On Discriminative GF models for Parsing and Translation

Xavier Carreras, Stefan Bott, Daniele Pighin, Lluís Màrquez

UPC

Discriminative GF

Main goal: increase robustness of GF grammars

We look at it from a machine learning perspective.

Outline:

- Discriminative learning methods for parsing
- Applications to GF
- Discussion

Two subgoals

- Parsing: map sentence x into a structure y
  - In GF $y = (c, a)$, c is concrete syntax, a is abstract syntax
  - The mapping could take a probabilistic form: $p(a, c|x)$
- Generation: map abstract syntax m into a sentence z
  - The mapping could take a probabilistic form: $p(z|a)$

Two subgoals

- Parsing: map sentence x into a structure y
  - In GF $y = (c, a)$, c is concrete syntax, a is abstract syntax
  - The mapping could take a probabilistic form: $p(a, c|x)$
- Generation: map abstract syntax m into a sentence z
  - The mapping could take a probabilistic form: $p(z|a)$

Then we can implement a translator via *interlingua*:

$$p(z|x) = \sum_{(c,a)} p(c, a|x)\; p(z|a)$$

Discriminative GF models

- Main focus: parsing
- Linear structured prediction model:

$$y^*(x) = \operatorname*{argmax}_{y \in \mathcal{G}(x)} \mathbf{w} \cdot F(x, y)$$

where

- $\mathcal{G}(x)$ enumerates derivations of x under a GF grammar
- $F(x, y)$ is a feature representation of x and y
- $\mathbf{w}$ are the parameters of the model

Discriminative GF models

- Main focus: parsing
- Linear structured prediction model:

$$
\begin{aligned}
y^*(x) &= \operatorname*{argmax}_{y \in \mathcal{G}(x)} \mathbf{w} \cdot F(x, y) \\
&= \operatorname*{argmax}_{y \in \mathcal{G}(x)} \sum_{r \in y} \mathbf{w} \cdot f(x, r)
\end{aligned}
$$

where

- $\mathcal{G}(x)$ enumerates derivations of x under a GF grammar
- $F(x, y)$ is a feature representation of x and y
- $\mathbf{w}$ are the parameters of the model
- $f(x, r)$ is a part-based feature represention

Discriminative parsing: three problems

$$y^*(x) = \operatorname*{argmax}_{y \in \mathcal{G}(x)} \sum_{r \in y} \mathbf{w} \cdot f(x, r)$$

- Representation: what are $r \in y$? what is $f(x, r)$?
- Inference: how to search for $y^*(x)$?
- Learning: how to obtain $\mathbf{w}$ from data?

Discriminative parsing: three problems

$$y^*(x) = \underset{y \in \mathcal{G}(x)}{\text{argmax}} \sum_{r \in y} \mathbf{w} \cdot f(x, r)$$

- Representation: what are $r \in y$? what is $f(x, r)$?
  - In GF, each r is related to a production (`lin` and `fun`)
  - $f(x, r)$ should capture predictive features
- Inference: how to search for $y^*(x)$?
  - With GF parsing algorithms, for weighted grammars
- Learning: how to obtain $\mathbf{w}$ from data?

Structured Prediction Framework

$$y^*(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{r \in y} \mathbf{w} \cdot f(x, r)$$

- Learning $\mathbf{w}$ from a training set of (x, y) pairs:
  - CRFs (Lafferty et al. '01)
  - Structured Perceptron (Collins '02)
  - Max-margin methods (Taskar et al. '03)
- Inference:
  - Black box wrt. type of structures and parsing methods
  - Required algorithms: 1-best solution, marginals
  - Used intensively during training

Structured Prediction Framework

$$y^*(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} \sum_{r \in y} \mathbf{w} \cdot f(x, r)$$

- Learning $\mathbf{w}$ from a training set of (x, y) pairs:
  - CRFs (Lafferty et al. '01) $\leftarrow$ probabilistic
  - Structured Perceptron (Collins '02)
  - Max-margin methods (Taskar et al. '03)
- Inference:
  - Black box wrt. type of structures and parsing methods
  - Required algorithms: 1-best solution, marginals
  - Used intensively during training

Structured Prediction for Parsing

- Formalisms:
  - CFG (Finkel et al. ’08)
  - Dependency grammars (McDonald et al. ’05; ’06)
  - CCG (Clark & Curran ’04)
  - TAG (Carreras et al. ’08)
- Efficiency of parsing algorithms is critical for training
- Good features: phrase structure, head-modifier,
  and lexicalized versions
- Some work on *partial supervision*:
  - Semi-supervised representations (Koo et al. ’08)
  - Grammar refinements (Petrov & Klein ’08; Musillo ’09)
  - Grammar induction in CCG semantic parsing (Zettlemoyer & Collins ’05; Kwiatkowski et al. ’10)

Structured Prediction for Parsing

- Formalisms:
  - CFG (Finkel et al. ’08)
  - Dependency grammars (McDonald et al. ’05; ’06)
  - CCG (Clark & Curran ’04)
  - TAG (Carreras et al. ’08)
- Efficiency of parsing algorithms is critical for training
- Good features: phrase structure, head-modifier, and lexicalized versions
- Some work on *partial supervision*:
  - Semi-supervised representations (Koo et al. ’08)
  - Grammar refinements (Petrov & Klein ’08; Musillo ’09)
  - Grammar induction in CCG semantic parsing (Zettlemoyer & Collins ’05; Kwiatkowski et al. ’10)

Structured Prediction for Parsing

- Formalisms:
  - CFG (Finkel et al. ’08)
  - Dependency grammars (McDonald et al. ’05; ’06)
  - CCG (Clark & Curran ’04)
  - TAG (Carreras et al. ’08)
- Efficiency of parsing algorithms is critical for training
- Good features: phrase structure, head-modifier,
  and lexicalized versions
- Some work on *partial supervision*:
  - Semi-supervised representations (Koo et al. ’08)
  - Grammar refinements (Petrov & Klein ’08; Musillo ’09)
  - Grammar induction in CCG semantic parsing (Zettlemoyer & Collins ’05; Kwiatkowski et al. ’10)

Scenario 1: ambiguous GF grammars

- problem: a GF grammar is ambiguous: for some sentences it defines several derivations and some are incorrect interpretations

  ```
  eats fish with sauce
  eats fish with chopsticks
  ```

Scenario 1: ambiguous GF grammars

- problem: a GF grammar is ambiguous: for some sentences it defines several derivations and some are incorrect interpretations

```
eats fish with sauce
eats fish with chopsticks
```

- A solution:
  - Use $p(a, c|x)$ to select best derivation

Scenario 1: ambiguous GF grammars

- problem: a GF grammar is ambiguous: for some sentences it defines several derivations and some are incorrect interpretations

  ```
  eats fish with sauce
  eats fish with chopsticks
  ```

- A solution:
  - Use $p(a, c|x)$ to select best derivation
- question: what kind of ambiguities are frequent in GF grammars for MOLTO?

Scenario 2: unknown words and phrases

- problem: a GF grammar is too restricted: it does not cover all lexical items of our data

  `the cat eats unknown food`

Scenario 2: unknown words and phrases

- problem: a GF grammar is too restricted: it does not cover all lexical items of our data

  `the cat eats unknown food`

- assumption: the abstract/concrete types of the GF grammar are complete
- A solution:
  - Consider all possible abstract/concrete types for an unknown word or phrase
  - Weight each assignment with discriminative methods
  - Use most likely assignment(s) in standard GF
  - Use a standard SMT system to translate the unknown phrase in context

Scenario 3: unknown concrete rules

- problem: a GF grammar does not cover all linearizations of some abstract rule

  `the cat fish eats`

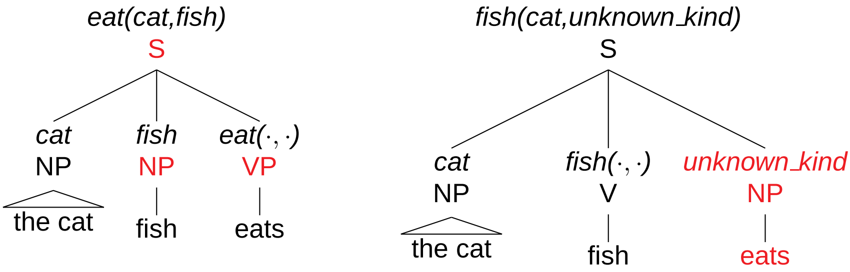
Scenario 3: unknown concrete rules

- problem: a GF grammar does not cover all linearizations of some abstract rule

  `the cat fish eats`

- A solution:
  - Consider all possible linearizations of abstract rules (i.e. permutations of the terms in an abstract rule)
  - Weight permutations with discriminative methods

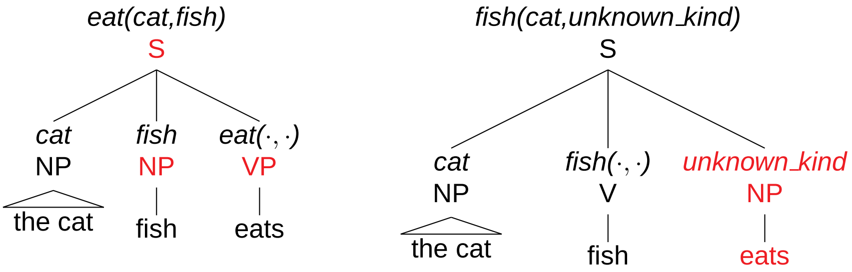
e.g. `the cat fish eats`



How can we predict the correct one?

- In this case, the correct tree (left side) has known abstract structure
- In general:
  - Features at lexical, concrete and abstract levels
  - Let the learning methods figure out the correct weightings

e.g. `the cat fish eats`



How can we predict the correct one?

- In this case, the correct tree (left side) has known abstract structure
- In general:
  - Features at lexical, concrete and abstract levels
  - Let the learning methods figure out the correct weightings

A real scenario?

- GF grammar: both ambiguous and restrictive
- Is abstract syntax always reliable?
- Main question: what are good decompositions in GF?
  - Efficiently parseable
  - Allow predictive features
- Main challenge for robustness:
  - How to make GF flexible (type prediction, permutations)?
  - How to discard spurious derivations?

A real scenario?

- GF grammar: both ambiguous and restrictive
- Is abstract syntax always reliable?
- Main question: what are good decompositions in GF?
  - Efficiently parseable
  - Allow predictive features
- Main challenge for robustness:
  - How to make GF flexible (type prediction, permutations)?
  - How to discard spurious derivations?

A real scenario?

- GF grammar: both ambiguous and restrictive
- Is abstract syntax always reliable?
- Main question: what are good decompositions in GF?
  - Efficiently parseable
  - Allow predictive features
- Main challenge for robustness:
  - How to make GF flexible (type prediction, permutations)?
  - How to discard spurious derivations?

Three forms of supervision

- **Full**
  - Training examples: sentences paired with concrete and abstract syntax
  - Learn a parser in the standard way
- **Abstract**
  - Training examples: sentences paired with abstract syntax
  - Learn a parser that induces the concrete syntax, following (Zettlemoyer & Collins '05)
- **Hybrid**
  - Training examples: sentences paired with abstract syntax
  - Take advantage of resource grammar
  - Learn alignments between output of RG and abstract syntax

Extra Slides

GF Deductive Rules from (Angelov '09)

INITIAL PREDICT

$$\frac{S \to f[\vec{B}]}{[^0_0 S \to f[\vec{B}]; 1 : \bullet\alpha]} \quad S \text{ - start category}, \alpha = \mathrm{rhs}(f, 1)$$

PREDICT

$$\frac{B_d \to g[\vec{C}] \qquad [^k_j A \to f[\vec{B}]; l : \alpha \bullet \langle d; r \rangle\ \beta]}{[^k_k B_d \to g[\vec{C}]; r : \bullet\gamma]} \quad \gamma = \mathrm{rhs}(g, r)$$

SCAN

$$\frac{[^k_j A \to f[\vec{B}]; l : \alpha \bullet s\ \beta]}{[^{k+1}_j A \to f[\vec{B}]; l : \alpha\ s \bullet \beta]} \quad s = w_{k+1}$$

COMPLETE

$$\frac{[^k_j A \to f[\vec{B}]; l : \alpha\bullet]}{N \to f[\vec{B}] \qquad [^k_j A; l; N]} \quad N = (A, l, j, k)$$

COMBINE

$$\frac{[^u_j A \to f[\vec{B}]; l : \alpha \bullet \langle d; r \rangle\ \beta] \qquad [^k_u B_d; r; N]}{[^k_j A \to f[\vec{B}\{d := N\}]; l : \alpha\ \langle d; r \rangle \bullet \beta]}$$

The Structured Perceptron

(Collins, 2002)

- Set $\mathbf{w} = \mathbf{0}$
- For $t = 1 \ldots T$
  - For each training example $(\mathbf{x}, \mathbf{y})$
    1. Compute $\mathbf{z} = \arg\max_{\mathbf{z}} \sum_{r \in \mathbf{z}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, r)$
    2. If $\mathbf{z} \neq \mathbf{y}$

$$\mathbf{w} \leftarrow \mathbf{w} + \sum_{r \in \mathbf{y}} \mathbf{f}(\mathbf{x}, r) - \sum_{r \in \mathbf{z}} \mathbf{f}(\mathbf{x}, r)$$

- Return $\mathbf{w}$

The Structured Perceptron with Averaging

(Freund and Schapire, 1998)

- Set $\mathbf{w} = \mathbf{0}$, $\mathbf{w_a} = \mathbf{0}$
- For $t = 1 \ldots T$
  - For each training example $(\mathbf{x}, \mathbf{y})$
    1. Compute $\mathbf{z} = \arg\max_{\mathbf{z}} \sum_{r \in \mathbf{z}} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, r)$
    2. If $\mathbf{z} \neq \mathbf{y}$
    $$\mathbf{w} \leftarrow \mathbf{w} + \sum_{r \in \mathbf{y}} \mathbf{f}(\mathbf{x}, r) - \sum_{r \in \mathbf{z}} \mathbf{f}(\mathbf{x}, r)$$
    3. $\mathbf{w_a} = \mathbf{w_a} + \mathbf{w}$
- Return $\mathbf{w_a}/NT$, where N is the number of training examples