**Aim and Motivation**

- Co-reference resolution is an important and complex task
- Provide toolkit for error analysis and visualization: facilitates research and system engineering

After the discussion, Obama confirmed he will return.

Then the president and his bodyguards left.

**Method**

- Implement framework of Martschat and Strube (2014)
- For recall errors, compare reference entity spanning trees with partition by system entities
- For precision errors, switch roles of reference and system

Obtain *cort*

- Python 2/3 library, available at PyPi: `pip install cort`
- Source code at http://github.com/smartscat/cort

**Features**

- Manage output on data following the CoNLL format:
  ```python
  ref = Corpus.from_file("ref", open("ref.conll"))
  ex = ErrorExtractor(ref, recall_accessibility, precision_system_output)
  pair = Corpus.from_file("pair", open("pair.conll"))
  pair.read_antecedents(open("pair.antecedents"))
  tree = Corpus.from_file("tree", open("tree.conll"))
  ex.add_system(pair)
  ex.add_system(tree)
  errors = ex.get_errors()
  ```

- Filter and categorize sets of errors:
  ```python
  close = errors.filter(lambda e: e[0].attributes["sentence_id"] - e[1].attributes["sentence_id"] <= 3)
  close = close.categorize(lambda e: e[0].attributes["type"])
  ```

- Plot error statistics:
  ```python
  plot([["pair", [(cat, len(errs)) for cat, errs in close["pair"]["recall_errors"]["all"].items()]],
  ["tree", [(cat, len(errs)) for cat, errs in close["tree"]["recall_errors"]["all"].items()]],
  ["Recall Errors", ["Type of anaphor", "Number of Errors"]])
  ```

- Train and run simple, well-performing coreference resolution systems:
  - Mention pair model trained via a perceptron
  - Customizable with respect to features, instance extraction, decoding and clustering
  - Command-line tools *cort-train* and *cort-predict*

**Visualize and browse errors:**

```
close.visualize("pair")
```