his death in 1951.

Goal: Select Key Input Facts

(Perez-Beltrachini and Lapata, NAACL 2018)

Other Topics

Selection

Multi-Task Objective

Word prediction (generation) objective: $$\mathcal{L}_{wNLL} = -\sum_{t=1}^{|Y|} \log P(y_t | y_{1:t-1}, X)$$

Content selection objective: $$\mathcal{L}_{aln} = -\sum_{t=1}^{|Y|} \log P(a_t | y_{1:t-1}, X)$$

Multi-Task Learning: $$\mathcal{L}_{MTL} = \lambda \mathcal{L}_{wNLL} + (1-\lambda)\mathcal{L}_{aln}$$

Generation Objective: Maximise the probability of the output text.

Content Selection Objective: Maximise the alignment with the input. For each predicted word, a_t indicates whether the word is aligned with some input or not

Other Topics

Interpretability

Black box end-to-end models vs. Modular Approaches

Session 5: 01/10, 10:15 - 12:15

MARQUER Esteban

Amit Moryossef, Yoav Goldberg, and Ido Dagan. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of NAACL-HLT 2019*, pages 2267--2277, Minneapolis, Minnesota, US, June 2019. Association for Computational Linguistics. [http]

Data-to-text generation can be conceptually divided into two parts: ordering and structuring the information (planning), and generating fluent language describing the information (realization). Modern neural generation systems conflate these two steps into a single end-to-end differentiable system. We propose to split the generation process into a symbolic text-planning stage that is faithful to the input, followed by a neural generation stage that focuses only on realization. For training a plan-to-text generator, we present a method for matching reference texts to their corresponding text plans. For inference time, we describe a method for selecting high-quality text plans for new inputs. We implement and evaluate our approach on the WebNLG benchmark. Our results demonstrate that decoupling text planning from neural realization indeed improves the system's reliability and adequacy while maintaining fluent output. We observe improvements both in BLEU scores and in manual evaluations. Another benefit of our approach is the ability to output diverse realizations of the same input, paving the way to explicit control over the generated text structure.

Other Topics

Interpretability

Black box end-to-end models vs. Modular Approaches

Session 5: 01/10, 10:15 - 12:15

NADAR Fatima

ThiagoCastro Ferreira, Diego Moussallem, Akos Kadar, Sander Wubben, and Emiel Krahmer. Neuralreg: An end-to-end approach to referring expression generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, page 1959â€“1969, Melbourne, Australia, 2018. [http]

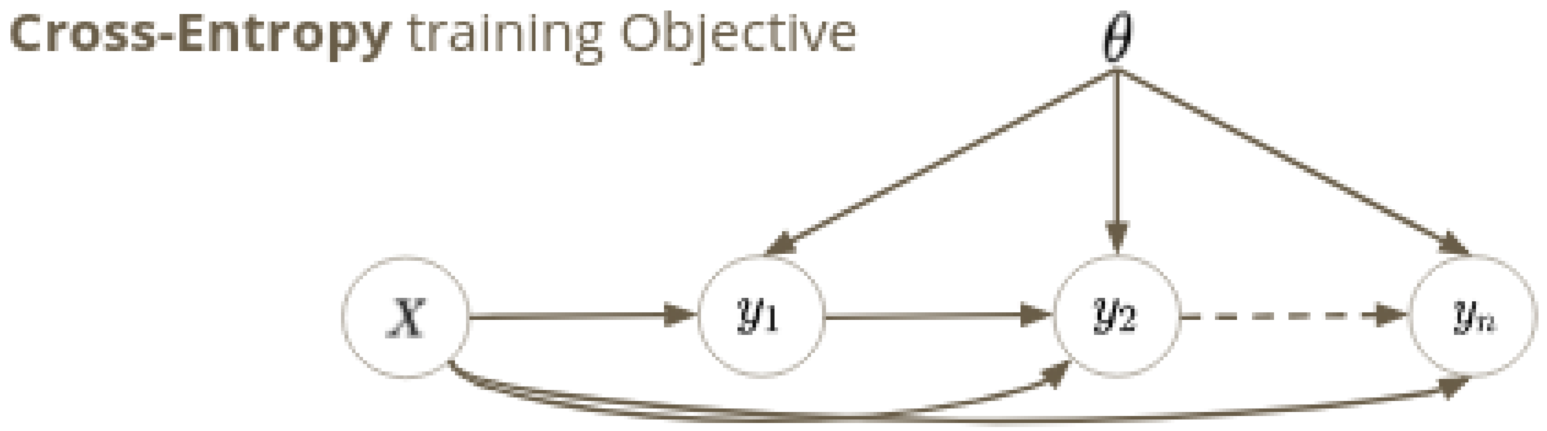
> Traditionally, Referring Expression Generation (REG) models first decide on the form and then on the content of references to discourse entities in text, typically relying on features such as salience and grammatical function. In this paper, we present a new approach (NeuralREG), relying on deep neural networks, which makes decisions about form and content in one go without explicit feature extraction. Using a delexicalized version of the WebNLG corpus, we show that the neural model substantially improves over two strong baselines. Data and models are publicly available1

Other Topics

Loss Function

Cross Entropy vs. Task-Specific Metric

$$L(\theta) = -\sum_{i=1}^{n} \log p(y_i | y_1, \ldots, y_{i-1}, X; \theta)$$



Cross entropy: maximize the likelihood of the training data

Other Topics

Loss Function

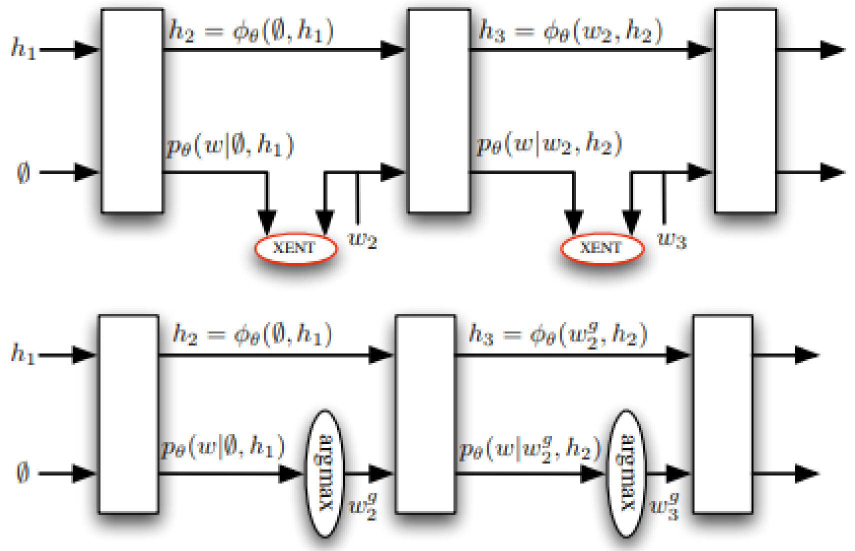
Two Problems with Cross Entropy

- It maximises the likelihood of the next correct word rather than the task-specific evaluation metrics (e.g., ROUGE or BLEU)
- Exposure Bias

Other Topics

Loss Function

The Exposure Bias Problem

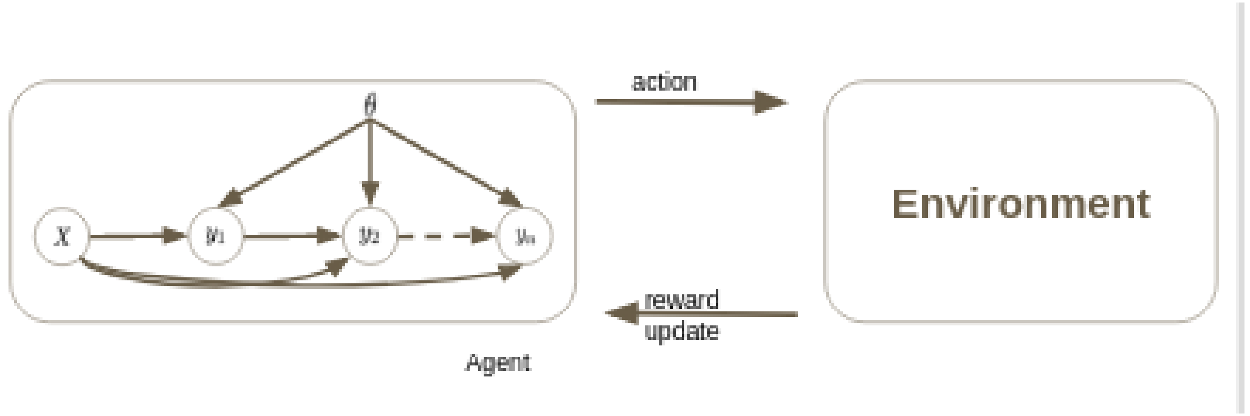

h_1
$\emptyset$
$h_2 = \phi_\theta(\emptyset, h_1)$
$p_\theta(w|\emptyset, h_1)$
XENT
w_2
$h_3 = \phi_\theta(w_2, h_2)$
$p_\theta(w|w_2, h_2)$
XENT
w_3
h_1
$\emptyset$
$h_2 = \phi_\theta(\emptyset, h_1)$
$p_\theta(w|\emptyset, h_1)$
argmax
w_2^g
$h_3 = \phi_\theta(w_2^g, h_2)$
$p_\theta(w|w_2^g, h_2)$
argmax
w_3^g

At **training time** , predicts the next word, given the previous reference word (w_2).
At **test time**, predicts the next word based on previous model prediction (w_2^g).

Other Topics

Loss Function

Text Production as Reinforcement Learning



- Action: generate sentence
- Environment: compares the generated sentence against the reference and gives back a reward (e.g., ROUGE score for summarization)
- Agent: updates parameters based on reward

REINFORCE algorithm

(Williams, 1992)

- Goal: find the parameters of the agent that maximize the expected reward
- Loss: the negative expected reward

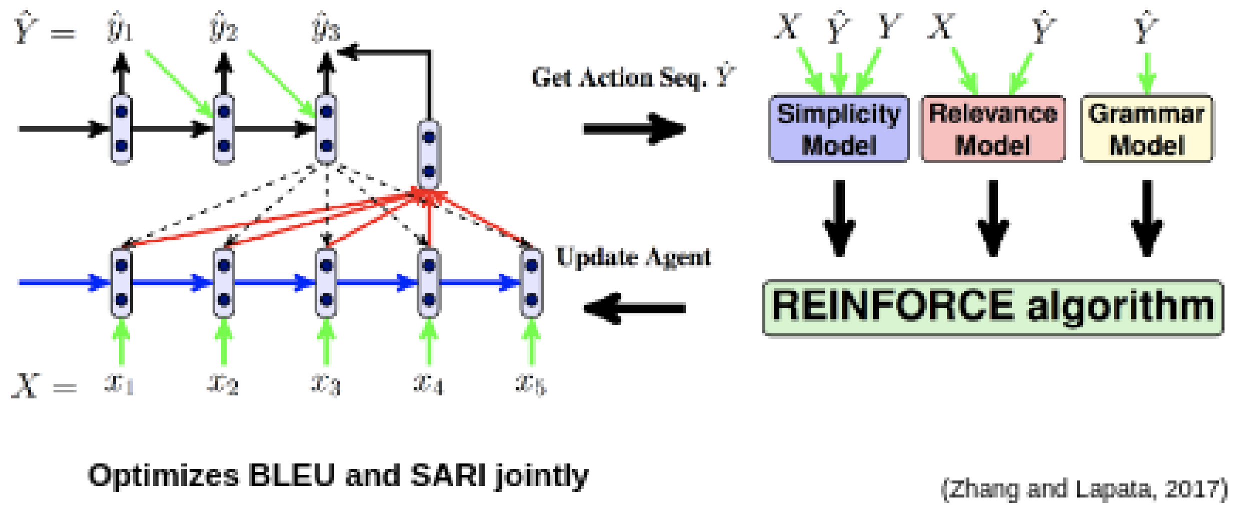
$$L(\theta) = -\mathbb{E}_{\hat{y}\sim\, p_\theta}[r(\hat{y})]$$

$$= -\sum_{\hat{y}\sim\, p_\theta} r(\hat{y})p(\hat{y}|\theta)$$

Other Topics

RL

Simplification using RL



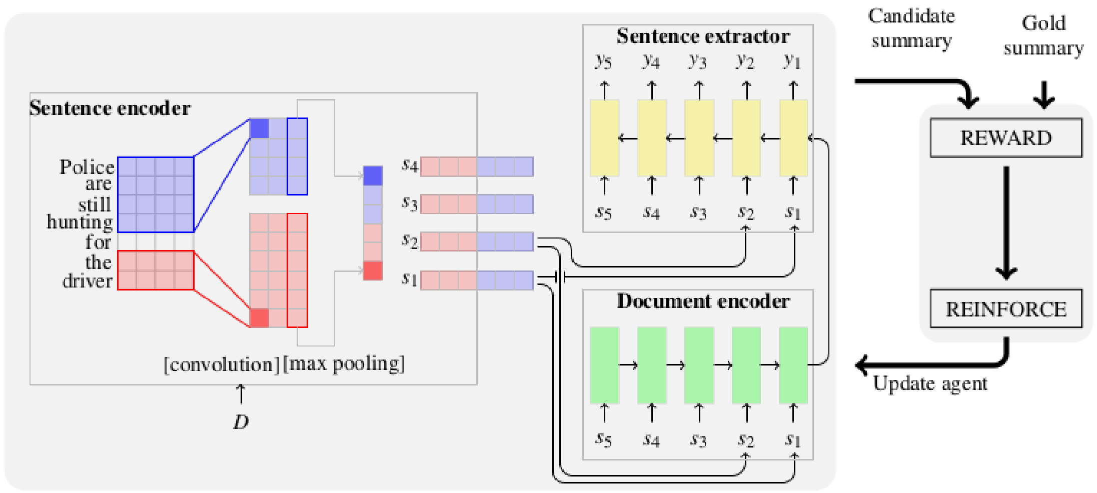
The reward combine different evaluation metrics.

- Grammar: a language model is used to measure perplexity
- Simplicity: SARI is used to measure the degree of simplification
- Relevance: BLEU is used to measure the similarity to the reference

Other Topics

RL

Extractive Summarization using RL


Sentence encoder
Police are still hunting for the driver
[convolution][max pooling]
D
s_4 s_3 s_2 s_1
Sentence extractor
y_5 y_4 y_3 y_2 y_1
s_5 s_4 s_3 s_2 s_1
Document encoder
s_5 s_4 s_3 s_2 s_1
Candidate summary
Gold summary
REWARD
REINFORCE
Update agent

- Sentence Encoder (CNN): creates a continuous representation for each sentence
- Document Encoder (LSTM): creates a **document representation** from the sequence of sentence representations.
- Sentence Extractor (LSTM): assigns to each sentence a 0-1 score conditioned on the **document representation** (obtained from the document encoder) and the previously labeled sentences.
- the REWARD generator compares the candidate summary against the gold summary to give a reward which is used by the REINFORCE algorithm to

Other Topics

RL

Abstractive Summarization using RL

- (Paulus 2017). *A Deep Reinforced Model for Abstractive Summarization*. Reward = ROUGE
- (Pasunuru and Bansal NAACL 2018). *Multi-Reward Reinforced Summarization with Saliency and Entailment*
- (Celikyilmaz et al NAACL 2018). *Deep Communicating Agents for Abstractive Summarization*

Questions ?