Vision, Language and Action: from Captioning to Embodied AI

Lorenzo Baraldi, Marcella Cornia, Massimiliano Corsini

{name.surname}@unimore.it

University of Modena and Reggio Emilia, Italy
Individually successful stories, that recently demand for a combination
INTRODUCTION

Image captioning
Multi-modal retrieval
Visual Question Answering

Language models
Dialogue agents
Question answering

Robots, agents
Manipulation

Simulated environments
Vision for Robotics

Vision and Language Navigation
Embodied Visual Question Answering
Embodied Recognition

Image understanding
Object detection
Video classification
3D perception

Vision

Language

Action

Intersecting fields of study:

Image understanding
Object detection
Video classification
3D perception

Simulated environments
Vision for Robotics

Vision and Language Navigation
Embodied Visual Question Answering
Embodied Recognition


1. Introduction

2. Describing images
   1. Structure of a captioning system
   2. Optimizing for metrics: Scheduled Sampling, Reinforcement Learning, Self-critical Sequence Training
   3. Visual encoding for sequences and sets, attentive mechanisms, the visual sentinel
   4. Convolutional and Transformer-based language models
   5. SoTA approaches and new challenges

3. Captioning applications: controllable captioning

4. Cross-modal retrieval

5. Embodied AI and Vision-and-Language navigation
   1. Datasets and Simulators overview
   2. Metrics and evaluation challenges
   3. State-of-the-art algorithms
   4. Open challenges
Part II

Image Captioning

Lorenzo Baraldi
{name.surname}@unimore.it

University of Modena and Reggio Emilia, Italy
Goal: describe a visual input in natural language.

Base technical idea: Combine visual feature extractors with language models

Base technical idea: combine visual feature extractors with language models.
Base technical idea: combine visual feature extractors with language models.

Many possibilities

- **Language model**:
  - RNN and variants (LSTM)
  - 1-d CNN
  - Transformer-based (recently)
Base technical idea: combine visual feature extractors with language models.

Many possibilities

- **Language model:**
  - RNN and variants (LSTM)
  - 1-d CNN
  - Transformer-based (recently)
- **Conditioning on the visual input:**
  - Single feature (e.g. pooling)
  - Sequence of features (e.g. video captioning)
  - Set of features (models based on attention)
Base technical idea: combine visual feature extractors with language models.

Many possibilities

- **Language model**:
  - RNN and variants (LSTM)
  - 1-d CNN
  - Transformer-based (recently)
- **Conditioning on the visual input**:  
  - Single feature (e.g. pooling)
  - Sequence of features (e.g. video captioning)
  - Set of features (models based on attention)

In this tutorial: *try to build a unifying perspective.*
Language model

- Prediction process is always **sequential**, i.e. we model the probability of outputting a word given previous words in the sentence.

- The probability distribution for $w_t$ is conditioned on $w_{t-1}$, $w_{t-2}$, ... $w_0$
Language model

- Prediction process is always **sequential**, i.e., we model the probability of outputting a word given previous words in the sentence.

- The probability distribution for $w_t$ is conditioned on $w_{t-1}$, $w_{t-2}$, … $w_0$

- A function $f$ models the computational graph for predicting the word at each step (the “step function”).
  - Any of \{RNN, CNN, Transformer, …\}
Most traditional Language model choice: **RNNs**

Any RNN cell can naturally model a step function. Plus, it holds and pass a **state** to the next iteration.

\[ h_t = f_W (h_{t-1}, x_t) \]

Many variants, same API: *all suitable for captioning*

\[
h_t = f_W(h_{t-1}, x_t)
\]

\[
i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1})
\]

\[
f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1})
\]

\[
o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1})
\]

\[
c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1})
\]

\[
m_t = o_t \odot c_t
\]

\[
p_{t+1} = \text{Softmax}(m_t)
\]
ANATOMY OF A CAPTIONING SYSTEM

LSTM

Visual

\( w_{t-2} \) \rightarrow \text{LSTM} \rightarrow \text{LSTM} \rightarrow \text{LSTM} \rightarrow \text{Visual}

Probability distribution for \( w_{t-1} \)

Probability distribution for \( w_{t} \)

Probability distribution for \( w_{t+1} \)
ANATOMY OF A CAPTIONING SYSTEM

- Visual
- Probability distribution for $w_{t-1}$
- Probability distribution for $w_t$
- Probability distribution for $w_{t+1}$
At training time

- *Train the model to predict a word given the previous ground-truth words.*

\[ w_t \text{: ground-truth words} \]
At training time

- Train the model to predict a word given the previous ground-truth words.

\( w_t \): ground-truth words
At training time

- *Train the model to predict a word given the previous ground-truth words.*

\( w_t \): ground-truth words
At training time: condition on the image and train to predict the next word given the previous (GT) words

\[
\max_w \sum_{t=1}^{T} \log \Pr(y_t | y_{t-1}, y_{t-2}, \ldots, y_0)
\]

Sum (average) losses over \( t \), then backward. This is called “teacher forcing”.
At training time

- Train the model to predict a word given the previous ground-truth words.

If the step function does not depend on its own output at previous timesteps:
- We can parallelize over the t axis.
  - Training time reduction
  - E.g. Conv1D, Transformer

\( w_t \) : ground-truth words
At training time

• Train the model to predict a word given the previous ground-truth words.

If the step function does not depend on its own output at previous timesteps:

• We can parallelize over the t axis.
  • \(\rightarrow\) Training time reduction
  • E.g. Conv1D, Transformer

\(w_t:\) ground-truth words
At training time

- Train the model to predict a word given the previous ground-truth words.

If the step function does not depend on its own output at previous timesteps:

- We can parallelize over the t axis.
  - → Training time reduction
  - E.g. Conv1D, Transformer

\( w_t \): ground-truth words
BACKPROPAGATION THROUGH TIME

![Diagram showing backpropagation through time](image_url)
At prediction time (sampling)

• We sample one word from the previous output, and use this as an input.

\[ w_t : \text{sampled words} \]
At prediction time (sampling)

- We sample one word from the previous output, and use this as an input.

\( w_t \): sampled words
At prediction time (sampling)

- We sample one word from the previous output, and use this as an input.
- Possible strategies:
  - Always sample the most probable word
  - Build a tree of possible choices, then select the chain of predictions with maximum probability (beam search)

\( w_t \): sampled words
Show and Tell

Combines RNN (LSTM) and global image feature vector. Beam search at decoding stage.

A baseline for following works: 85.5 CIDEr on COCO development set.

```python
def step(self, t, state, prev_output, images, seq, *args, mode='teacher_forcing'):
    assert (mode in ['teacher_forcing', 'feedback'])
    device = images.device
    b_s = images.size(0)
    if t == 0:
        xt = self.fc_image(images)
    else:
        if mode == 'teacher_forcing':
            it = seq[:, t - 1]
        elif mode == 'feedback':
            it = prev_output

        xt = self.embed(it)

    out, state = self.lstm_cell(xt, state)
    out = F.log_softmax(self.out_fc(out), dim=-1)
    return out, state
```
def step(self, t, state, prev_output, images, seq, *args, mode='teacher_forcing'):
    assert (mode in ['teacher_forcing', 'feedback'])
    device = images.device
    b_s = images.size(0)

    if t == 0:
        xt = self.fc_image(images)
    else:
        if mode == 'teacher_forcing':
            it = seq[:, t - 1]
        elif mode == 'feedback':
            it = prev_output

        xt = self.embed(it)

    out, state = self.lstm_cell(xt, state)
    out = F.log_softmax(self.out_fc(out), dim=-1)

    return out, state
def step(self, t, state, prev_output, images, seq, *args, mode='teacher_forcing'):
    assert (mode in ['teacher_forcing', 'feedback'])
    device = images.device
    b_s = images.size(0)
    if t == 0:
        xt = self.fc_image(images)
    else:
        if mode == 'teacher_forcing':
            it = seq[:, t - 1]
        elif mode == 'feedback':
            it = prev_output

        xt = self.embed(it)

    out, state = self.lstm_cell(xt, state)
    out = F.log_softmax(self.out_fc(out), dim=-1)
    return out, state

Training time: input is previous GT word
def step(self, t, state, prev_output, images, seq, *args, mode='teacher_forcing'):
    assert (mode in ['teacher_forcing', 'feedback'])
    device = images.device
    b_s = images.size(0)
    if t == 0:
        xt = self.fc_image(images)
    else:
        if mode == 'teacher_forcing':
            it = seq[:, t - 1]
        elif mode == 'feedback':
            it = prev_output
        xt = self.embed(it)

    out, state = self.lstm_cell(xt, state)
    out = F.log_softmax(self.out_fc(out), dim=-1)
    return out, state

Test time: input is previously sampled word
def step(self, t, state, prev_output, images, seq, *args, mode='teacher_forcing'):
    assert (mode in ['teacher_forcing', 'feedback'])
    device = images.device
    b_s = images.size(0)
    if t == 0:
        xt = self.fc_image(images)
    else:
        if mode == 'teacher_forcing':
            it = seq[:, t - 1]
        elif mode == 'feedback':
            it = prev_output
        xt = self.embed(it)

    out, state = self.lstm_cell(xt, state)
    out = F.log_softmax(self.out_fc(out), dim=-1)

    return out, state
A library for Visual-Semantic tasks built on top of PyTorch

Data-loading

- Off-the-shelf datasets (COCO, Flickr30k, Flickr8k, and more)
- Support for custom datasets with custom fields
- Customizable pipelines for text and image processing

Models

- Boostraps the development of captioning and visual-semantic retrieval models
- Easy XE and RL pipelines
- Pre-trained state-of-the-art models with implementation and weights

Evaluation

- All standard metrics (BLEU@k, CIDEr, ROUGE, ...)

https://github.com/aimagelab/speaksee
Training

• Input: ground truth token at every time step

Inference/Evaluation

• Input: previous predicted token

→ Errors tend to accumulate over long range sequences

3. Huszár, F. How (not) to train your generative model: Scheduled sampling, likelihood, adversary?. arXiv 2015
Training

- Input: ground truth token at every time step

Inference/Evaluation

- Input: previous predicted token

→ Errors tend to accumulate over long range sequences

Scheduled sampling: alternate between using GT words and sampled ones during training.


3. Huszár, F. How (not) to train your generative model: Scheduled sampling, likelihood, adversary?. *arXiv 2015*
Training
• Input: ground truth token at every time step

Inference/Evaluation
• Input: previous predicted token

Errors tend to accumulate over long range sequences

Scheduled sampling: alternate between using GT words and sampled ones during training.

But... “despite its impressive empirical performance, it leads the objective function underlying is improper” (Huszar et al.)

3. Huszár, F. How (not) to train your generative model: Scheduled sampling, likelihood, adversary?. arXiv 2015
Idea: optimize (finetune) a captioning metric via Reinforcement Learning

- Language model is the policy
- Sampled words are actions
- Compute a (non-differentiable, often) captioning metric \textit{on the whole sequence} and use it as reward

Idea: optimize (finetune) a captioning metric via Reinforcement Learning

- Language model is the policy
- Sampled words are actions
- Compute a (non-differentiable, often) captioning metric on the whole sequence and use it as reward

Self-critical sequence training

- Generate a caption sampling from output distribution, compute the captioning metric wrt GT, use this as a reward for every t
- Apply REINFORCE with a baseline

\[
\frac{\partial L(\theta)}{\partial s_t} \approx (r(w^*) - b)(p_{\theta}(w_t|h_t) - 1_{w_t^*})
\]

As a baseline (b): reward of caption predicted with beam search (i.e., how the model would perform at test time)

Self-critical sequence training

- Generate a caption sampling from output distribution, compute the captioning metric wrt GT, use this as a reward for every t
- Apply REINFORCE with a baseline

\[ \frac{\partial L(\theta)}{\partial s_t} \approx (r(w^*) - b)(p_{\theta}(w_t|h_t) - 1_{w^*_t}) \]

As a baseline (b): reward of caption predicted with beam search (i.e., how the model would perform at test time)

*At present, the best way to get SOTA results.*

<table>
<thead>
<tr>
<th>Training Metric</th>
<th>Evaluation Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIDEr</td>
</tr>
<tr>
<td>XE</td>
<td>90.9</td>
</tr>
<tr>
<td>XE (beam)</td>
<td>94.0</td>
</tr>
<tr>
<td>MIXER-B</td>
<td>101.9</td>
</tr>
<tr>
<td>MIXER</td>
<td>104.9</td>
</tr>
<tr>
<td>SCST</td>
<td><strong>106.3</strong></td>
</tr>
</tbody>
</table>

Many possibilities to encode the visual input:

- **Single feature (e.g. pooling)**
- Sequence of features (e.g. video captioning)
- Set of features (models based on attention)
Many possibilities to encode the visual input:

- Single feature (e.g. pooling)
- **Sequence of features** (e.g. video captioning)
- Set of features (models based on attention)

**E.g. HBE,**

A video encoding network which can **adaptively modify its structure.**

The result is a **variable length and adaptive encoding of the video,** whose length and granularity depends on the input video itself.

Many possibilities to encode the visual input:

- Single feature (e.g. pooling)
- Sequence of features (e.g. video captioning)
- **Set of features (models based on attention)**

1. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *ICML 2015*. 

![Diagram showing image encoding process]

**WHAT ABOUT IMAGE ENCODING**
Attention

Provides a way to focus on part of an input set.

Given a query and pairs of keys and values,

- Compute similarities between queries and keys
- Normalizes similarities via softmax to obtain attention scores
- Multiplies values by the scores
Attention

Provides a way to focus on part of an input set.

Given a query and pairs of keys and values,

- Compute similarities between queries and keys
- Normalizes similarities via softmax to obtain attention scores
- Multiplies values by the scores

*Similarity function*

- Additive attention, e.g. $w^T \tanh(w_q q + w_k k)$
- Dot-product attention
Attention

Provides a way to focus on part of an input set.

Given a query and pairs of keys and values,

• Compute similarities between queries and keys
• Normalizes similarities via softmax to obtain attention scores
• Multiplies values by the scores

*Similarity function*

• Additive attention, e.g. $w^\top \tanh(w_q q + w_k k)$
• Dot-product attention

Can be *multihead*: re-compute the same process multiple times with different learnable weights, then aggregate.
Classical visual attention

Keys and values: “pixels” of the activation map of a CNN

Query: hidden state of the language model (typically LSTM)

Single head

Result becomes the visual input of the language model

\[
V = \{v_1, \ldots, v_{100}\}
\]
Image captioning model with attention

Attention function, $f$

\[
a_i = w^T \tanh(W_v v_i + Wh h) \\
\alpha = \text{softmax}(\alpha) \\
\hat{\nu} = \sum_{i=1}^{k} \alpha_i v_i
\]
1. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... & Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *ICML* 2015.

(b) A large white bird standing in a forest.
Extends the pool of keys/values by adding a “visual sentinel”

“a component that the model can fall back on when it chooses to not attend to the image”

Technically, obtained by gating the LSTM memory cell via a learned function.

Recently, repurposed by [2] with learnable persistent memories on the Transformer.

So far:

- Captioning with global feature vector (early approaches)
- Captioning with attention over a grid of features
  - Fixed grid doesn’t go well with objects of variable size
  - Grid pyramid?

... anything better?
So far:
- Captioning with global feature vector (early approaches)
- Captioning with attention over a grid of features
  - Fixed grid doesn’t go well with objects of variable size
  - Grid pyramid?

... anything better?

Object-based attention!
Use objects are attention candidates
• Detections features (i.e. keys and values) are from Faster-RCNN trained on Visual Genome (objects, context, attributes)

• Uses predictions from beam search for SCST finetuning

• 1° place in the COCO-captions leaderboard (July 2017 e for quite a long time)

1-D Convolutions as the step function

Operates on all words in parallel

Convolution needs to be masked so that predictions do not depend on the future.

Convolutional image captioning (CVPR 2018)

Multi-layer architecture with GLU (Gated Linear Units)

+ Attention on grid of features

At training time

- Train the model to predict a word given the previous ground-truth words.

If the step function does not depend on its own output at previous timesteps:

- We can parallelize over the t axis.
  - \( \rightarrow \) Training time reduction
  - E.g. Conv1D, Transformer

\( w_t \) : ground-truth words
At training time

- *Train the model to predict a word given the previous ground-truth words.*

If the step function does not depend on its own output at previous timesteps:

- We can parallelize over the t axis.
  - → Training time reduction
  - E.g. Conv1D, Transformer

\( w_t \) : ground-truth words
At training time

- *Train the model to predict a word given the previous ground-truth words.*

If the step function does not depend on its own output at previous timesteps:

- We can parallelize over the t axis.
  - → Training time reduction
  - E.g. Conv1D, Transformer

\( w_t \): ground-truth words
1D Convolutions as the step function

Very efficient implementations, compared to hand-designed LSTM implementations

Trivial to parallelize when pre-training with cross-entropy

Manages local dependencies
1D Convolutions as the step function

Very efficient implementations, compared to hand-designed LSTM implementations

Trivial to parallelize when pre-training with cross-entropy

Manages local dependencies

Train/test discrepancy and RL-finetuning applies also here

• They break parallelism on t axis
• Might not be easy to implement efficiently
  • Naïve solution: design a unique function that works on the whole sequence, than call it for [0, t] with 0<=t<=T. Very inefficient
  • Embed “states” in modules to memorize the results of previous iterations.

---

Self Attention

“Refine” each element of the sequence by treating it as \textit{query}, and the whole sequence as keys and values.

Receptive field is infinite!

Constant path length between two different positions

Trivial to parallelize during pre-training

Multi-head

Different attention layers in parallel (heads), each using different linear transformations.

Different heads can learn different relationships.
Masking

Inside the language model, both convolution and self-attention need to be masked.
Self Attention

Usually: scaled dot-product attention
(dot-product + scaling by sqrt(D))

Queries, keys and values are obtained from a linear transformation of the input sequence.

Multi-head

Different attention layers in parallel (heads), each using different linear transformations.

*Easy to express it in matrix form.*

Then concatenate and linear transformation to return to the model dimensionality.

Encoder
Uses self-attention on its input
Multiple attention layers stacked together (with add+norm) and feed-forward layers (linear layers applied timewise).

Decoder
Self-attention on words
Cross-attention on encoder outputs: use decoder sequences as queries, encoder outputs as key/values.

Object Relation Transformer: weights attentive scores in the region encoder using pairwise geometric features to encode spatial relationships between regions.

There is more than just a set of detections!

Encodes spatial and (predicted) semantic relationships through Graph Convolutional Neural Networks, and uses it as a replacement of plain attention.

The decoder has no idea whether the attended vector is related to the query.

**Attention on attention**

From query and attention result, generate an “attention gate” and an “information vector”, which are then multiplied.

Reminds the GLU 😊

Used both to encode regions and in the language model.

**Current SoTA on COCO:** 129.8 CIDEr with a single model
132.0 CIDEr with an ensemble

---

At some point... we will want to exploit alternative sources other than a paired training set

**Nocaps**: novel object captioning at scale

Contains 15k images with objects from OpenImages. Ground-truth captions are not available.

Nearly 400 object classes not seen during training.

• Captioning deals with the fundamental goal of connecting Vision and Language in a generative way.
• State of the art practices which are still fundamental to get SoTA performances:
  • SCST fine-tuning
  • object-based attention
• New challenges, new players:
  • The transformer: faster training, possibly a way to surpass SoTA
  • We won’t get the entire world as training set: extending to unseen objects will be crucial
Thanks!
Questions?

Lorenzo Baraldi
{name.surname}@unimore.it

University of Modena and Reggio Emilia, Italy