When an Image Tells a Story: The Role of Visual and Semantic Information for Generating Paragraph Descriptions

Nikolai Ilinykh , Simon Dobnik

Centre for Linguistic Theory and Studies in Probability (CLASP)
Department of Philosophy, Linguistics and Theory of Science (FLoV)
University of Gothenburg, Sweden

17^{th} December 2020

Describing images with longer sequences[1]

People are standing on the grass behind a concrete patch that looks like it was just set. There are two orange cones in front of the concrete and yellow tape surrounding it. There are three people in yellow vests and white hard hats. There are some people sitting on a bench next to them.

[1] Krause, J., Johnson, J., Krishna, R., & Fei-Fei, L. (2017). A Hierarchical Approach for Generating Descriptive Image Paragraphs. In Computer Vision and Pattern Recognition (CVPR).

Properties of Image Paragraphs (IP)

People are standing on the **grass** behind **a concrete patch** that looks like it was just set. There are **two orange cones** in front of **the concrete and yellow tape** surrounding it. There are **three people in yellow vests and white hard hats**. There are **some people sitting on a bench** next to them.

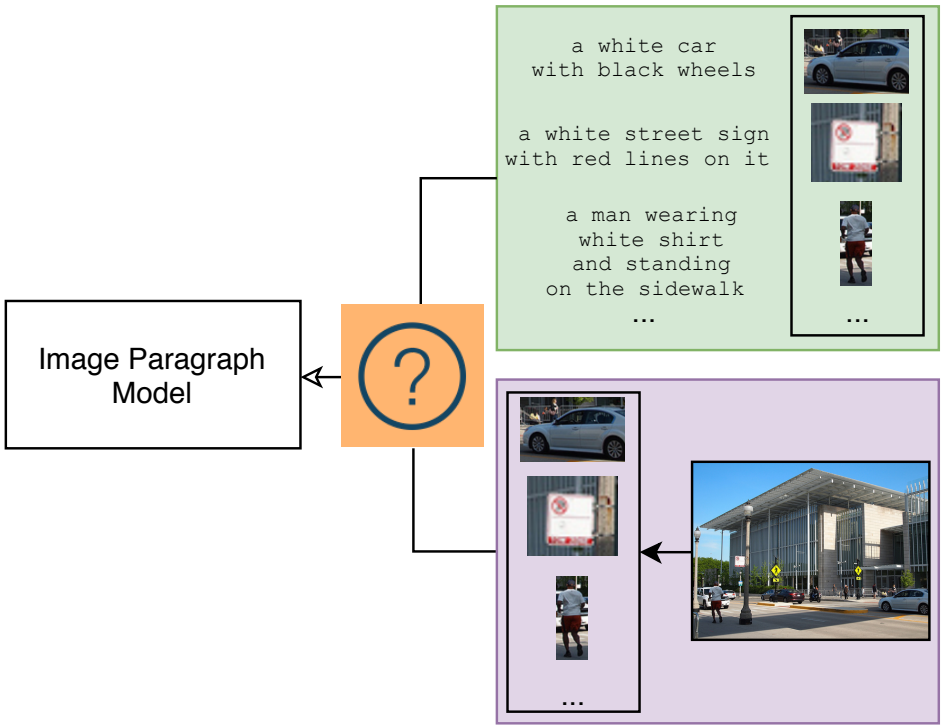
Two Sources of Important Information for IP

1. visual features of perceived objects (*what* to refer to)
2. background knowledge and communicative intent (*when* and *how* to refer)

People are standing on the **grass** behind **a concrete patch** that looks like it was just set. There are **two orange cones** in front of **the concrete and yellow tape** surrounding it. There are **three people in yellow vests and white hard hats**. There are **some people sitting on a bench** next to them.

Our paper

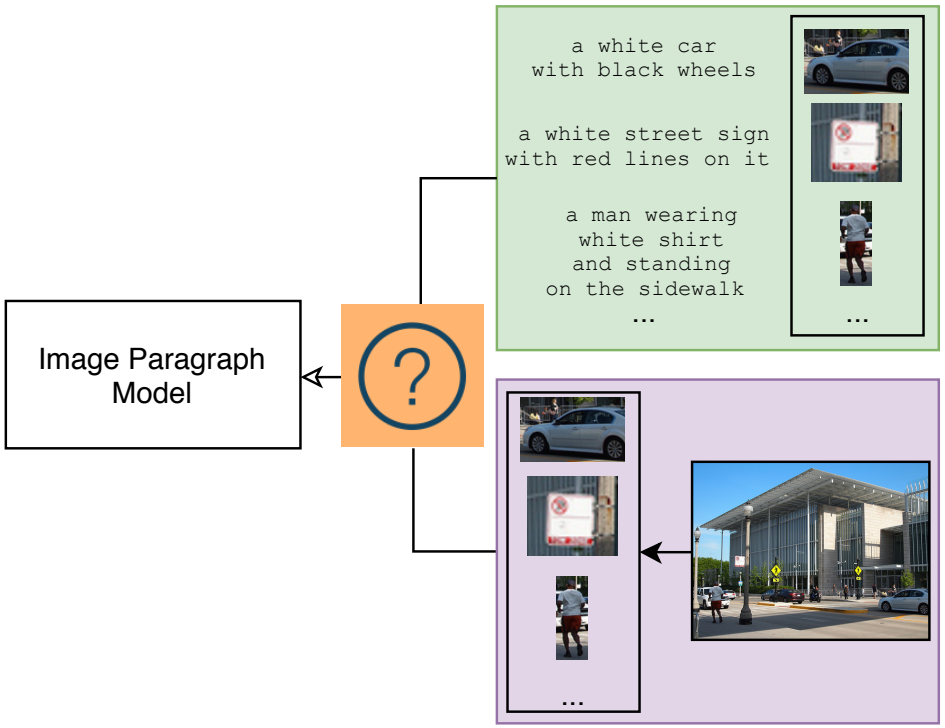
How to improve both *accuracy* and *diversity* of generated image paragraphs?



- **model input**:
  `unimodal` (visual / textual)
  vs. `multimodal`

Our paper

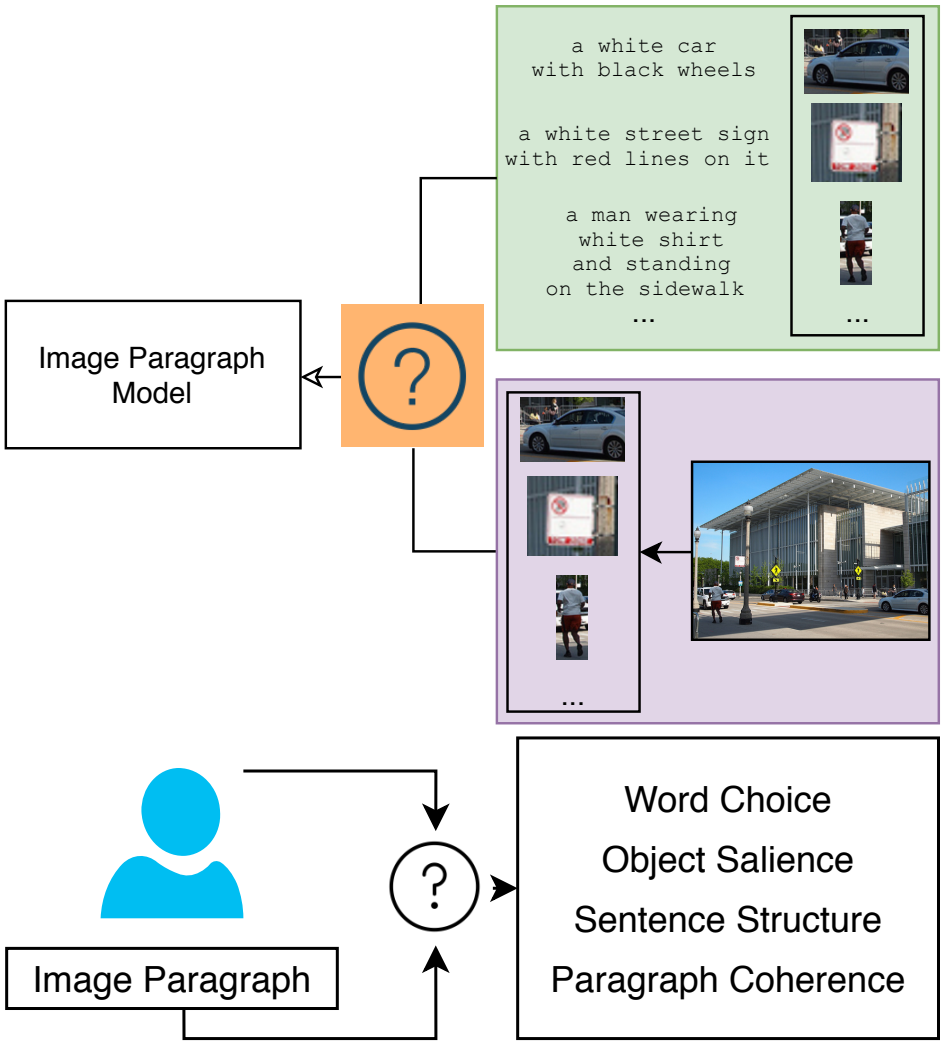
How to improve both *accuracy* and *diversity* of generated image paragraphs?



- **model input**:
  `unimodal` (visual / textual)
  vs. `multimodal`
- **information fusion**:
  `max-pooling` vs. `attention`

Our paper

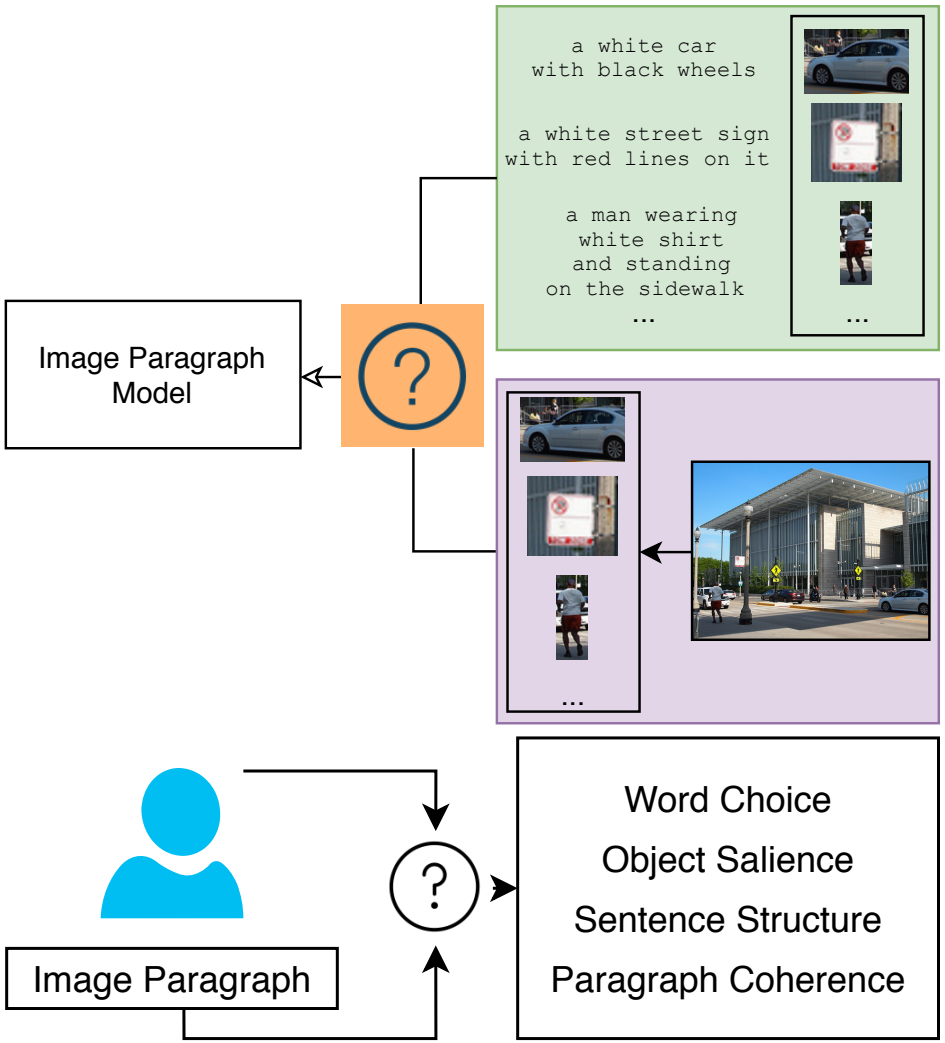
How to improve both *accuracy* and *diversity* of generated image paragraphs?



- **model input**: `unimodal` (visual / textual) vs. `multimodal`
- **information fusion**: `max-pooling` vs. `attention`
- **paragraph evaluation**: `automatic` vs. `human`

Our paper

How to improve both *accuracy* and *diversity* of generated image paragraphs?



- **model input**: `unimodal` (visual / textual) vs. `multimodal`
- **information fusion**: `max-pooling` vs. `attention`
- **paragraph evaluation**: `automatic` vs. `human`
- **human evaluation**: `accuracy` and `diversity` of generated paragraphs

Unimodal Features: Vision and Language

Tasks' description

- one sheet with exercises every week (consists of several tasks)
- corresponds to the topic of the lecture
- prerequisite for successful completion of the course: **pass all assignments**
- assistance and discussion of the solution during the weekly tutorials
- submission: Lernraum+ (can be uploaded in the respective folder's assignment)
- deadlines are important!

Assignments

are considered passed, if

- they are submitted on time
- you have solved the task and proposed solution to it (in other words, **your solution** must be clear and precise)
- your code (in case you've submitted it) runs (contains no bugs) and approximately does what it should to do
- you did not copy stuff from your fellow students

Deadlines

Programming project

- due to the last day of February
- provide ideas for the project before the end of January
  - if you have any now, you are welcome to discuss them with us
  - otherwise, you will be given the topic from the pool of projects

Homework

- each homework is due to the following Thursday, 10 AM

Tutorials

- once per week, participation is free
- discussion of weekly assignments
- detailed and practical exploration of programming knowledge
- a nice opportunity for collective learning, testing, solving programming issues...

Programming is ...

Programming is ...

building machine-friendly instructions

Programming is ...

building machine-friendly instructions

- **Do this, then do that.**
- **If this condition is true, perform this action; otherwise, do that action.**
- **Do this action that number of times.**
- **Keep doing that until this condition is true.**

Do you speak Python?

Python Quellcode

```
passwordFile = open('SecretPasswordFile.txt')
secretPassword = passwordFile.read()
print('Enter your password.')
typedPassword = input()
if typedPassword == secretPassword:
    print('Access granted.')
    if typedPassword == '12345':
        print('That password is one that an idiot \
        puts on their luggage.')
else:
    print('Access denied')
```

Quelle: https://automatetheboringstuff.com/chapter0/

Programming is not hard!

*https://devrant.com/rants/10875/multi-threaded-programming-explained-with-puppies

Python is...

... learning something readable and elegant

Saying 'hello' in Java

```
class Hello {

  public static void main (String[] args) {
      System.out.println ("Hello, world.");
  }
}
```

Saying 'hello' in Python

```
print('Hello, world.')
```

Literature

Books:

- Scripts
- Learning Python, O'Reilly [**?**]
- Programming Python, O'Reilly [**?**]
- Core Python Programming, Prentice Hall [**?**]

Literature

Online:

- Python Dokumentation
  `http://docs.python.org/`
- How to Think Like a Python Programmer, Green Tea Press
  `http://www.greenteapress.com/thinkpython/`
- Practical Programming for Total Beginners
  `https://automatetheboringstuff.com/`

Want to do some real-world applied programming?

Whenever possible: steal code!
(But not from your fellow students!)

Technical Details

1. install Python using one of the next methods:
   - download from `http://www.python.org`
   - version: 3.5.0
   - linux: Package manager
   - macOS: MacPorts
   - windows: Use Installer from the website
2. look at the editors out there
3. first look at the System Console ("Terminal", "Command Line Interface (CLI)")

Python Versions

- up-to-date versions:
  - 2.7.10
  - 3.5.0 +
- all versions are not completely compatible with each other
- new in 3.∗:
  - print is a function
  - all strings are written in unicode
  - new function names
  - faster iterations
  - ..., check `http://docs.python.org/3/whatsnew/3.0.html`
- we will use Python3.x in the class (sometimes show differences to Python2.x*).

Hand tools: Editors

- editors are **essential** tools in programming environment
- editors are your assistants when writing a bunch of code
- an emergency help to clear your ‘programming syntax’
- editors help with error detection (ideally)

Properties of good editors

- syntax highlighting
- automatic indentation
- good support of various encodings
- tabs, spaces, shiftwidth, ...
- abbreviations, templates, ...
- undo/redo-stack
- good code navigation
- plain text
- spelling correction in comments
- code completion

Editors

- (g)vim
- (x)emacs
- gedit
- kwrite
- scite
- jedit
- notepad++
- ...

Tips for dealing with editors

- try out different ones
- **work with the keyboard as much as possible**
  - Ctrl +Arrow keys: jump over words
  - Shift + Arrow keys: mark text pieces
  - learn more keyboard shortcuts (copying, pasting, skipping paragraphs, commenting text out, etc.)
- 'machine typing course' ?

pydoc

- help in the CLI
- is normally installed with Python
- quick overview of problems, questions, ...

ipython (notebook)

- interactive python
- tab expansion
- coloured display
- error messages are easily detected
- notebooks: browser-based variants of Python editors, elegant combination of code, comments, formulas, plots, etc.

First Homework

- install Python3.6+ **or**
- install iPython
- familiarise yourself with the CLI and Python IDLE
- install an editor
- scan/skim through the Python Documentation
- complete first exercises (will be available after the class)

Tips for the homework (click on bold text to access links)

- you can install Python from **the official website** or from the **Anaconda distribution**
  - Anaconda allows you to safely remove various Python versions if some of them are broken
- we will use Jupyter Notebook in our class, please look at **the official JN guidelines** for more information, follow them and complete installations

Literature