Learning the Semantics of some Natural Language Constructions

Nina Gierasimczuk

Institute for Logic, Language and Computation
Universiteit van Amsterdam

PALMYR-V
June 1th, 2007

Outline

Outline

Quantifiers Learning

- Investigated by van Benthem, Clark, Costa Florencio, Tiede.
- Acquisition of *NL* quantifiers — collecting procedures for computing their denotations.
- Analysing *NL* quantifiers from the point of view of syntax learning models.

Outline

Quantifiers

Restrictions

- Finite models only.
- Quantifiers as classes of finite models.
- Restriction to monadic quantifiers — sufficient for linguistics.

Quantifiers

Lindström definition

Definition

A monadic generalized quantifier of type $\underbrace{(1, \ldots, 1)}_{n}$ is a class Q of structures of the form $M = (U, A_1, \ldots, A_n)$, where A_i is a subset of U. Additionally, Q is closed under isomorphism.

Quantifiers

Examples of Classes

- Existential Quantifier:
  $K_{\exists} = \{(U, A) : A \subseteq U \wedge A \neq \emptyset\}$.
- Universal Quantifier:
  $K_{\forall} = \{(U, A) : A = U\}$.
- Most:
  $K_{MOST} = \{(U, A) : |A - B| < |A \cap B|\}$.

Quantifiers

Examples of Classes

- Existential Quantifier:
  $K_{\exists} = \{(U, A) : A \subseteq U \wedge A \neq \emptyset\}$.
- Universal Quantifier:
  $K_{\forall} = \{(U, A) : A = U\}$.
- Most:
  $K_{MOST} = \{(U, A) : |A - B| < |A \cap B|\}$.

Quantifiers

Examples of Classes

- Existential Quantifier:
  $K_{\exists} = \{(U, A) : A \subseteq U \wedge A \neq \emptyset\}$.
- Universal Quantifier:
  $K_{\forall} = \{(U, A) : A = U\}$.
- Most:
  $K_{MOST} = \{(U, A) : |A - B| < |A \cap B|\}$.

Quantifiers
Encoding

Aim

- Encode models as words with certain features.
- Class of models as sets of words (language).
- Use of the concept of constituents.

Quantifiers

Encoding

- List all elements of the model, e.g.: $c_1, \ldots, c_5$.
- Label every element with one of the letters: $a_{\bar{A}\bar{B}}, a_{A\bar{B}}, a_{\bar{A}B}, a_{AB}$, according to constituents it belongs to.
- Get the word $\alpha_M = a_{\bar{A}\bar{B}} a_{A\bar{B}} a_{AB} a_{\bar{A}B} a_{\bar{A}B}$.
- α_M describes the model in which: $c_1 \in \bar{A}\bar{B}, c_2 \in A\bar{B}, c_3 \in AB, c_4 \in \bar{A}B, c_5 \in \bar{A}B$.
- The class K_Q is represented by set of words describing all models from the class.

Quantifiers

Encoding — Illustration

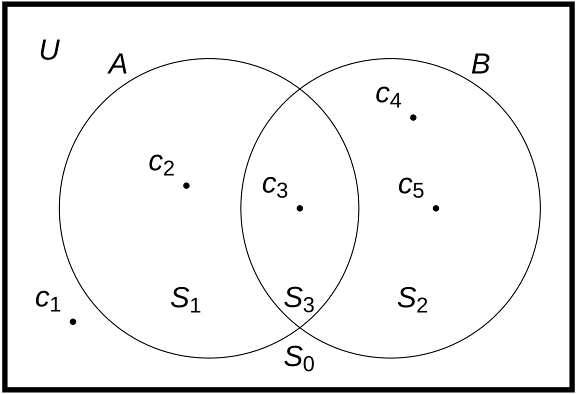

Figure: This model is uniquely described by

$\alpha_M = a_{\bar{A}\bar{B}} a_{A\bar{B}} a_{AB} a_{\bar{A}B} a_{\bar{A}B}$.

Quantifiers
Corresponding Devices

Definition

Let $\mathcal{D}$ be a class of recognizing devices, and Ω be a class of monadic quantifiers. $\mathcal{D}$ *accepts* Ω if and only if for every monadic quantifier Q:

$$(Q \in \Omega \iff \text{there is device } A \in \mathcal{D}(A \textit{ accepts } L_Q)).$$

Quantifiers

Corresponding Devices - Results

Theorems

- *Quantifier Q is first–order definable* $\iff$ L_Q *is accepted by some acyclic finite automaton. [van Benthem 1984]*
- *Monadic quantifier Q is definable in the divisibility logic* $FO + D_\omega$ $\iff$ L_Q *is accepted by some finite automaton. [M. Mostowski 1998]*
- *Quantifier of type (1) is semilinear (elementary definable in the structure* $(\omega, +)$*)* $\iff$ L_Q *is accepted by push–down automaton. [van Benthem 1986]*

There are *NL* quantifiers outside context–free languages.

Outline

Learning
Identification in the limit

Procedure is infinite. In each step:

- learner is given a unit of data;
- learner has only a finite set of information;
- learner chooses a name of a language.

The language is identified in the limit if after some time guesses remain the same and correct.

Class of languages is identified in the limit, if there is a learner that identifies in the limit every language from this class.

Learning

Identification in the Limit — Data Presentation

Definition

TEXT for language L is an ω – sequence, I, of words $\alpha_1, \alpha_2, \ldots \in L$, such that every word $\alpha \in L$ occurs at least once in I. (also: positive information)

Definition

INFORMANT for language L is an ω – sequence, I, of elements of $(A^* \times \{0, 1\})$, such that for each $\alpha \in A^*$:

$$(\alpha, 1) \textit{ is in } I \quad \textit{if } \alpha \in L$$
$$(\alpha, 0) \textit{ is in } I \quad \textit{if } \alpha \notin L.$$

(also: positive and negative information)

Learning

Identification in the Limit — Main Results

Anomalous text	Recursively enumerable
	Recursive
Informant	Primitive recursive
	Context–sensitive
	Context–free
	Regular
Text	Finite cardinality languages

Table: Identifiability in the Limit Results.

Learning of Quantifiers
Identification Result

Fact

The classes of FO, FO $+ D_\omega$ and semilinear quantifiers are not identified in the limit using text but are identified using informant.

Learning — again...

Effective Algorithms — L^* [Angluin]

- Finite and effective identification of regular language.
- Identifies language by finding a proper *DFA*.
- Use of queries.
- Controlled by the so-called *Minimally Adequate Teacher*.

Learning

Effective Algorithms — Angluin's L^*

- Learner ask two types of questions:
    1. Membership: Is sequence α in the unknown language?
    2. Extensional equivalence: Is *DFA* produced by L^* equivalent to the *DFA* corresponding to unknown language. If not, the teacher gives a counterexample.
- L^* identifies every regular language.
- L^* works in polynomial time.

Learning

Effective Algorithms — Sakakibara's *LA*

- Developement of L^* — wider classes of languages.
- Translation of L^* to context–free grammars.
- The same procedure as in L^*.
- *LA* identifies every *CFG*.
- *LA* works in polynomial time.

Outline

Effective Learning of Quantifiers

Results

Fact

Monadic FO $+ D_\omega$–definable quantifiers are learnable using L^*–algorithm.

Fact

Semilinear quantifiers of type (1) are learnable using LA–algorithm.

Further Research

- Rethink adequacy of syntax learning models for the problem of semantics learning.
- Ordering semantic constructions by their learning complexity.
- Comparison of learning complexity and model–checking complexity.