Basic SRL Taggers using AdaBoost

Lluís Màrquez and **Xavier Carreras**

TALP Research Center

Technical University of Catalonia

UIUC, June 9th, 2004

Problem Decomposition and Learning Architecture

- CoNLL-2004 datasets and setting.
- Proposition treated independently: each taget verb generates a sequence to be annotated.
- Learning one binary classifier for each label (B-A{0-5,A}, B-AM-*, I-A{0-5,A}, I-AM-*, B-C-*, I-C-*, I-V, O) for a total of 75 labels. But only the 40 most frequent labels were treated.
- Learning algorithm: AdaBoost with real-valued weak rules in the form of decision trees of fixed depth d (Schapire & Singer, 1999). Own implementation using Perl and C++.

- SRL tagging: sliding window based (sw) BIO tagging; left to right; greedy; also recurrent sliding window (rsw) with information about the left context.

- Sequence processing by "tokens": tokens are the top-most syntactic elements in the region of the sentence considered (always according to the chunk and clause annotation predicted for the sentence). Training set: 19,992 sequences totaling 216,042 tokens.

- The sentence "regions" explored for finding aruments are: 1) left and right contexts of the immediate clause containing the target predicate; 2) Left contexts of the parent clauses containing the target predicate's clause, up to the most general clause.

- Basic constraints ensured: B-I-O coherence. No embedding nor overlapping arguments. Arguments do not cross clause boundaries nor base chunk boundaries (except verb chunk containing the target predicate.

- Unicity of predicates not checked nor enforced; Continuation tags are not checked for a previous argument of the same type. No postprocessing performed.

An Example

```
********* development sentence #10
(S) +- (NP) +- [0: NNP Balcor]
    +- [1: , ,]
    +- (S) +- (NP) +- [2: WDT which]
    |      +- (S) +- (VP) +- [3: VBZ has]
    |             +- (NP) +- [4: NNS interests]
    |             +- (PP) +- [5: IN in]
    |             +- (NP) +- [6: JJ real]
    |                     +- [7: NN estate]
    +- [8: , ,]
    +- (VP) +- [9: VBD said]
    +- (S) +- (NP) +- [10: DT the]
    |      |       +- [11: NN position]
    |      +- (VP) +- [12: VBZ is]
    |              +- [13: RB newly]
    |              +- [14: VBN created]
    +- [15: . .]
```

Target predicates are: *have*, *say*, and *create*.

say has arguments: A0_(0 8) and A1_(10 14).

An Example

```
>>>>> target 10 3 V *have*: seq. #1
>>> region-PRE (0,1):
1 B-A0 0 1 type:chunk NP
2 O 1 1 type:word ,
>>> region-PRE (2,2):
3 B-R-A0 2 1 type:chunk NP
>>> region-ICL (3,2):
>>> target predicate (3,3):
4 B-V 3 1 has
>>> region-ICR (4,7):
5 B-A1 4 1 type:chunk NP
6 I-A1 5 1 type:chunk PP
7 I-A1 6 2 type:chunk NP
>>> region-OUT (8,15):
8 O 8 8 Not-treated
```

```
>>>>> target 10 9 V *say*: seq. #2
>>> region-ICL (0,8):
1 B-A0 0 1 type:chunk NP
2 I-A0 1 1 type:word ,
3 I-A0 2 6 type:clause S
4 I-A0 8 1 type:word ,
>>> target predicate (9,9):
5 B-V 9 1 said
>>> region-ICR (10,15):
6 B-A1 10 5 type:clause S
7 O 15 1 type:word .
```

```
>>>>> target 10 14 V *create*: seq. #3
>>> region-PRE (0,9):
1 O 0 1 type:chunk NP
2 O 1 1 type:word ,
3 O 2 6 type:clause S
4 O 8 1 type:word ,
5 O 9 1 type:chunk VP
>>> region-ICL (10,13):
6 B-A1 10 2 type:chunk NP
7 O 12 1 type:word is
8 B-AM-ADV 13 1 type:word newly
>>> target predicate (14,14):
9 B-V 14 1 created
>>> region-ICR (15,14):
>>> region-OUT (15,15):
10 O 15 1 Not-treated
```

Upper Bounds on Global Performance

	Precision	Recall	$F_{\beta=1}$
Training	99.09%	98.28%	98.68
Development	97.95%	96.22%	97.08
Test	97.32%	95.57%	96.44

Feature representation

(i) On the verb predicate:

- verb base form
- verb form
- verb POS
- type of VP in which verb is included: single/complex
- verb voice: active/passive

(ii) On the focus token (s, e) and its context:

- type and head
- POS and word forms at positions: $s-1, s, e, e+1$

- "surrounding" POS tags ($s - 1\%e + 1$)
- POS sequence from s to e, and $\{2, 3\}$-grams
- [-2,+2] token window. For each position: type, head
- left and right context to the next clause boundary: token-sequence and $\{2, 3\}$-grams, lexicalization, bag of words

(iii) On the relation between verb predicate and focus token:

- distance in tokens: {<-2,-2,-1,0,+1,+2,>+2}
- path in tokens (and taking into account clause embedding), and $\{2, 3, 4\}$-grams

Total number of features: 270,413. Filtered by frequency. Reduced to 55,544 features.

Basic Models Tested

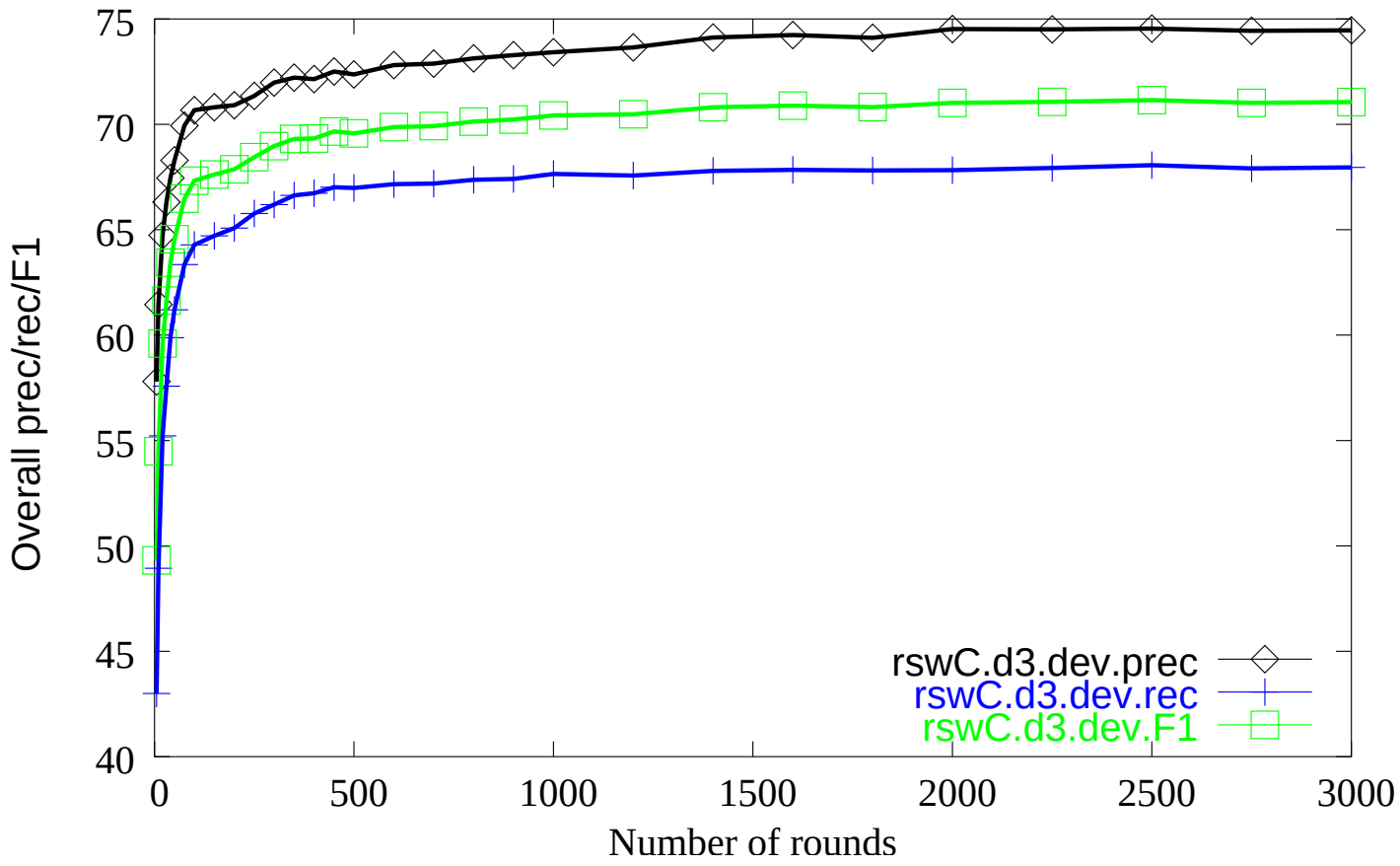
- **rswC.stumps**: recurrent sliding window model. The context considered are the three preceding argument labels assigned (correct labels during training, predicted labels when testing), e.g., "previous label is B-A0". Weak rules are decision stumps.We trained up to 3,000 weak rules for each of the labels.

- **rswC.d3**: same model as rswC.stumps but now the weak rules are decision trees of a maximum depth of three. We trained up to 3,000 weak rules for each of the labels.

- **rswA.d3**: recurrent sliding window model. The context considered include the three preceding argument labels (as in the rswC.d3 model) and the "bag-of-labels" of the left context of the token into consideration (from the estart of the sequence), e.g., "B-A1 appears in the left context". the labels.

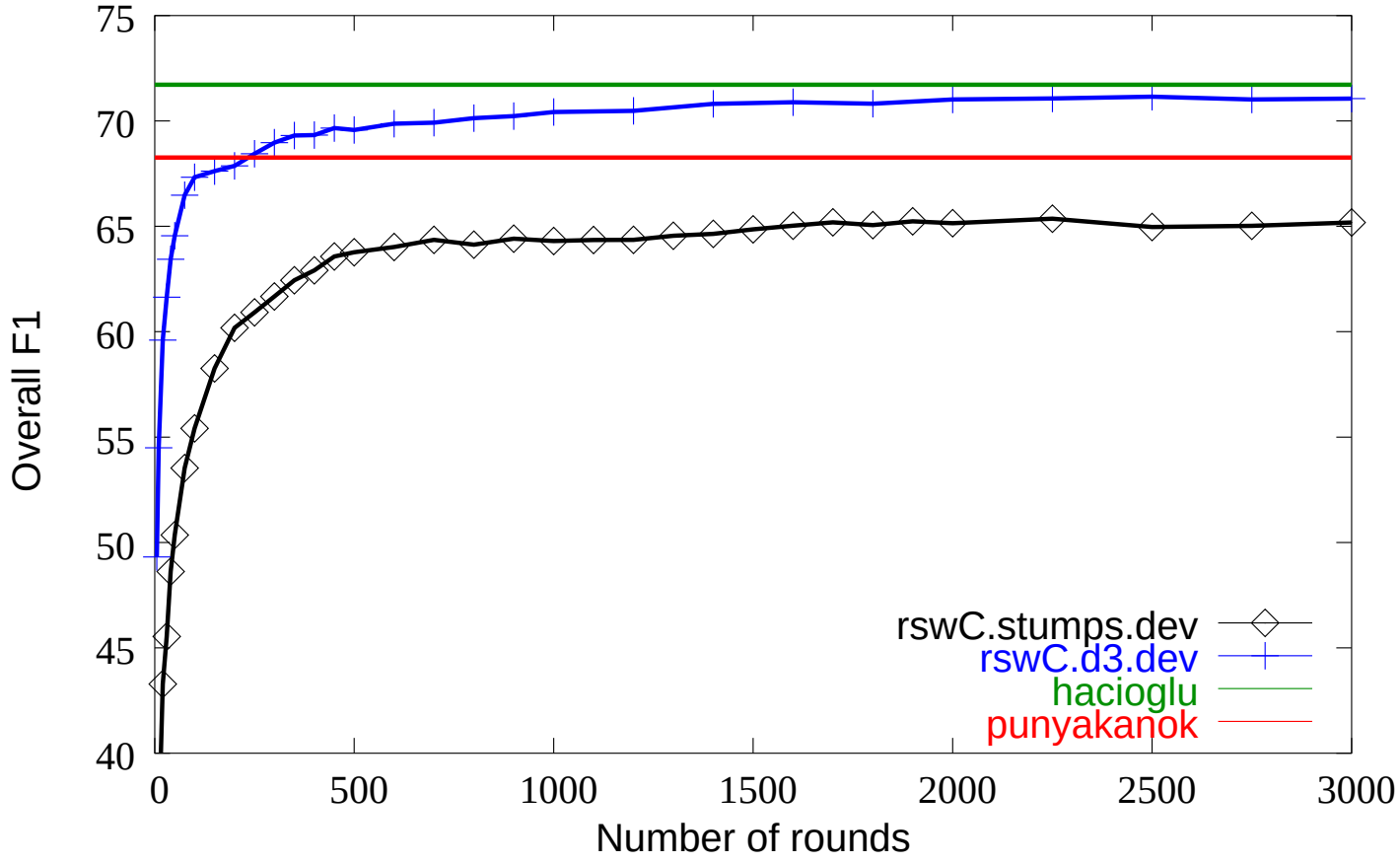
- **sw.d3**: sliding window model. Independent tagging decisions. Weak rules are decision trees of a maximum depth of three. We trained up to 2,000 weak rules for each of the labels.

Experimental Results



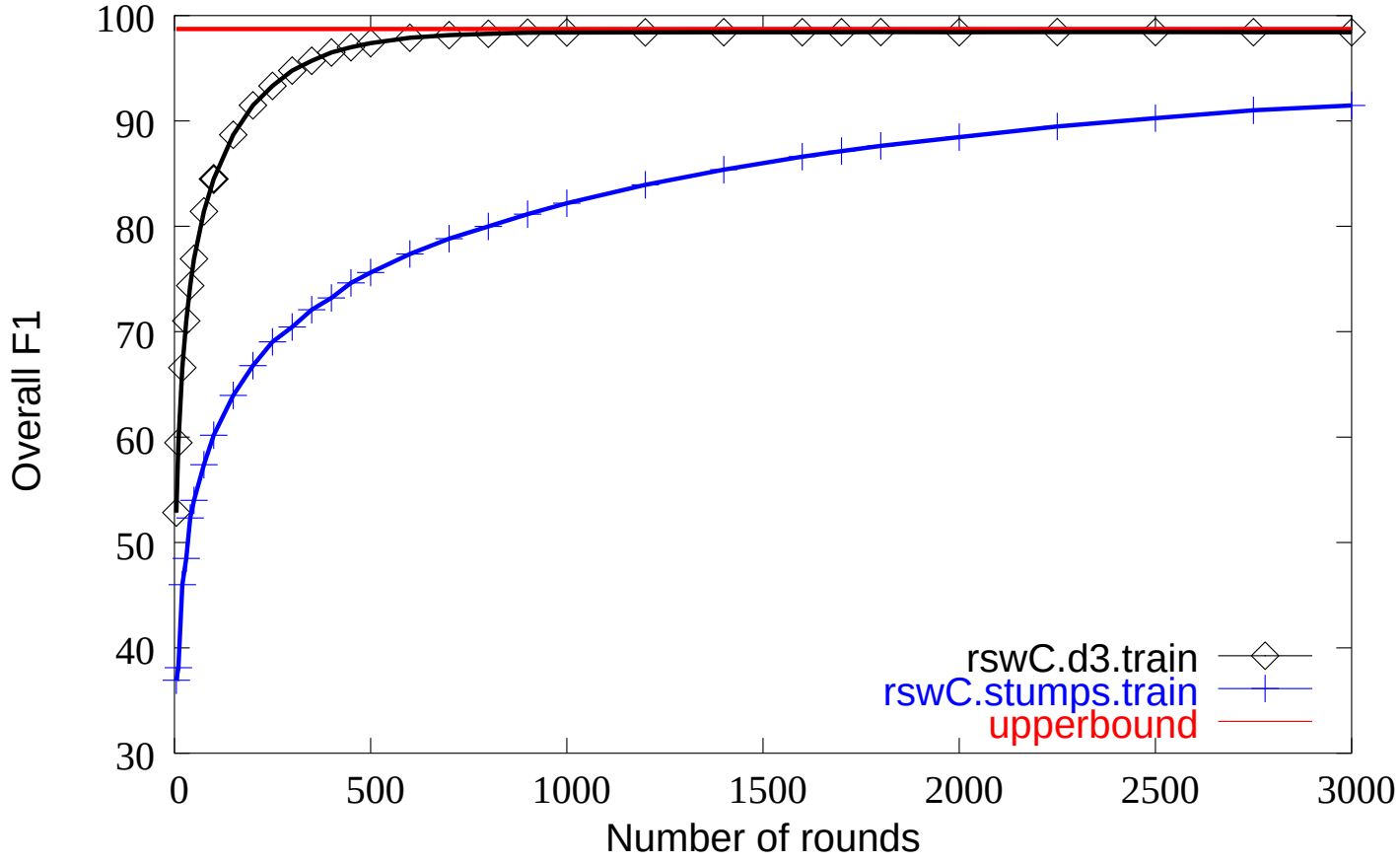
Overall prec/rec/F_1 of **rswC.d3** on the development set with respect to the number of rounds

Experimental Results



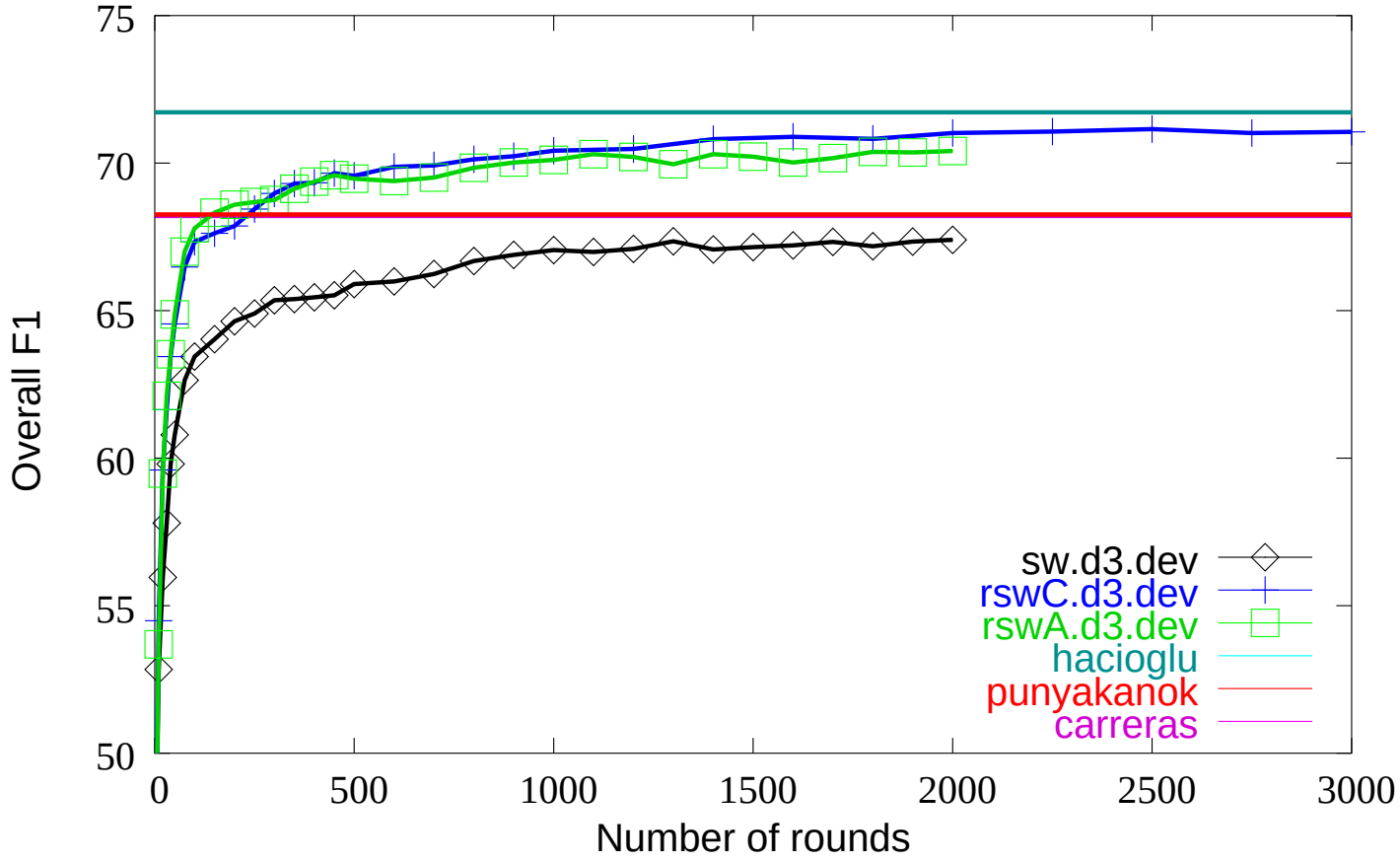
Overall F_1 results of **rswC.stumps** and **rswC.d3** on the development set with respect to the number of rounds

Experimental Results



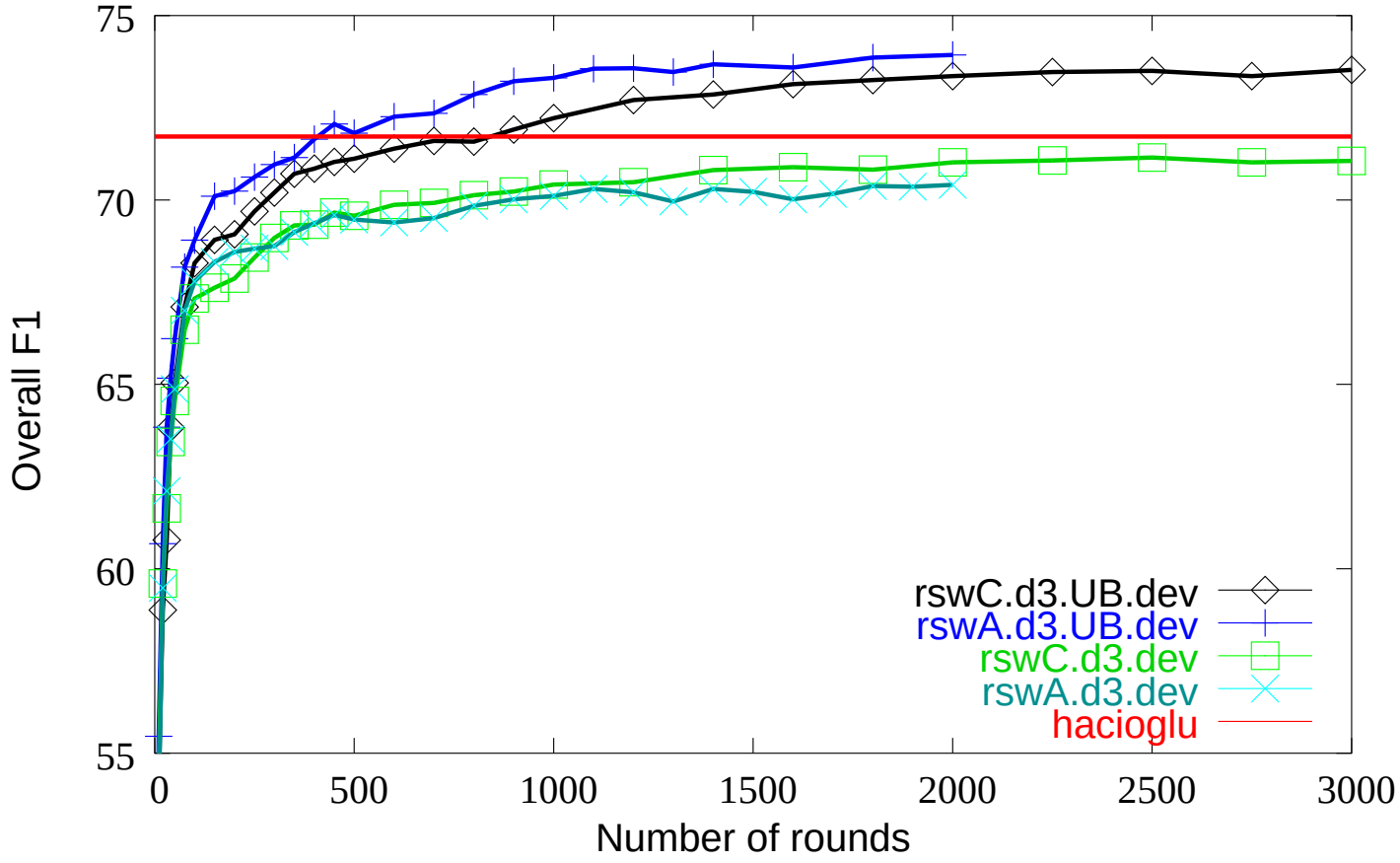
Overall results of **rswC.d3** and **rswC.stumps** on the training set with respect to the number of rounds = training convergence

Experimental Results



Overall F_1 results of **sw.d3**, **rswC.d3**, and **rswA.d3** on the development set with respect to the number of rounds

Experimental Results



Overall F_1 results of *rswC.d3* and *rswA.d3* systems assuming perfect context knowledge on the development set = upper bounds

Comparative Results

development	Precision	Recall	F_1
hacioglu	74.18%	69.43%	71.72
rswC.d3	**74.54%**	**68.06%**	**71.15**
punyakanok	71.96%	64.93%	68.26
carreras	73.40%	63.70%	68.21
sw.d3	**69.92%**	**65.05%**	**67.40**
lim	69.78%	62.57%	65.97
park	67.27%	64.36%	65.78
rswC.stumps	**67.96%**	**62.95%**	**65.36**
higgins	65.59%	60.16%	62.76
van den bosch	69.06%	57.84%	62.95
kouchnir	44.93%	63.12%	52.50
baldewein	64.90%	41.61%	50.71
williams	53.37%	32.43%	40.35
baseline	50.63%	30.30%	37.91

Comparative Results

test	Precision	Recall	F_1
hacioglu	72.43%	66.77%	69.49
rswC.d3	**73.33%**	**65.35%**	**69.11**
punyakanok	70.07%	63.07%	66.39
sw.d3	**69.71%**	**62.84%**	**66.09**
carreras	71.81%	61.11%	66.03
lim	68.42%	61.47%	64.76
park	65.63%	62.43%	63.99
rswC.stumps	**67.06%**	**60.94%**	**63.85**
higgins	64.17%	57.52%	60.66
van den bosch	67.12%	54.46%	60.13
kouchnir	56.86%	49.95%	53.18
baldewein	65.73%	42.60%	51.70
williams	58.08%	34.75%	43.48
baseline	54.60%	31.39%	39.87

Open Problems

- Determine the role of the segmentation and features on the final performance
- Proof that a "complex" learning architecture can perform better than the simple RSW approach. A fair comparison is needed.
- Exploit interactions between arguments of different verbs when learning and tagging
- Better exploit semantic information of the problem

Thank you very much for your attention!