Learning Word-to-Meaning Mappings in Logical Semantics

Makoto Kanazawa
University of Tokyo

Abstract. I generalize Siskind's problem of learning word-to-meaning mappings to logical semantics, and formulate the heart of the problem as a problem in typed lambda calculus. I show that every instance of this 'mapping problem' has infinitely many solutions, but many of them are equivalent in a certain sense. I show an algorithm for finding a reasonably small solution with respect to a certain 'definability' relation between terms.

Siskind (1996, 2000) discusses the problem of learning word-to-meaning mappings from sentences coupled with their meanings. In his setup, sentence meanings are represented by first-order terms built up from constants and function symbols (called *conceptual symbols*), and word meanings are represented by first-order terms with zero or more individual variables. For instance, the meaning of *John lifted Mary* may be represented by

$$\mathrm{CAUSE}(\mathbf{John}, \mathrm{GO}(\mathbf{Mary}, \mathrm{UP})),$$

where CAUSE, **John**, GO, **Mary**, and UP are conceptual symbols. The learner is fed with a stream of word strings coupled with expressions like the one above, and is expected to arrive at a word-to-meaning mapping like:

$$\begin{array}{lcl} \textit{John} & \mapsto & \mathbf{John} \\ \textit{lifted} & \mapsto & \mathrm{CAUSE}(x, \mathrm{GO}(y, \mathrm{UP})) \\ \textit{Mary} & \mapsto & \mathbf{Mary} \end{array} \tag{1}$$

Semantic composition is supposed to be done by substitution. An equivalent way of looking at it is to view word meanings as λ-terms like $\lambda xy.\mathrm{CAUSE}(x, \mathrm{GO}(y, \mathrm{UP}))$, where all subterms are of *first-order* types, and semantic composition as application plus β-reduction.

Siskind shows that one can reduce the complexity of learning by breaking down the learning process into two phases. In the first phase, the learner forms, for each word, the set of conceptual symbols involved in the meaning of that word. This can be done efficiently by an online algorithm that maintains two sets of conceptual symbols: the set of symbols that are possibly in the correct word meaning, and the set of symbols that are necessarily in the correct word meaning. The second phase then builds the correct meaning expression out of the symbol set learned in the first phase. For instance, suppose that the learner has arrived at the following association of words with symbol sets:

$$\begin{array}{lcl} \textit{John} & : & \{\mathbf{John}\} \\ \textit{lifted} & : & \{\mathrm{CAUSE}, \mathrm{GO}, \mathrm{UP}\} \\ \textit{Mary} & : & \{\mathbf{Mary}\} \end{array}$$

Then, given that the sentence *John lifted Mary* has meaning $\mathrm{CAUSE}(\mathbf{John}, \mathrm{GO}(\mathbf{Mary}, \mathrm{UP}))$, the learner can now uniquely determine the word-to-meaning mapping given in (1).

In Robert van Rooy and Martin Stokhof, editors, *Proceedings of the Thirteen Amsterdam Colloquium*, December 17-19, 2001, pp. 126-131.

The aim of this paper is to study a generalization of Siskind's problem, where conceptual symbols, as well as variables, can be of arbitrary types, and meaning expressions are typed λ-terms built out of these, using λ-abstraction as well as application. This is the usual setup of logical semantics. For instance, the sentence *every man finds a unicorn* may have a meaning represented by $\forall x[\mathrm{MAN}(x) \to \exists y[\mathrm{UNICORN}(y) \wedge \mathrm{FIND}(x,y)]]$, which is just a shorthand for

$$\forall^{(e\to t)\to t}(\lambda x^e.\to^{t\to t\to t}(\mathrm{MAN}^{e\to t}(x)) \\ (\exists^{(e\to t)\to t}(\lambda y^e.\wedge^{t\to t\to t}(\mathrm{UNICORN}^{e\to t}(y))(\mathrm{FIND}^{e\to e\to t}(y)(x))))). \quad (2)$$

The meanings of the words in this sentence may be (suppressing the type when it is clear):

$$\begin{array}{lcl} \textit{every} & \mapsto & \lambda u^{e\to t}v^{e\to t}.\forall(\lambda x^e.\to(ux)(vx)) \\ \textit{man} & \mapsto & \mathrm{MAN} \\ \textit{finds} & \mapsto & \mathrm{FIND} \\ \textit{a} & \mapsto & \lambda u^{e\to t}v^{e\to t}.\exists(\lambda y^e.\wedge(uy)(vy)) \\ \textit{unicorn} & \mapsto & \mathrm{UNICORN} \end{array} \quad (3)$$

(I mostly follow the notational convention of Hindley 1997, except that I may write the application of two terms M and N either MN or $M(N)$, depending on which is more readable.)

Just as in the simple first-order case of Siskind, the problem of learning word-to-meaning mappings can be broken down into two subproblems: the problem of finding for each word the set of constants that appear in the meaning of that word, and the problem of building the correct λ-term for each word using the symbol set associated with that word. The first problem can be solved efficiently in exactly the same way as in the case of Siskind. How to solve the second problem was rather obvious in the case of Siskind, but becomes far from so in our generalized setting. I formulate the heart of the problem as a problem in typed λ-calculus.

Mapping Problem. Given m sets of distinct variables

$$\begin{array}{c} \vec{x}_1 = x_{1,1}^{\sigma_{1,1}},\ldots,x_{1,n_1}^{\sigma_{1,n_1}} \\ \vdots \\ \vec{x}_m = x_{m,1}^{\sigma_{m,1}},\ldots,x_{m,n_m}^{\sigma_{m,n_m}} \end{array}$$

and a λI-term $T^\rho[\vec{x}_1,\ldots,\vec{x}_m]$ (with free variables $\vec{x}_1,\ldots,\vec{x}_m$) in $\beta\eta$-normal form, find m λI-terms $S_1^{\tau_1}[\vec{x}_1],\ldots,S_m^{\tau_m}[\vec{x}_m]$ and a $BCI\lambda$-term $D^\rho[y_1^{\tau_1},\ldots,y_m^{\tau_m}]$, each in $\beta\eta$-normal form, such that

$$D[S_1[\vec{x}_1],\ldots,S_m[\vec{x}_m]] \rhd_{\beta\eta} T[\vec{x}_1,\ldots,\vec{x}_m].$$

I denote this problem by $T^\rho[\vec{x}_1;\ldots;\vec{x}_m]$, and I call $\langle S_1^{\tau_1}[\vec{x}_1],\ldots,S_m^{\tau_m}[\vec{x}_m]\rangle$ a *solution* to this problem.

In the solution to a mapping problem, $S_i^{\tau_i}[\vec{x}_i]$'s represent the 'meaning recipe' of individual words, where the free variables $\vec{x}_i$ are supposed to be filled by conceptual symbols. $D^\rho[y_1^{\tau_1},\ldots,y_m^{\tau_m}]$ then gives the meaning recipe for the sentence. Thus I

assume that simultaneous binding of multiple occurrences of a variable is possible in word meanings, but not in meaning recipes for sentences.[1]

Note that in the above formulation, the free variables $\vec{x}_1,\ldots,\vec{x}_m$ in a mapping problem $T^{\rho}[\vec{x}_1;\ldots;\vec{x}_m]$ are all distinct; in actual learning, the conceptual symbol sets of some words in a sentence may overlap. So the problem for the learner facing a single sentence-meaning pair corresponds, in general, to (a disjunction of) multiple mapping problems. Also, the learner may have already formed a hypothesis meaning for some of the words in the sentence currently processed, in which case an additional issue of combining two constraints arises (more on this below). For these reasons, the mapping problem as formulated above models only the 'heart' of the problem for the learner. Nevertheless, it is a good place to start formal analysis.

A 'mapping problem' of a more restricted kind corresponding to Siskind's setup always has at most one solution, and the condition under which a solution exists is easy to state. As for the general version, we have

Theorem 1. *Every mapping problem has infinitely many solutions.*

Theorem 2. *For any sequence of terms* $\vec{M} = M_1^{\sigma_1},\ldots,M_n^{\sigma_n}$, *let* $U_{\rho}[\vec{M}] = \lambda y^{\sigma_1\to\cdots\to\sigma_n\to\rho}.yM_1\ldots M_n$. *Then, for every mapping problem* $T^{\rho}[\vec{x}_1;\ldots;\vec{x}_m]$ *such that* $T^{\rho}[\vec{x}_1,\ldots,\vec{x}_m]$ *is a BCI*λ*-term,* $\langle U_{\rho}[\vec{x_1}],\ldots,U_{\rho}[\vec{x_m}]\rangle$ *solves* $T^{\rho}[\vec{x}_1;\ldots;\vec{x}_m]$.

We can also prove a slightly more complex form of Theorem 2 for λI mapping problems in general.

'Universal' solutions like $U_{\rho}[\vec{x}]$ are not very interesting: they not only derive correct sentence meanings but also many unwanted ones. The following example illustrates the case of a λI mapping problem.

Example 3. Consider a mapping problem $T^t[\vec{x}_1;\vec{x}_2]$, where $\vec{x}_1 = x_{1,1}^{(e\to t)\to t}, x_{1,2}^{t\to t\to t}, x_{1,3}^{e\to t}$, $\vec{x}_2 = x_{2,1}^{e\to t}$, and $T^t[\vec{x}_1,\vec{x}_2] = x_{1,1}(\lambda z^e.x_{1,2}(x_{1,3}z)(x_{2,1}z))$. This corresponds to the situation where the learner is faced with a sentence *everyone walks* coupled with the meaning:

$$\forall z(\mathrm{PERSON}(z)\to\mathrm{WALK}(z)),$$

having arrived at the word-to-symbol-set association:

$$\begin{array}{lcl} \textit{everyone} & : & \{\forall,\to,\mathrm{PERSON}\} \\ \textit{walks} & : & \{\mathrm{WALK}\} \end{array}$$

A solution $\langle U_t[\vec{x}_1,\mathsf{W}^{(e\to e\to t)\to e\to t}],\ x_{2,1}\rangle$, where $\mathsf{W}^{(e\to e\to t)\to e\to t} = \lambda y^{e\to e\to t}x^e.yxx$, corresponds to the word-to-meaning mapping:

$$\begin{array}{lcl} \textit{everyone} & \mapsto & \lambda y^{((e\to t)\to t)\to(t\to t\to t)\to(e\to t)\to((e\to e\to t)\to e\to t)\to t}.y(\forall)(\to)(\mathrm{PERSON})(\mathsf{W}) \\ \textit{walks} & \mapsto & \mathrm{WALK} \end{array}$$

$\langle U_t[\vec{x}_1,\mathsf{W}^{(e\to e\to t)\to e\to t}],\ x_{2,1}\rangle$ solves not only $T^t[\vec{x}_1;\vec{x}_2]$, but also $V^t[\vec{x}_1;\vec{x}_2] = x_{1,1}(\lambda z^e.x_{1,2}(x_{2,1}z)(x_{1,3}z))$, corresponding to

$$\forall z(\mathrm{WALK}(z)\to\mathrm{PERSON}(z)).$$

[1] One might want to restrict $D^{\rho}[y_1^{\tau_1},\ldots,y_m^{\tau_m}]$ even further to the so-called Lambek-van Benthem fragment (van Benthem 1991); this will complicate some results slightly.

Of course, $\langle \lambda v^{e\to t}.x_{1,1}(\lambda z^e.x_{1,2}(x_{1,3}z)(vz)),\ x_{2,1}\rangle$, corresponding to the word-to-meaning mapping:

$$\begin{array}{lcl} \textit{everyone} & \mapsto & \lambda v^{e\to t}.\forall z(\mathrm{PERSON}(z)\to vz) \\ \textit{walks} & \mapsto & \mathrm{WALK} \end{array}$$

only solves the former.

Certainly, the second solution in the above example is preferable to the first as a hypothesis for the learner. Perhaps what the learner should find is a 'minimal' solution, in the sense of solving as few unwanted problems as possible. To define a suitable sense of minimality, I introduce a 'definability relation' $\preceq$ between terms and a slightly weaker notion $\preceq_\rho$.

Definition 4. Let $M^\sigma[\vec{x}]$ and $N^\tau[\vec{x}]$ be two λ-terms in $\beta\eta$-normal form with the same set of free variables.

(i) We say that $N^\tau[\vec{x}]$ is *BCI-definable by* $M^\sigma[\vec{x}]$ and write $N^\tau[\vec{x}]\preceq M^\sigma[\vec{x}]$ if there is a closed $BCI\lambda$-term $P^{\sigma\to\tau}$ such that

$$P^{\sigma\to\tau}M^\sigma[\vec{x}]\rhd_{\beta\eta}N^\tau[\vec{x}].$$

We write $N^\tau[\vec{x}]\simeq M^\sigma[\vec{x}]$ if $N^\tau[\vec{x}]\preceq M^\sigma[\vec{x}]$ and $N^\tau[\vec{x}]\succeq M^\sigma[\vec{x}]$; and write $N^\tau[\vec{x}]\prec M^\sigma[\vec{x}]$ if $N^\tau[\vec{x}]\preceq M^\sigma[\vec{x}]$ but $N^\tau[\vec{x}]\not\simeq M^\sigma[\vec{x}]$.

(ii) We write $N^\tau[\vec{x}]\preceq_\rho M^\sigma[\vec{x}]$ if $\lambda y^{\tau\to\rho}.yN^\tau[\vec{x}]\preceq M^\sigma[\vec{x}]$. We write $N^\tau[\vec{x}]\simeq_\rho M^\sigma[\vec{x}]$ if $N^\tau[\vec{x}]\preceq_\rho M^\sigma[\vec{x}]$ and $N^\tau[\vec{x}]\succeq_\rho M^\sigma[\vec{x}]$; and write $N^\tau[\vec{x}]\prec_\rho M^\sigma[\vec{x}]$ if $N^\tau[\vec{x}]\preceq_\rho M^\sigma[\vec{x}]$ but $N^\tau[\vec{x}]\not\simeq_\rho M^\sigma[\vec{x}]$.

Clearly, $N^\tau[\vec{x}]\preceq M^\sigma[\vec{x}]$ implies $N^\tau[\vec{x}]\preceq_\rho M^\sigma[\vec{x}]$ for all ρ. Also, if τ 'ends in' ρ, $N^\tau[\vec{x}]\preceq_\rho M^\sigma[\vec{x}]$ implies $N^\tau[\vec{x}]\preceq M^\sigma[\vec{x}]$. The two relations do not coincide in general: $x^e\preceq_t\lambda y^{e\to t}.yx$, but $x^e\not\preceq\lambda y^{e\to t}.yx$. Both $\preceq$ and $\preceq_\rho$ are reflexive and transitive. Also, both relations are decidable. The latter follows from the known fact that every type is inhabited by finitely many $BCI\lambda$-terms in $\beta\eta$-normal form (van Benthem 1991).

Theorem 5. *If* $\langle S_1[\vec{x}_1],\ldots,S_m[\vec{x}_m]\rangle$ *solves* $T^\rho[\vec{x}_1;\ldots;\vec{x}_m]$, *and* $S_i[\vec{x}_i]\preceq_\rho P[\vec{x}_i]$, *then* $\langle S_1[\vec{x}_1],\ldots,S_{i-1}[\vec{x}_{i-1}],P[\vec{x}_i],S_{i+1}[\vec{x}_{i+1}],\ldots,S_m[\vec{x}_m]\rangle$ *solves* $T^\rho[\vec{x}_1;\ldots;\vec{x}_m]$.

Even though a mapping problem has infinitely many solutions, many of them turn out to be equivalent with respect to $\simeq$ and $\simeq_\rho$. A trivial example is $T^t[x^e;y^{e\to t}]=yx$, which has just two solutions modulo $\simeq$, namely $\langle x,\ y\rangle$ and $\langle\lambda y^{e\to t}.yx,\ y\rangle$ (which, in turn, are equivalent with respect to $\simeq_t$). I conjecture that every mapping problem has only finitely many solutions modulo $\simeq$.

The relations $\preceq$ and $\preceq_\rho$ can be naturally extended to solutions of mapping problems. In general, a mapping problem $T^\rho[\vec{x}_1;\ldots;\vec{x}_n]$ can have more than one $\preceq_\rho$-incompatible minimal solution.

Example 6. A problem $T^t[x^e;y^{e\to e\to t};z^e]=yzx$ has the following solutions, among others:

$$\langle\lambda y^{e\to e\to t}z^e.yzx,\ y,\ \lambda y^{e\to e\to t}v^{(e\to e\to t)\to e\to t}.v(\lambda xz.yzx)z\rangle,$$
$$\langle\lambda y^{e\to e\to t}v^{(e\to e\to t)\to e\to t}.vyx,\ y,\ \lambda y^{e\to e\to t}.yz\rangle.$$

Since $\lambda y^{e\to e\to t} z^e.yzx \succ_t \lambda y^{e\to e\to t} v^{(e\to e\to t)\to e\to t}.vyx$ and $\lambda y^{e\to e\to t} v^{(e\to e\to t)\to e\to t}.v(\lambda xz.yzx)z\rangle \prec_t \lambda y^{e\to e\to t}.yz$, the two solutions are incompatible with respect to $\preceq_t$. Their 'meet',

$$\langle \lambda y^{e\to e\to t} z^e.yzx,\ y,\ \lambda y^{e\to e\to t}.yz\rangle,$$

does not solve $T_t[x^e; y^{e\to e\to t}; z^e]$, and indeed the two solutions can be shown to be minimal with respect to $\preceq_t$. (Note that in this particular example, all solutions solve the same two problems, namely, $T^t[x^e; y^{e\to e\to t}; z^e]$ and $V^t[x^e; y^{e\to e\to t}; z^e] = yxz$. In the presence of an additional word-meaning recipe (e.g., $u^{e\to t}$), however, they generate different sets of sentence meanings, so they should not be treated as equivalent.)

The above example suggests that it may not be desirable for the purpose of learning to collect all minimal solutions to a given mapping problem. Below, I present a simple algorithm that finds a 'reasonably small' solution to a mapping problem.

In a term $\lambda x_1 \ldots x_n.yN_1 \ldots N_m$ in β-normal form, each N_i is called an *argument*. A *subargument* is either an argument or a subargument of an argument.

Algorithm A. Let a mapping problem $T^\rho[\vec{x}_1; \ldots; \vec{x}_m]$ be given. Associate with each $\vec{x}_i$ the smallest subargument P_i that contains all occurrences of variables in $\vec{x}_i$. Reorder $\vec{x}_1, \ldots, \vec{x}_m$ so that if P_i is a subargument of P_j, $i < j$. At the i-th cycle of the following iteration, a new variable u_i of a suitable type will be created. $V_i^\rho[\vec{x}_{i+1}, \ldots, \vec{x}_m]$ will contain as its free variables some (but not necessarily all) of $u_1, \ldots, u_i$, in addition to $\vec{x}_{i+1}, \ldots, \vec{x}_m$.

Let $V_0^\rho[\vec{x}_1, \ldots, \vec{x}_m] = T^\rho[\vec{x}_1, \ldots, \vec{x}_m]$.

For $i = 1, \ldots, m$, do the following:

- Find the smallest subargument $Q_i^\tau[\vec{x}_i, v_1, \ldots, v_k, y_1^{\tau_1}, \ldots, y_l^{\tau_l}]$ of $V_{i-1}^\rho[\vec{x}_i, \ldots, \vec{x}_m]$ that contains all occurrences of variables in $\vec{x}_i$, where $v_1, \ldots, v_k$ are among $u_1, \ldots, u_{i-1}$, and $y_1^{\tau_1}, \ldots, y_l^{\tau_l}$ are the free variables of Q_i^τ other than $\vec{x}_i, v_1, \ldots, v_k$.
- Let $S_i[\vec{x}_i] = \lambda v_1 \ldots v_k y_1^{\tau_1} \ldots y_l^{\tau_l}.M^\tau[\vec{x}_i, v_1, \ldots, v_k, y_1^{\tau_1}, \ldots, y_l^{\tau_l}]$.
- Replace Q_i^τ by $u_i^{\tau_1\to\cdots\to\tau_l\to\tau} y_1^{\tau_1} \ldots y_l^{\tau_l}$ in $V_{i-1}[\vec{x}_i, \ldots, \vec{x}_m]$, obtaining $V_i[\vec{x}_{i+1}, \ldots, \vec{x}_m]$.

When the above procedure is over, $\langle S_1[\vec{x}_1], \ldots, S_m[\vec{x}_m]\rangle$ is a solution to $T^\rho[\vec{x}_1; \ldots; \vec{x}_m]$.

Algorithm A is nondeterministic and different executions can lead to different results when $P_i = P_j$ for some $i \neq j$. This can only be so if the head variable of P_i is not among $\vec{x}_i$ for some i; otherwise the output of the algorithm is unique.

Example 7. Let us apply Algorithm A to the mapping problem corresponding to the following situation. The learner has arrived at the word-to-symbol-set association:

every	:	$\{\forall, \to\}$
man	:	$\{\text{MAN}\}$
finds	:	$\{\text{FIND}\}$
a	:	$\{\exists, \wedge\}$
unicorn	:	$\{\text{UNICORN}\}$

and is faced with the sentence *every man finds a unicorn*, coupled with the meaning (2). Algorithm A produces just one result in this case, and the solution corresponds to the following word-to-meaning mapping:

$$\begin{array}{lcl} \textit{every} & \mapsto & \lambda u^{e\to t} v^{e\to t}.\forall(\lambda x^{e}.\to(ux)(vx)) \\ \textit{man} & \mapsto & \text{MAN} \\ \textit{finds} & \mapsto & \text{FIND} \\ \textit{a} & \mapsto & \lambda u^{e\to t} w^{e\to e\to t} x^{e}.\exists(\lambda y^{e}.\wedge(uy)(wyx)) \\ \textit{unicorn} & \mapsto & \text{UNICORN} \end{array} \tag{4}$$

Note that the solution corresponding to (4) in the above example is smaller ($\prec_t$) than the solution corresponding to (3). The relative merit of (4) vis-a-vis (3) is debatable. It is possible to modify Algorithm A, so that it produces (3) rather than (4) on this example, but the resulting variant will be a more complicated algorithm.[2]

A mapping problem is supposed to model an individual stage in the second phase of the learning process, when the learner is processing a single sentence coupled with its meaning. In actual learning, some words in the current sentence may have already appeared in the preceding sentences during the second phase, so that the learner has meanings already hypothesized for them. When an algorithm like Algorithm A is applied to the mapping problem corresponding to the current sentence-meaning pair, it may produce a meaning for some old word that is $\preceq_t$-incompatible with the meaning already hypothesized. In such a case, the learner presumably needs to find an upper bound with respect to $\preceq_t$ of the two word meanings, assuming that the old hypothesis is not to be completely abandoned. An upper bound always exists ($U_t[\vec{x}]$ is one), but to formulate an algorithm for finding a reasonably small upper bound is not trivial. A question like this motivates investigation of the structure of the partial orders on equivalence classes of terms that are induced by the two relations $\preceq$ and $\preceq_\rho$. I have to leave this and other issues for future work.

References

Benthem, Johan van. 1991. *Language in Action: Categories, Lambdas and Dynamic Logic*. Amsterdam: North-Holland. (Second edition, 1995, Cambridge, Mass.: MIT Press.)

Hindley, J. Roger. 1997. *Basic Simple Type Theory*. Cambridge: Cambridge University Press.

Siskind, J. Mark. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition* **61**(1–2), 39–91. (Reprinted in Michael Brent, ed., 1996, *Computational Approaches to Language Acquisition*, pp. 39–91, Amsterdam: Elsevier.)

Siskind, J. Mark. 2000. Learning word-to-meaning mappings. In Peter Broeder and Jaap Murre, eds., *Models of Language Acquisition: Inductive and Deductive Approaches*, pp. 121–153. Oxford: Oxford University Press.

[2] Algorithm A (and its variant) can produce a counter-intuitive result when a duplicating β-contraction is involved in the reduction $D[S_1[\vec{x}_1],\ldots,S_n[\vec{x}_n]] \triangleright_{\beta\eta} T[\vec{x}_1,\ldots,\vec{x}_n]$, where $\langle S_1[\vec{x}_1],\ldots,S_n[\vec{x}_n]\rangle$ is the desired solution.