Learning to Disambiguate Natural Language Using World Knowledge

Antoine Bordes, Nicolas Usunier
LIP6, Université Paris 6, Paris, France
{bordes,usunier}@poleia.lip6.fr

Jason Weston,[*] Ronan Collobert
NEC Labs, Princeton, USA
{jasonw,collober}@nec-labs.com

Abstract

We present a general framework and learning algorithm for the task of *concept labeling*: each word in a given sentence has to be tagged with the unique physical entity (e.g. person, object or location) or abstract concept it refers to. Our method allows both *world knowledge* and *linguistic information* to be used during learning and prediction. We show experimentally that we can handle natural language and *learn* to use world knowledge to resolve *ambiguities* in language, such as word senses or coreference, without the use of *hand-crafted rules or features*.

1 Introduction

Much of the focus of the natural language processing community lies in solving syntactic or semantic tasks with the aid of sophisticated machine learning algorithms and the encoding of linguistic prior knowledge. For example, a typical way of encoding prior knowledge is to hand-code syntax-based input features or rules for a given task. However, one of the most important features of natural language is that its real-world use (as a tool for humans) is to communicate something about our physical reality or metaphysical considerations of that reality. This is strong prior knowledge that is simply ignored in most current systems.

For example, in current parsing systems there is no allowance for the ability to disambiguate a sentence given knowledge of the physical reality of the world. So, if one happened to know that Bill owned a telescope while John did not, then this should affect parsing decisions given the sentence "John saw Bill in the park with his telescope." Likewise, in terms of reference resolution one could disambiguate the sentence "He passed the exam." if one happens to know that Bill is taking an exam and John is not. Further, one can improve disambiguation of the word bank in "John went to the bank" if you happen to know whether John is out for a walk in the countryside or in the city. Many human disambiguation decisions are in fact based on whether the current sentence agrees well with one's current *world model*. Such a model is dynamic as the current *state of the world* (e.g. the existing entities and their relations) changes over time.

Concept labeling In this paper, we propose a general framework for learning to use *world knowledge* called *concept labeling*. The "knowledge" we consider is rudimentary and can be viewed as a database of physical entities existing in the world (e.g. people, locations or objects) as well as abstract concepts, and relations between them, e.g. the location of one entity can be expressed in terms of its relation with another entity. Our task thus consists of labeling each word of a sentence with its corresponding *concept* from the database.

The solution to this task does not provide a full semantic interpretation of a sentence, but we believe it is a first step towards that goal. Indeed, in many cases, the meaning of a sentence can only be uncovered after knowing exactly which concepts, e.g. which unique objects in the world, are

[*] Now at Google Labs, New York, USA.

involved. If one wants to interpret "He passed the exam", one has to infer not only that "He" refers to a "John", and "exam" to a school test, but also exactly which "John" and which test it was. In that sense, concept labeling is more general than the traditional tasks like word-sense disambiguation, co-reference resolution, and named-entity recognition, and can be seen as a unification of them.

Learning algorithm We then go on to propose a tractable algorithm for this task that seamlessly learns to integrate both world knowledge and linguistic content of a sentence *without the use of any hand-crafted rules or features*. This is a challenging goal and standard algorithms do not achieve it. Our algorithm is first evaluated on human generated sentences from RoboCup commentaries [2]. Yet this data set does not involve world knowledge. Hence we present experiments using a novel simulation procedure to generate natural language and concept label pairs: the simulation generates an evolving world, together with sentences describing the successive evolutions. These experiments demonstrate that our algorithm can learn to use world knowledge for word disambiguation and reference resolution when standard methods cannot.

Although clearly only a first step towards the goal of language understanding, which is AI complete, we feel our work is an original way of tackling an important and central problem. In a nut-shell, we show one can learn (rather than engineer) to resolve ambiguities using world knowledge, which is a prerequisite for further semantic analysis, e.g. for communication.

Previous Work Our work concerns learning the connection between two symbolic systems: the one of natural language and the one, non-linguistic, of the concepts present in a database. Making such an association has been studied as the *symbol grounding problem* [8] in the literature. More specifically, the problem of connecting natural language to another symbolic system is called *grounded language processing* [11]. Some of such earliest works involved hand-coded parsing and no learning at all, perhaps the most famous being situated in blocks world [15]. More recent works on grounded language acquisition have focused on *learning* to match language with some other representations. Grounding text with a visual representation also in a blocks-type world was tackled in [6]. Other works use visual grounding [13, 17, 1], or a representation of the intended meaning in some formal language [18, 7, 9, 16, 2]. Example applications of such grounding include using the multimodal input to improve clustering (with respect to unimodal input) (e.g. [13]), word-sense disambiguation [1, 7] or semantic parsing [18, 9, 16, 2].

2 Concept Labeling

We consider the following setup. One must learn a mapping from a natural language sentence $x \in \mathcal{X}$ to its labeling in terms of *concepts* $y \in \mathcal{Y}$, where y is an ordered set of concepts, one concept for each word in the sentence, i.e. $y = (c_1, \ldots, c_{|x|})$ where $c_i \in \mathcal{C}$, the set of concepts. These concepts belong to a current model of the world which expresses one's knowledge of it. We term it a "*universe*". The framework of concept labeling is illustrated in Figure 1.

Universe We define the universe as a set of concepts and their relation to other concepts: $\mathcal{U} = (\mathcal{C}, \mathcal{R}_1, \ldots, \mathcal{R}_n)$ where n is the number of types of relation and $\mathcal{R}_i \subset \mathcal{C}^2$, $\forall i = 1, \ldots, n$. Much work has been done on knowledge representation itself (see [12] for an introduction). This is not the focus of this paper and so we made the simplest possible choice.

The *universe* we consider is in fact nothing more than a relational database, where records correspond to concepts and each kind of interaction between concepts is a relation table. To make things concrete we now describe the template database we use in this paper.

1. Each concept c of the database is identified using a unique string *name*(c). Each physical object or action (verb) of the universe has its own referent. For example, two different cartons of milk will be referred to as <*milk1*> and <*milk2*>[1].
2. We consider two relation tables[2] that can be expressed with the following formula:
   - *location*$(c) = c'$ with $c, c' \in \mathcal{C}$: the location of the concept c is the concept c'.
   - *containedby*$(c) = c'$ with $c, c' \in \mathcal{C}$: the concept c' physically holds the concept c.

[1] Here, we use understandable strings as identifiers for clarity but they have no meaning for the system.

[2] Of course this list is easily expanded upon. Here, we give two simple properties of physical objects.

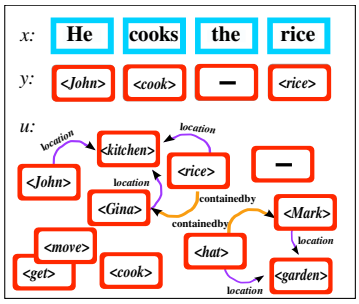

Figure 1: **Concept labeling.** The universe u contains all the known concepts that exist, and their relations. The goal is to predict the sequence y of the concepts that each word in the sentence x refers to, including the empty concept "-".

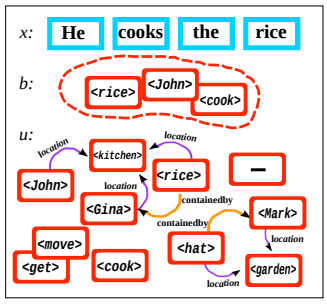

Figure 2: **Training triple.** In our setting, one trains under weak supervision. The supervision bag b consists of the set of concepts related to the sentence x. The alignment between words and concepts is not given and must be discovered.

What is this useful for? Our overall goal is to construct a semantic representation of a sentence which can be used to build and modify the underlying world model. Our approach can be applied when a current model of the environment (equivalent to our *universe*) is available. A immediate realistic setting is within a computer game environment, e.g. multiplayer Internet games. Real-world settings are also possible but require, for example, technologies for building world knowledge beyond the scope of this work.

Concept labeling on its own is necessary but not sufficient to provide a complete semantic representation; but adding a grammar as it is done in semantic parsing [18, 9, 16, 2], can provide promising semantic representations[3]. Hence, as a first step, simply adding semantic role labeling [10] would allow to know both the predicate concepts and the roles of other concepts with respect to those predicates. For example, "He cooks the rice" from Figure 1 would be labeled with "He/ARG1 cooks/REL the/- rice/ARG2" as well as with the concepts y.

Our system would then have the potential to disambiguate examples such as the following: *"John went to the kitchen and Mark stayed in the living room. He cooked the rice and served dinner."* The world knowledge that John is in the kitchen comes from the semantic representation predicted from the first sentence. This is used to resolve the pronoun "he" using further background knowledge that cooking is done in the kitchen. All of this inference could be *learnt from examples*.

Weak supervision Usually in sequence labeling, learning is carried out using fully supervised data composed of input sequences explicitly aligned with label annotations. To learn the task of concept labeling, this would result in training examples composed by triples $\{x, y, u\} \in \mathcal{X} \times \mathcal{Y} \times \mathcal{U}$ such as the one displayed on Figure 1.

Yet, in our case, we want our model to be able to learn from weakly supervised data of just observing language given the evolving world-state context. Hence, in this paper, we consider weakly labeled training triples $\{x, b, u\}$: for a given input sentence x, the supervision b is a "bag" (set) of labels of size $|x|$ (there is no information about ordering/alignment of elements of b to x). This is a realistic setting and similar to the setting of [9] except we learn to use world knowledge. An illustrating example of a training triple (x, b, u) is given on Figure 2. Hence, our algorithm *trains using weakly supervised triples*, such as the one depicted on the right hand side.

Why is this task challenging? The main difficulty of concept labeling arises with *ambiguous words* that can be mislabeled. A concept labeling algorithm must be able to use the available information to solve the ambiguities. In our work, we consider the following kinds of ambiguities, that can be mixed within a sentence:

[3]Contrary to our work, current semantic parsers do not take into account any world knowledge and thus can not interact with their environment.

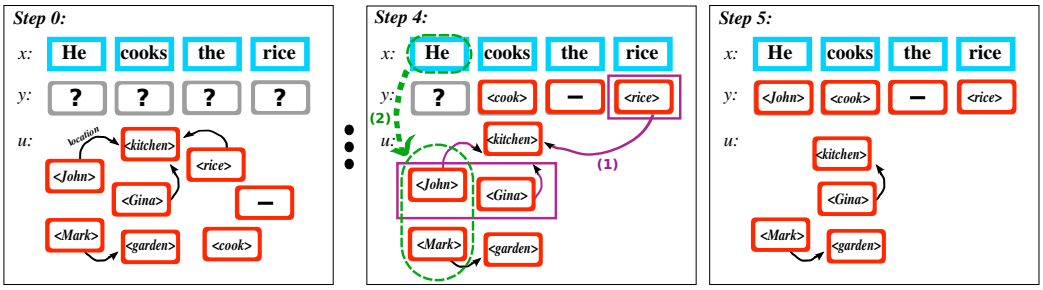

Figure 3: **Inference scheme.** Step 0 defines the task: find the concepts y given a sentence x and the current state of the *universe* u. For simplicity only relevant concepts and location relations are depicted. First, non-ambiguous words are labeled in steps 1-3 (not shown). In step 4, to tag the ambiguous pronoun "he", the system has to combine two pieces of information: (1) $<$*rice*$>$ and the unknown concept might share the same location, $<$*kitchen*$>$, and, (2) "he" only refers to a subset of concepts in u (the males).

- **Location-based** ambiguities that can be resolved by the locations of the concepts. Examples: "The father picked *it* up" or "*He* got the coat in the hall". Information about the location of the father, co-located objects and so on can improve the disambiguation.
- **Containedby-based** ambiguities that can be resolved through knowledge of *containedby* relations as in "the *milk* in the closet" or "the *one* in the closet" where there are several cartons of milk (e.g. one in the fridge and one in the closet).
- **Category-based:** A concept is identified in a sentence by an ambiguous term and the disambiguation can be resolved by using semantic categorization. Example: "*He* cooks the rice in the kitchen" where two persons, a male and a female, are in the kitchen.

The first two kinds of ambiguities require the algorithm to be able to *learn* rules based on its available universe knowledge. The last kind can be solved using linguistic information such as word gender or category. However, the necessary rules or linguistic information are *not* given as input features and again the algorithm has to *learn* to infer them. This is one of the main goals of our work.

Figure 3 describes the necessary steps for an algorithm to perform such disambiguation. Even for a simple sentence the procedure is rather complex and somehow requires "reasoning".

3 Learning Algorithm

Inference A straight-forward approach one could adopt to learn a function that maps from sentence x to concept sequence y given u is to consider a model of the form:

$$y = f(x, u) = \text{argmax}_{y'} \; g(x, y', u), \tag{1}$$

where $g(\cdot)$ returns a scalar that should be a large value when the output concepts y' are consistent with both the sentence x and the current state of the universe u. To find such a function, one can choose a family of functions $g(\cdot)$ and pick the member which minimizes the error: $L = \sum \ell(b, f(x, u))$ where the loss function ℓ is 1 if b and the set containing the elements of $f(x, u)$ differ, and 0 otherwise. However, one practical issue of this choice of algorithm is that the exhaustive search over all possible concepts in equation (1) could be rather slow.

The intuition in Figure 3 rather suggests us to use a greedy "order-free" inference process: we label the word we are most confident in (possibly the least ambiguous, which can be in any position in the sentence) and then use the known features of that concept to help label the remaining ones.

This is detailed in Algorithm 1. At each step, the set of output candidates S_t does not depend on the position in the sentence, one can label any thus far unlabeled word with a concept (line 3). The confidence-based choice is carried out with the function $g(\cdot)$ (line 4), which is learnt to optimize the loss of interest L. Note that Algorithm 1 is quite efficient as it requires only $|\mathcal{C}| \times |x|^2$ computations of $g(\cdot)$, whereas solving (1) requires $|\mathcal{C}|^{|x|}$ (and $|\mathcal{C}| \gg |x|$).

Algorithm 1 Order-free inference for a given input (x, u)

1: Start with predictions $\hat{y}_j^0 = \perp$, $j = 1, \ldots, |x|$, where $\perp$ means *unlabeled*.
2: **while** $t < |x|$ **do**
3: Increment $t = t + 1$.
4: Define S_t, the set of output candidates : $S_t = \bigcup_{j:\, \hat{y}_j^{t-1} = \perp} \{y' \,|y'_j \in \mathcal{C} \text{ and } \forall i \neq j, y'_i = \hat{y}_i^{t-1}\}$.
5: Label greedily the concept with the highest score: $\hat{y}^t = \text{argmax}_{y' \in S_t} \; g(x, y', u)$.
6: **end while**
7: Output the predicted sequence of concepts $\hat{y}^{|x|}$.

Family of functions Following [4], we chose a neural-network architecture for $g(\cdot)$ because we expected the network to encode some of the linguistic information required to perform disambiguation (such as, "he" only refers to males, in the example of Figure 3). The actual form of $g(\cdot)$ is:

$$g(x, y, u) = \sum_{i=1}^{|x|} g_i(x, y_{-i}, u)^\top h(y_i, u) \tag{2}$$

where $g_i(\cdot) \in \mathbb{R}^N$ is a "sliding window" representation of width w centered on the i^{th} position in the sentence, y_{-i} is the same as y except that the i^{th} position $(y_{-i})_i = \perp$, and $h(\cdot) \in \mathbb{R}^N$ is a linear mapping into the same space as $g(\cdot)$. We constrain $||h(\perp, u)|| = 0$ so that as yet unlabeled outputs do not play a role. A less mathematical explanation of this model is as follows: $g_i(\cdot)$ takes a window of the input sentence *and* previously labeled concepts centered around the i^{th} word and embeds them into an N dimensional space. $h(y_i, u)$ embeds the i^{th} concept into the same space, where both mappings are *learnt*. The magnitude of their dot product in this space indicates how confident the model is that the i^{th} word, given its context, should be labeled with concept y_i. This representation is also useful from a computational point of view because $g_i(\cdot)$ and $h(\cdot)$ can be cached and reused in Algorithm 1, making inference fast.

$g_i(\cdot)$ and $h(\cdot)$ are simple two-layer linear neural networks in a similar spirit to [4]. The first layer of both are so-called "Lookup Tables". We represent each word $\mathcal{W}$ in the dictionary with a unique vector $D(\mathcal{W}) \in \mathbb{R}^d$ and every unique concept name *name*(c) also with a unique vector $C(\textit{name}(c)) \in \mathbb{R}^d$, where we *learn these mappings*. No hand-crafted syntactic features are used.

To represent a concept *and* its relations we do something slightly more complicated. A particular concept c (e.g. an object in a particular location, or being held by a particular person) is expressed as the concatenation of the three unique concept name vectors:

$$\bar{C}(c) = (C(\textit{name}(c)), C(\textit{name}(\textit{location}(c))), C(\textit{name}(\textit{containedby}(c)))). \tag{3}$$

In this way, the learning algorithm can take these *dynamic* relations into account, if they are relevant for the labeling task. Hence, the first layer of the network $g_i(\cdot)$ outputs[4]:

$$g_i^1(x, y_{-i}, u) = \left(D(x_{i-\frac{w-1}{2}}), \ldots, D(x_{i+\frac{w-1}{2}}), \bar{C}((y_{-i})_{i-\frac{w-1}{2}}), \ldots, \bar{C}((y_{-i})_{i+\frac{w-1}{2}}) \right) .$$

The second layer is a linear layer that maps from this $4wd$ dimensional vector to the N dimensional output, i.e. overall we have the function (with $W_g \in \Re^{4wd \times N}$ and $b_g \in \Re^N$) :

$$g_i(x, y_{-i}, u) = W_g \; g_i^1(x, y_{-i}, u) + b_g.$$

Likewise, $h(y_i, u)$ has a first layer which outputs $\bar{C}(y_i)$, followed by a linear layer mapping from this $3d$ dimensional vector to N, i.e. (with $W_h \in \Re^{3d \times N}$ and $b_h \in \Re^N$)

$$h(y_i, u) = W_h \; \bar{C}(y_i) + b_h.$$

Overall, we chose a linear architecture that avoids strongly engineered features, assumes little prior knowledge about the mapping task in hand, but is powerful enough to capture many kinds of relations between words and concepts.

[4]Padding must be used when indices are out of bounds.

LaSO-type training We train our system online by employing a variation on the LaSO (Learning As Search Optimization) training process [5]. LaSO's central idea is to mix training and inference in a single *learning for search* strategy. During training, we thus perform inference on each given triple (x, b, u) using a version of Algorithm 1 modified in two ways.

The first modification is the addition of an update step of the model parameters. At each round, we define the predicted labeling $\hat{y}^t$ as y-good, compared to the supervision bag b, if either $\hat{y}_i^t \in b$ or $\hat{y}_i^t = \perp$ for all i (all the words are either unlabeled or labeled with an element of bag). As soon as $\hat{y}^t$ is no longer y-good, the model is updated with a stochastic gradient step, similar to the "early update" of [3], to satisfy the ranking constraints:

$$\forall i : \hat{y}_i^{t-1} = \perp \ , \ \forall c \in b, \quad g(x, \hat{y}_{+(i,c)}^{t-1}, u) > g(x, \hat{y}^t, u) \tag{4}$$

where $\hat{y}_{+(i,c)}^{t-1}$ is a vector which is the same as $\hat{y}^{t-1}$ except its i^{th} element is set to the concept c. Intuitively, if a label prediction for the word x_j in position j does not belong to the bag b then we require any prediction that *does* belong to it to be ranked above this incorrect prediction. When all such constraints are satisfied, the correct bags are predicted.

The second modification concerns the management of the supervision bag b: after a concept c from b has been picked, we remove it from the bag. Otherwise the same element could be picked again and again, resulting in an output sequence that does not violate the constraints (4).

Why does this work? Consider again the example "He cooks the rice" in Figure 3. We cannot resolve the first word in the sentence "He" with the true concept label <*John*> until we know that "rice" corresponds to the concept <*rice*> which we know is located in the kitchen, as is John, thereby making him the most likely referent. This is why we choose to label words with concepts in an order independent of the position in the sentence in Algorithm 1 because simply label from left to right does not work. The algorithm has to learn which word to label first, and presumably, it labels the least ambiguous words first. This is what we have observed experimentally. Once <*rice*> has been identified, its features including its location will influence the function $g(x, y, u)$ and the word "He" is more easily disambiguated. Simultaneously, our method must learn the N dimensional representations $g_i(\cdot)$ and $h(\cdot)$ such that "He" matches with <*John*> rather than <*Gina*>, i.e. equation (2) is a larger value. This should happen because during training <*John*> and "He" often co-occur. This concludes the disambiguation.

Note that our system can learn the general principle that two things that are in the same place are more likely to be referred to in the same sentence, and does not have to re-learn that for all possible places and things. In general, our model can resolve many kinds of ambiguities, both from syntax, semantics, or a combination, for example all the cases given in Section 2. Furthermore, the LaSO-type training based on the constraints (4) can implicitly learn the alignment between words and concepts, even though it is never given.

4 Experiments

4.1 RoboCup Commentaries

Before using any world knowledge, we wanted to assess that our algorithm was able to be trained under weak supervision on natural language. Hence we tested it on the RoboCup commentary dataset[5]. This data contains human commentaries on football simulations over four games labeled with semantic descriptions of actions: passes, offside, penalties, ... along with the players involved. We treated each semantic description as a "bag" of concepts for weak supervision. We trained on one match and tested on the other three, averaging over all four possible splits.

As in [2], we report the "matching" score: each input commentary is associated with several actions, we measure how often the system retrieve the correct one. We tackled this in two successive steps: (1) a bag of labels is predicted using Algorithm 1, (2) we choose to match to the bag from the set of actions that has the highest cosine similarity with the prediction. Our system achieves an F1 score of 0.669. Previously reported methods from [2] Krisper (0.645 F1) and Wasper-Gen (0.65 F1) achieve similar results (random matching yields 0.465 F1). Results on this benchmark clearly indicate that our method can handle both weak supervision and natural language sentences.

[5] see [2] or `http://www.cs.utexas.edu/~ml/clamp/sportscasting/#data` for details.

Method	Supervision	Features	Train Err	Test Err
SVM_{struct}	strong	$x + u$ (*loc*, *contain*)	18.68%	23.57%
NN_{LR}	strong	$x + u$ (*loc*, *contain*)	5.42%	5.75%
NN_{OF}	strong	x	32.50%	35.87%
NN_{OF}	strong	$x + u$ (*loc*, *contain*)	**0.0%**	**0.11%**
NN_{OF}	weak	$x + u$ (*loc*, *contain*)	**0.64%**	**0.72%**

Table 1: **Simulation results.** We compare our order-free neural network using world knowledge and weak supervision (line 5) to *strongly* supervised variants (line 1-4). A strong supervision provides training examples in which words and concepts are aligned.

4.2 Simulated World

Simulation RoboCup sentences do not involve any lexical ambiguity. To evaluate the ability of our method to use world knowledge to perform disambiguation, we thus created a simulation.

To conduct experiments on an environment with a reasonably large size we built the following artificial universe designed to simulate a house interior. It contains 58 concepts: 15 verbs (<*move*>, <*get*>, <*give*>,...) along with 10 actors (<*John*>, <*dog*>,...), 15 small objects (<*water*>, <*chocolate*>, <*doll*>,...), 6 rooms (<*kitchen*>,...) and 12 pieces of furniture (<*couch*>, ...). We define the set of describing words for each concept to contain at least two terms: an ambiguous one (using a pronoun) and a unique one. 75 words are used for generating sentences $x \in \mathcal{X}$. The simulation generates actions in the world along with sentences that describe them using a simple grammar. For example a simulation step could produce the results:

1. The event <*move*>(<*Gina*>, <*hall*>) is picked.
2. Generate the training sample (x, b, u) = ("she goes from the bedroom to the hall", {<*hall*>,<*Gina*>,<*bedroom*>,<*move*>}, u).
3. Modify (update) u with *location*(<*Gina*>) = <*hall*>.

The simulation code can generate sentences like "he sits on the chair", "she goes from the bedroom to the kitchen" or "the brother gives it to his sister". For our experiments we record 50,000 triples (x, y, u) for training and 20,000 for testing. Around 55% of sentences contain lexical ambiguities.

Algorithms We compare several models. Firstly, we evaluate our "order-free" neural network based algorithm presented in Section 3 (NN_{OF} using x + u) trained with two kinds of supervision: our realistic weak setting using a bag and the *strong* setting for which words-concepts alignments are given. We also train a model with strong supervision but no access to the *universe* (NN_{OF} using x). The models *with* world knowledge have access to the *location* and *containedby* features of all concepts in the universe. For the model *without* world knowledge we remove the $C(\textit{name}(\textit{location}(c)))$ and $C(\textit{name}(\textit{containedby}(c)))$ features from the concept representation in equation (3) and are left with a pure tagging task, no different to tasks like named entity recognition.

Finally, we compare our weakly trained model to two more strongly supervised methods: a greedy left-to-right labeling NN (NN_{LR}) and a structured output SVM [14] (SVM_{struct}). For the SVM, the features from the world model are used as additional input features and Viterbi is used to decode the outputs. Only a linear model was used due to the infeasibility of training non-linear ones (all the NNs are linear). In all experiments we used word and concept dimension $d = 20$, $g(\cdot)$ and $h(\cdot)$ have dimension $N = 200$, a sliding window width of $w = 13$ (i.e., 6 words on either side of a central word), and we chose the learning rate that minimized the training error as described in Section 3.

Results The results are given in Table 1. The error rates express the proportion of predicted sequences with *at least one* incorrect tag. Our model (NN_{OF}) learns to use world knowledge to disambiguate on this task: it obtains a test error close to 0% with this knowledge, and around 35% error without. It is worth noting that training under weak supervision (line 5) does not degrade accuracy: the same model using the less realistic strong setting (line 4) is only slightly better.

Confirming our intuitions about the inference (as in Figure 3), the comparison with other algorithms highlights the following points: (i) order-free labeling of concepts is important compared to more

restricted labeling schemes such as left-right labeling (NN_{LR}); (ii) the architecture of our NN which embeds concepts is able to capture some useful linguistic information and thus helps generalization; this should be compared to SVM_{struct} which does not perform as well. Note that a nonlinear SVM *or* a linear SVM with hand-crafted features are likely to perform better, but the former is too slow and the latter is what we are trying to avoid as such methods are brittle.

We constructed our simulation such that all ambiguities could be resolved with world knowledge, which is why we can obtain almost 0%: this is a good sanity check showing that our method is working well. We believe it is a prerequisite that we do well here if we hope to do well on harder tasks. The simulation we built uses rules to generate actions and utterances, but our learning algorithm uses no such hand-built rules but instead successfully *learns* them. This flexibility is the key to success in real tasks, where brittle engineering approaches have been tried with moderate success.

Finally, if the amount of training data is reduced we can still perform well. With 5000 training examples for NN_{OF} ($x + u(loc, \textit{contain})$) with the same parameters we obtain 3.1% test error.

5 Conclusion and Future Work

We have described a *general* framework for language grounding based on the task of *concept labeling*. The learning algorithm we propose is *scalable* and *flexible*: it learns with only weakly supervised raw data, and no prior knowledge of how concepts in the world are expressed in natural language. We have tested our framework within a simulation, showing that it is possible to *learn* (rather than engineer) to resolve ambiguities using world knowledge. We also showed we can learn with real *human annotated* data (RoboCup commentaries). Although clearly only a first step towards the goal of language understanding we feel our work is an original way of tackling an important and central problem. The most direct application of our work is within computer games, but other communication tasks could also apply with an increased effort.

References

[1] K. Barnard and M. Johnson. Word Sense Disambiguation with Pictures. *Artificial Intelligence*, 167(1-2):13–30, 2005.

[2] D. Chen and R. Mooney. Learning to Sportscast: A Test of Grounded Language Acquisition. In *ICML '08*.

[3] M. Collins and B. Roark. Incremental parsing with the perceptron algorithm. In *ACL '04*.

[4] R. Collobert and J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *ICML '08*.

[5] H. Daumé III and D. Marcu. Learning as Search Optimization: Approximate Large Margin Methods for Structured Prediction. In *ICML '05*.

[6] J. Feldman, G. Lakoff, D. Bailey, S. Narayanan, T. Regier, and A. Stolcke. L 0-The first five years of an automated language acquisition project. *Artificial Intelligence Review*, 10(1):103–129, 1996.

[7] M. Fleischman and D. Roy. Intentional Context in Situated Language Learning. In *CoNLL '05*.

[8] S. Harnad. The Symbol Grounding Problem. *Physica D*, 42(1-3):335–346, 1990.

[9] R. Kate and R. Mooney. Learning Language Semantics from Ambiguous Supervision. In *AAAI '07*.

[10] P. Kingsbury and M. Palmer. From Treebank to PropBank. In *ICLRE '02*.

[11] D. Roy and E. Reiter. Connecting Language to the World. *Artificial Intelligence*, 167(1-2):1–12, 2005.

[12] S. Russell, P. Norvig, J. Canny, J. Malik, and D. Edwards. *Artificial intelligence: a modern approach*. Prentice Hall Englewood Cliffs, NJ, 1995.

[13] J. Siskind. Grounding Language in Perception. *Artificial Intelligence Review*, 8(5):371–391, 1994.

[14] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large Margin Methods for Structured and Interdependent Output Variables. *JMLR*, 6:1453–1484, 2005.

[15] T. Winograd, M. Barbour, and C. Stocking. *Understanding natural language*. Academic Press NY, 1972.

[16] Y. Wong and R. Mooney. Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus. In *ACL '07*.

[17] C. Yu and D. Ballard. On the Integration of Grounding Language and Learning Objects. In *AAAI '04*.

[18] L. Zettlemoyer and M. Collins. Learning to Map sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *UAI '05*.