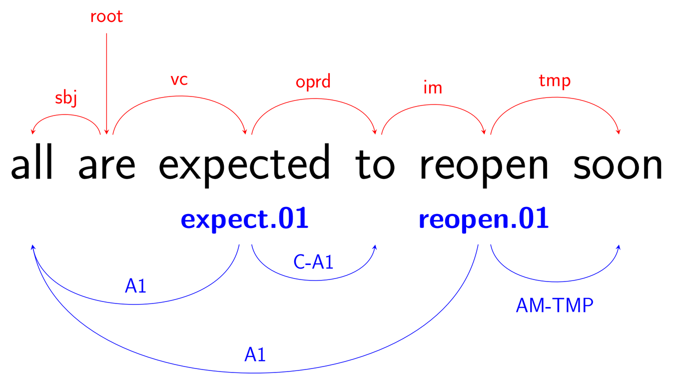
Multi-Task Learning for Incremental Parsing using Stack LSTMs

Swabha Swayamdipta[1] Miguel Ballesteros[2] Chris Dyer[3] Noah A. Smith[4]

[1]Carnegie Mellon University, USA [2]Universitat Pompeu Fabra, Spain [3]Google DeepMind, UK [4]University of Washington, USA

Joint dependency syntactic parsing and semantic role labeling

Given a sentence, the task is to simultaneously learn the syntactic dependency tree and the semantic role labeling graph underlying the sentence.



Incremental parsing algorithm

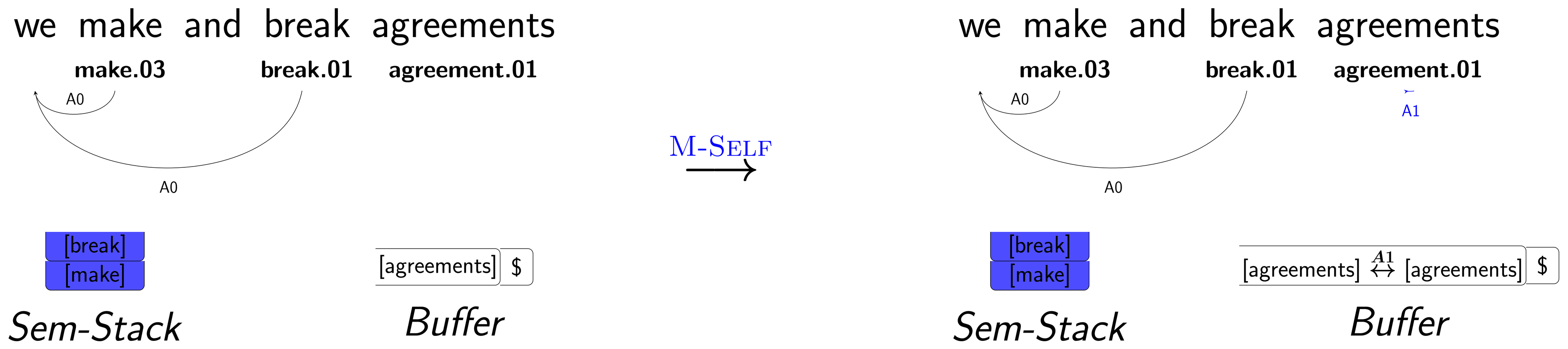
We propose an incremental algorithm to predict the entire joint parse one edge at a time. Our algorithm converts the joint parse structure into a sequence of actions, to be executed on some data structures - a Syntactic Stack(S), a Semantic Stack(M) and an input Buffer(B).

The algorithm makes a single pass through the sentence from left to right, moving words and partial parses between the data structures, in linear time (Henderson et. al, 08).

Syntactic and Semantic Actions Set

- S-SHIFT, S-REDUCE, S-LEFT, S-RIGHT
- M-SHIFT, M-REDUCE, M-LEFT, M-RIGHT, M-SELF, M-SWAP, M-PRED

An example transition (M-SELF)



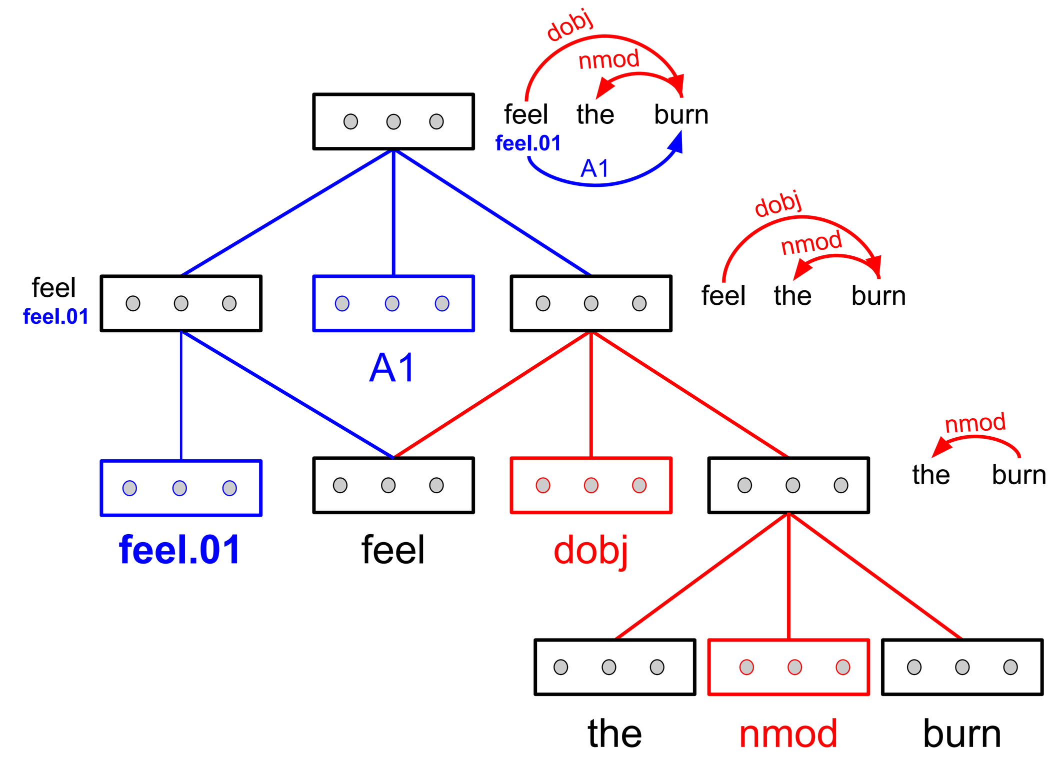
[x] denotes a partial substructure.

Example run

Action	S	M	B	Edge
	[]	[]	[all, are, expected, to, reopen, soon, $]	—
S-SHIFT	[all]	[]	[all, are, expected, to, reopen, soon, $]	—
M-SHIFT	[all]	[all]	[are, expected, to, reopen, soon, $]	—
S-LEFT(*sbj*)	[]	[all]	[are, expected, to, reopen, soon, $]	all $\xleftarrow{\text{sbj}}$ are
S-SHIFT	[are]	[all]	[are, expected, to, reopen, soon, $]	—
M-SHIFT	[are]	[all, are]	[expected, to, reopen, soon, $]	—
S-RIGHT(*vc*)	[are, expected]	[all, are]	[expected, to, reopen, soon, $]	are $\xrightarrow{\text{vc}}$ expected
M-PRED(**expect.01**)	[are, expected]	[all, are]	[expected, to, reopen, soon, $]	—
M-REDUCE	[are, expected]	[all]	[expected, to, reopen, soon, $]	—
M-LEFT(*A1*)	[are, expected]	[all]	[expected, to, reopen, soon, $]	all $\xleftarrow{\text{A1}}$ expect.01
M-SHIFT	[are, expected]	[all, expected]	[to, reopen, soon, $]	—
S-RIGHT(oprd)	[are, expected, to]	[all, expected]	[to, reopen, soon, $]	expected $\xrightarrow{\text{oprd}}$ to
M-RIGHT(*C-A1*)	[are, expected, to]	[all, expected]	[to, reopen, soon, $]	expect.01 $\xrightarrow{\text{C-A1}}$ to
	⋮			
S-REDUCE	[are]	[soon]	[$]	—
S-LEFT(*root*)	[]	[soon]	[$]	are $\xleftarrow{\text{root}}$ $
S-SHIFT	[$]	[soon]	[$]	—
M-REDUCE	[$]	[]	[$]	—
M-SHIFT	[$]	[$]	[]	—

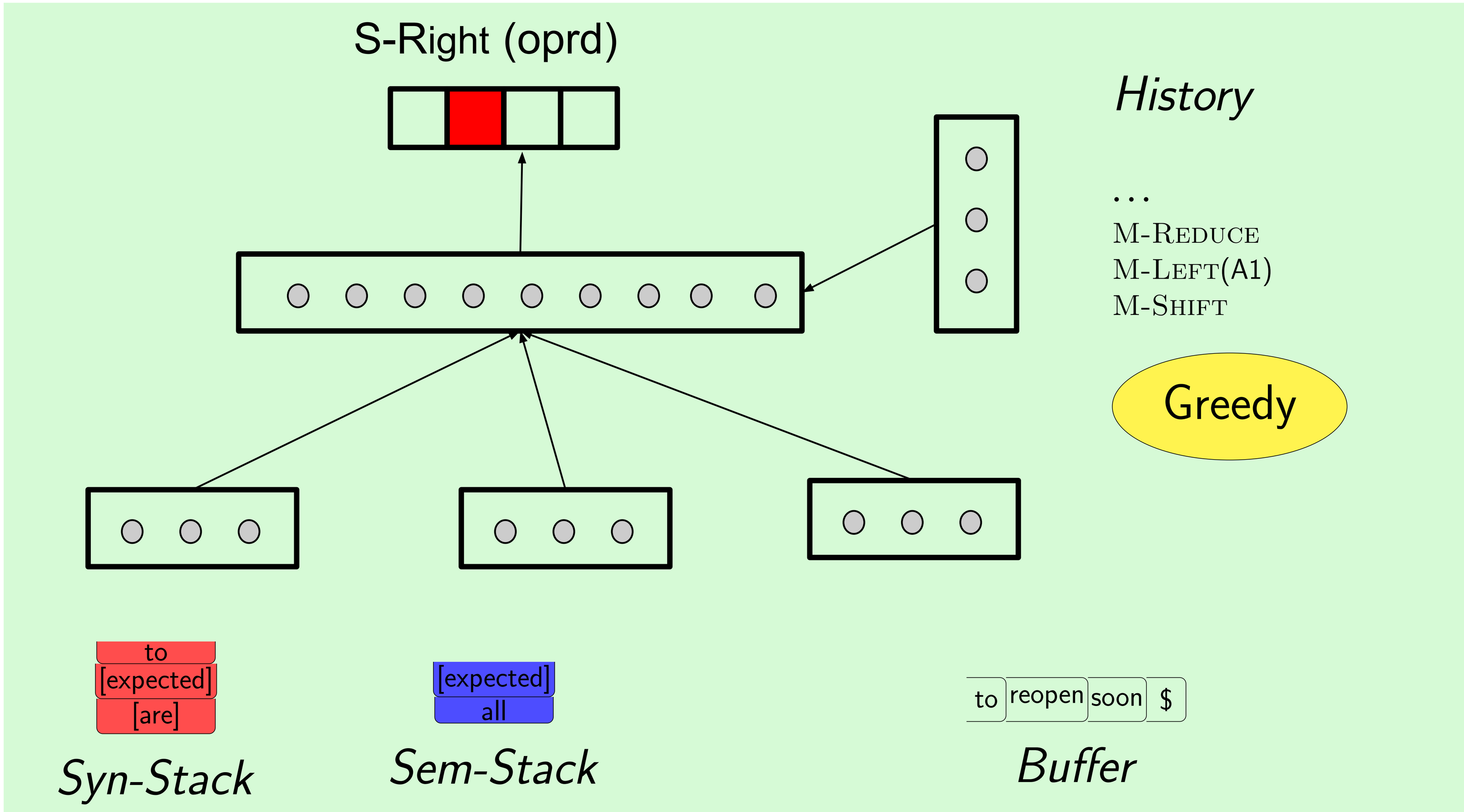
Representing partial substructures

Our data structures may contain atomic words of the sentence as well as partial parse substructures; we use distributed vector representations for both. Our model uses **recursive neural nets** for composing a new edge with a partial parse.



Stack LSTM model

We propose a model which learns representations for each of our data structures - the stacks, the buffer and the history of actions, using **stack LSTMs**. Stack LSTMs are LSTMs equipped with *push*, *pop* and *summary* operations (Dyer et. al, 15).



The action selected at timestep t is

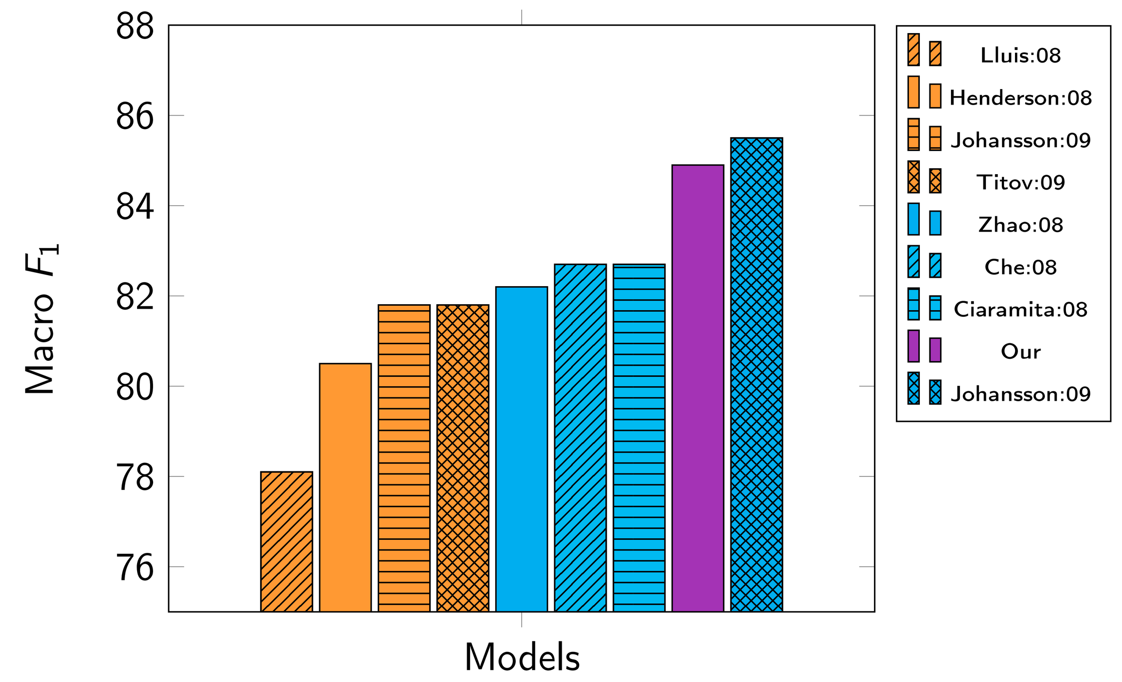
$$\tau^* = \arg\max_{\tau \in \mathcal{A}_t} \text{score}(\tau; S_t, M_t, B_t, A_t)$$

score is calculated by a nonlinear combination of the stack LSTMs. Only allowed transitions, $\mathcal{A}_t$ are considered in the decision rule.

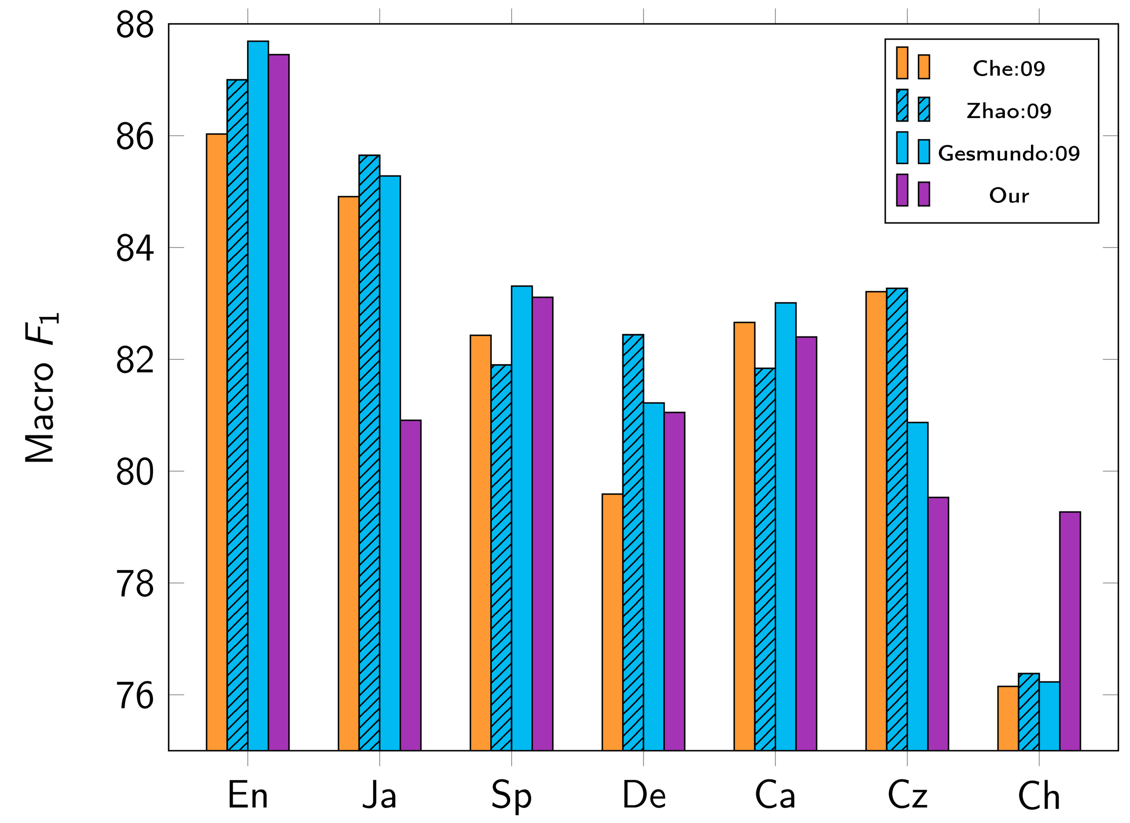
To learn the model parameters, we maximize the log likelihood, using stochastic gradient descent, q and θ being model parameters and y_t being a non-linear combination of the stack LSTMs.

$$\log \frac{\exp(\mathrm{q}_{\tau_t} + \theta_{\tau_t} \cdot \mathrm{y}_t)}{\sum_{\tau' \in \mathcal{A}_t} \exp(\mathrm{q}_{\tau'} + \theta_{\tau'} \cdot \mathrm{y}_t)}$$

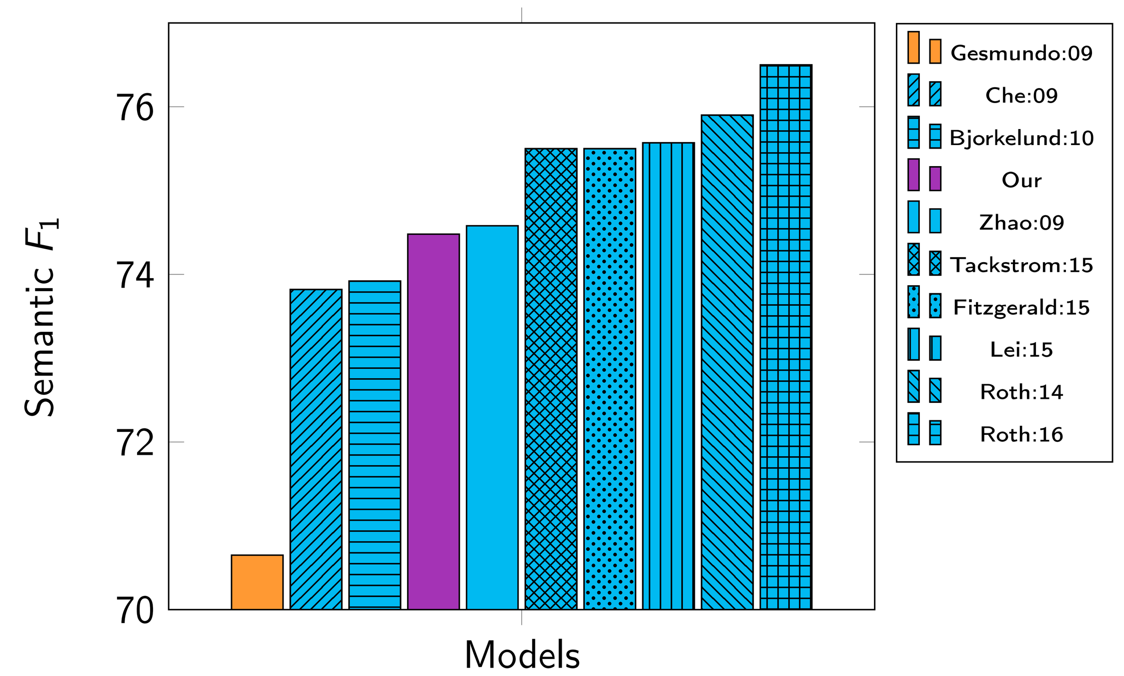
CoNLL 2008-09 shared task results



Our performance on the **CoNLL 2008 shared task** is on par with the best-performing, albeit pipelined model, despite not using a syntax-semantics pipeline.



Our model performance in the **CoNLL 2009 multilingual shared task**, is on par with the best multi-task learning systems, averaged across several languages.



In the CoNLL 2009 shared task, our SRL-only performance on out-of-domain English data is better than other multi-task models, but falls behind pipelined models which have access to more information (entire syntactic tree) than we do.