Going from UD towards AMR

Kelvin Han, Siyana Pavlova

3rd June 2019

UNIVERSITÉ DE LORRAINE | IDMC Institut des sciences du Digital Management & Cognition | Loria Laboratoire lorrain de recherche en informatique et ses applications

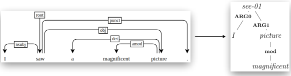
Supervised Project

with Bruno Guillaume and Maxime Amblard

Overview

Objective

Develop a system which transforms **Universal Dependencies (UD)** annotated sentences to **Abstract Meaning Representation (AMR)**

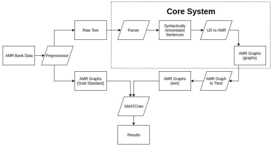


Motivation

- Limited amount of semantically-oriented annotated data, especially for languages other than English.
- Semantically-oriented annotated data useful for many aspects of NLP.
- Large amount of UD annotated data in more than 70 languages.

Therefore, it is important to expand the pool of semantically-oriented annotated data and UD annotated data can serve as an excellent starting point.

System Architecture



Frameworks

AMR

AMR is a **semantic representation framework**, which focuses on the **predicate-argument structure** of a sentence. It captures "who does what to whom" in a single simple data structure. AMR makes extensive use of PropBank predicate frames.

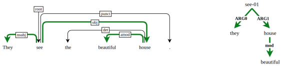
UD

UD is a framework for **syntactic annotation** whose goal is to build a **cross-linguistically consistent** treebank. The principles driving the UD syntactic annotation are of **dependency** and **lexicalism**, meaning that syntactic words are the units of grammatical annotation. Furthermore, UD prefers **content words** over functional words and punctuation as the heads of relations.

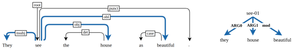
UD to AMR

Due to the nature of both frameworks, there is some parallel between UD graphs and AMR graphs.

"They see the beautiful house"



"They see the house as beautiful"



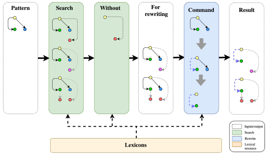
Walkthrough

Grew

Grew is a **graph rewriting tool** for NLP, developed by researchers at Loria. It uses the Ocaml programming language and there is a Python package available. Grew allows for:

- **Searching** through graphs using **patterns**;
- Applying **rewrite commands**;
- Creating parameters for rules using **lexicons;**
- Combining patterns into **strategies and packages.**

Graph Rewriting System (GRS)



Lexicons

There are two main reasons why lexicons are useful in the context of this project:

- Predicate sense disambiguation;
- Choosing the correct argument structure.

Our lexicon was constructed by manually annotating **834 PropBank predicates**, enriching them with semantic role labels.

Results

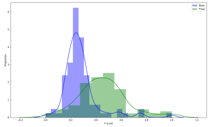
System Evaluation

The system was evaluated on the first 100 sentences of the AMR Bank*. **Our Final GRS** encompases 183 GRS rules, grouped in 10 packages. 55 of the rules do not have a repetitive structure. A lexicon containing annotations of all predicates contained in the first 100 sentences was used too.

A comparison with **Base**, which contains a minimal set of preparatory rules and without a lexicon, shows a **0.19** increase in F1-score.

	Base			Our Final GRS		
	Min	Max	Arithmetic Mean	Min	Max	Arithmetic Mean
Precision	0.00	1.00	0.27	0.00	1.00	0.47
Recall	0.00	1.00	0.28	0.00	1.00	0.46
F1-score	0.00	1.00	**0.27**	0.00	1.00	**0.46**

*The AMR Bank contains the 1,562 sentences of The Little Prince in English, annotated by human annotators. It it provided by the Linguistics Data Consortium.



Lexicon Inter-Annotator Agreement

497 randomly selected PropBank predicates were annotated by the two authors. An **inter-annotator agreement** of **0.65** across the 1,273 annotated semantic roles was achieved. This was computed using Cohen's Kappa.

The code for this project can be found under https://github.com/siyanapavlova/AMR_annotations.