Large-Scale Sequence Labelling

Antoine Bordes[*,†], Léon Bottou[†]
Ronan Collobert[†], and Jason Weston[†]

[*] LIP6, Université de Paris 6, Paris.

[†] NEC Laboratories America, Princeton.

NIPS Workshop on Efficient Machine Learning
December 8, 2007

Motivation

- Why working on large-scale learning
  if not for solving more complex problems?

- Recent works on SVM-like systems
  for structured learning tasks with kernels.
  - SVM-like = large margins + kernels
  - Sequence labelling = simplest structured learning task.

- There are very fast algorithms for SVM-like systems.
  - LaSVM: 8M examples on a single CPU (Loosli et al., 2006.)
  - LaRank: extension to multiclass problems and beyond (Bordes et al., 2007.)

- Could we mix?

Outline

I. Sequence Labelling

II. LaSVM & Co

III. Structure and Inference

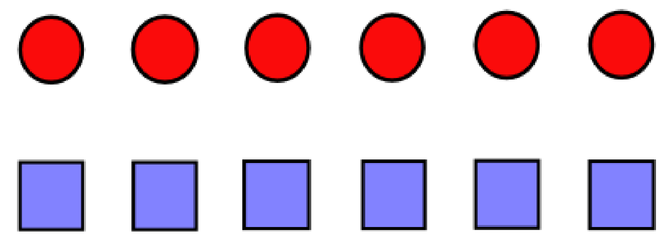
IV. Kernels

Part I

Sequence Labelling

Task

Goal: Given an input sequence (x_i) of tokens, produce an output sequence (y_i) of discrete labels.



Common applications:
– Speech recognition
– Language processing (tagging, chunking, etc.),

– Optical character recognition (OCR),
– Scene analysis (see workshop "grammar of vision"),
– etc.

Traditional methods:
– Probabilistic models (HMMs, CRFs).
– Rare works with non probabilistic losses
(Driancourt et al., 91; Katagiri et al., 92; LeCun et al., 98)

Sequence Labelling With Kernels

Structured Outputs with Kernels:

Recent works combine two ideas:

(Taskar et al., 2003; Altun et al., 2003)

– Using joint kernels to represent the global model.

– Using margin losses to train it.

Speed issues:

These methods are not considered to be very fast.

Virtually no experiments with real-size datasets.

Joint Kernels

A pattern-class pair $(\mathbf{x}, y)$ is either correct or incorrect.
This is treated as a two-class SVM without bias
using a joint kernel function: $K(\mathbf{x}, y, \bar{\mathbf{x}}, \bar{y}) = \langle \Phi(\mathbf{x}, y), \Phi(\bar{\mathbf{x}}, \bar{y}) \rangle$

Primal formulation:

$$\min_{w} \quad \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \quad \begin{cases} \forall i \quad \xi_i \geq 0 \\ \forall i \quad \forall y \neq y_i \quad \langle w, \Phi(\mathbf{x}_i, y_i) - \Phi(\mathbf{x}_i, y)\rangle \geq \delta - \xi_i \end{cases}$$

Dual formulation:

$$\max_{\boldsymbol{\beta}} \quad \sum_i \beta_i^{y_i} - \frac{1}{2}\sum_{i,j,y,\bar{y}} \beta_i^{y}\beta_j^{\bar{y}} K(\mathbf{x}_i, y, \mathbf{x}_j, \bar{y})) \quad \text{subject to} \quad \begin{cases} \forall i \quad \forall y \quad \beta_i^y \leq C\delta(y, y_i) \\ \forall i \quad \sum_y \beta_i^y = 0 \end{cases}$$

Support Patterns and Support Vectors:

– *Support Vector*: any pair $(\mathbf{x}_i, y)$ with $\beta_i^y \neq 0$.
– *Support Pattern*: any $\mathbf{x}_i$ with a nonzero β_i^y.

Benchmarks

Common datasets in recent literature:
– Optical Character Recognition (Taskar et al. 2003).
– Part-Of-Speech Tagging (CoNLL 2002).
– Text Chunking (CoNLL 2000).

$\Rightarrow$ Fully supervised tasks: labels are provided for every time index.
$\neq$ Weakly supervised tasks: one only knows constraints on labels.

Simpler problems $\longrightarrow$ **simpler approaches**.

First Question
Can these simpler approaches speed-up training?

Part II

LaSVM & Co

Main Properties

1. **Coordinate ascent** in dual space:
– One coordinate at a time.
– Two coordinates at a time when equality constraints force it.

2. Balance **coordinate choices** that:
– *reprocess* already seen examples that became SVs or SPs.
– *process* fresh examples.

3. **Equivalent view**:
– Computational cost vs statistical gain (*manage time*)
– Insertion vs deletion of SVs/SPs. (*manage memory*)

4. **Convergence**:
– Exhaustive convergence analysis.
– Number of iterations grows linearly with number of examples.

Performance Highlights

- LaSVM/LaRank achieves the SVM test performance after **a single pass on the training set** (Bordes et al., 2005, 2007).
- Speed gains are usually derived from a more conservative use of kernel cache memory.
- LaSVM has been used to train SVMs with 8M examples on a single CPU (Loosli et al., 2006). Training requires 20h and 6GB. This compares with a parallel SMO algorithm using 64 processors and 64GB (Durdanovic et al., 2006).

$\Rightarrow$ Family of efficient SVM solvers with interesting **online behavior**.

Second question:
Can this family speed-up training of structured output models?

Part III

Structure and Inference

Inference

- Use **LaRank** on benchmark tasks.

- Different modelling assumptions
  $\longrightarrow$ different inference algorithms
  $\longrightarrow$ different costs.

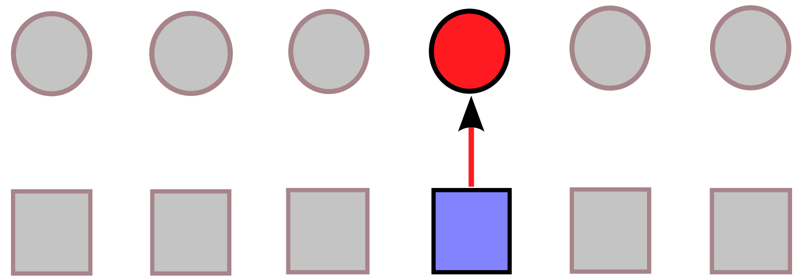
- Modelling assumptions:
  - Conditional independence
    Label y_t function of (x_{t+i}), $i \in \mathcal{I}$ and (y_{t+j}), $j \in \mathcal{J}$.
  - Invariance
    This function does not depend on t.

Multiclass Classification over Tokens

- $\mathcal{J} = \emptyset$.
- $\mathcal{I} = \{0\}$.

$$\Downarrow$$

- $K(x, y, \bar{x}, \bar{y}) = \delta_{y\bar{y}}\, k(x, \bar{x})$



- Prediction of successive labels y_t given their related token x_t.
- Input and output structures are not used.
- A basic multiclass classifier that can be easily refined.

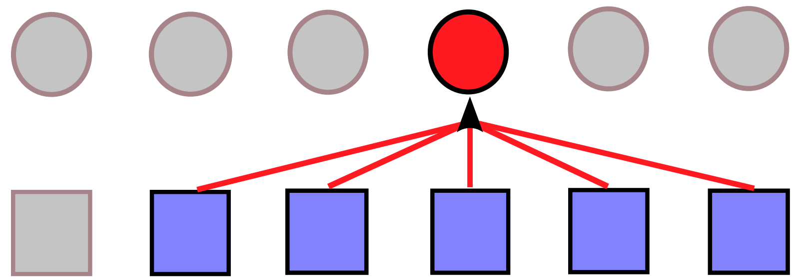
Greedy Inference using Input Context

– $\mathcal{J} = \emptyset$.

– $\mathcal{I} = [-n; m]$, $n, m > 0$.

$\Downarrow$

– $K(x, y, \bar{x}, \bar{y}) = \delta_{y\bar{y}} \left[k(x, \bar{x}) + \sum_{i \in \mathcal{I}} k(x_i, \bar{x}_i) \right]$



- Greedy prediction of successive labels y_t using an extended input time frame (x_{t+i}), $i \in \mathcal{I}$.

- Each time frame is an independent example.

- Output dependency is expressed via the overlapping inputs.

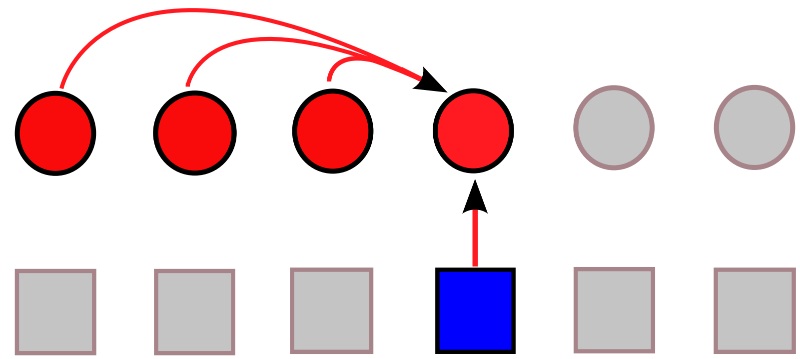
- It might be necessary to use a large input context $\mathcal{I}$: costly.

Greedy Inference using Output Context

- $\mathcal{J} = [-n; 0[, \; n > 0$.
- $\mathcal{I} = \{0\}$.

$$\Downarrow$$

- $K(x, y, \bar{x}, \bar{y}) = \delta_{y\bar{y}} \left[k(x, \bar{x}) + \sum_{j \in \mathcal{J}} \delta_{y_j \bar{y}_j} \right]$



- Greedy prediction of the successive labels y_t on the basis of the already predicted labels (y_{t+j}), $j \in \mathcal{J}$.

- During training the mapping function can influence the label y_t directly or via the previous predictions.

- Simple heuristics work relatively well (Daume at al., 2005).

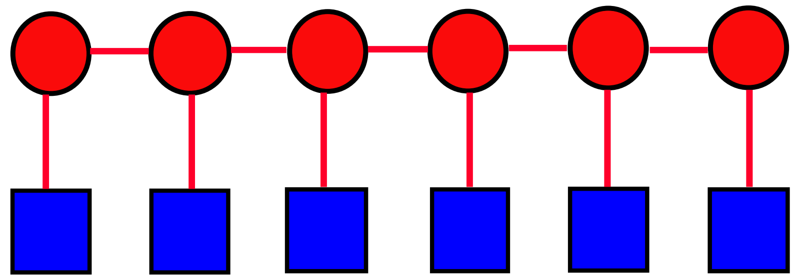
- No information about future labels.

Global Inference

– $\mathcal{J} = [-n; m]$, $n, m > 0$.
– $\mathcal{I} = \{0\}$.

$$\Downarrow$$

– $K(\mathbf{x}, y, \bar{\mathbf{x}}, \bar{y}) = \sum_{s,t} \delta_{y_s, \bar{y}_t} k(x_s, \bar{\mathbf{x}}_t)$
$+ \sum_{s,t} \delta_{y_s, \bar{y}_t} \delta_{y_{s+1}, \bar{y}_{t+1}}$.



- Label y_t depends on both past and future output labels.
- The output structure is considered as a whole object.
- Sophisticated inferences methods, such as, in simple cases, the Viterbi algorithm (Taskar et al., 2005),(Tsochantaridis et al., 2005).
- This involves bigger output spaces, larger number of features, higher computational costs.

Summary

- **Compared algorithms:**

  LaRank with **online** and **offline** stopping criteria using as inference step:

  1. Multiclass classification over tokens *denoted* **Multiclass**.
  2. Greedy inference using input context *denoted* **Greedy (inputs)**.
  3. Greedy inference using output context *denoted* **Greedy (labels)**.
  4. Global inference *denoted* **Global**.

  $\Rightarrow$ **8 slightly different methods**.

- **Experimental remarks:**
  - Comparison with external reference: SVMstruct or CRF.
  - Same features for all algorithms.
  - Only linear input kernel functions have been used.
  - **Greedy (labels)** has been trained using correct previous labels as context.

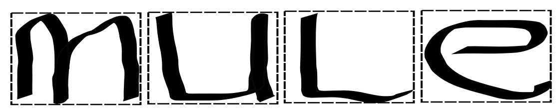
1^{st} task: Optical Character Recognition

Task description:

– Recognize handwritten characters of a word.

– Example:

m u l e



Dataset:

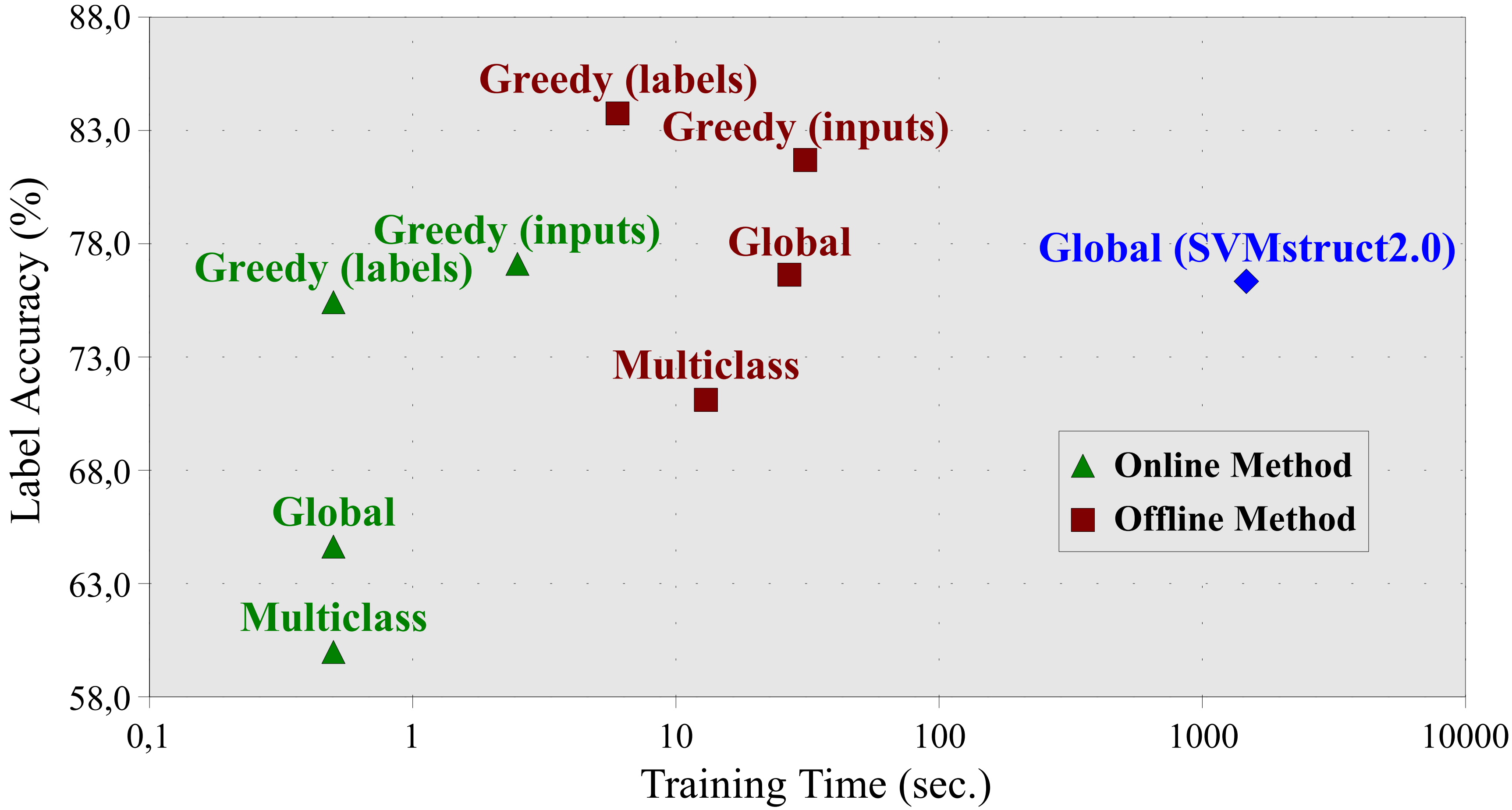
Classes	Training Seq.(Tokens)	Testing Seq.(Tokens)	Features
26	650 (4,600)	5,500 (43,000)	128

Context Size:

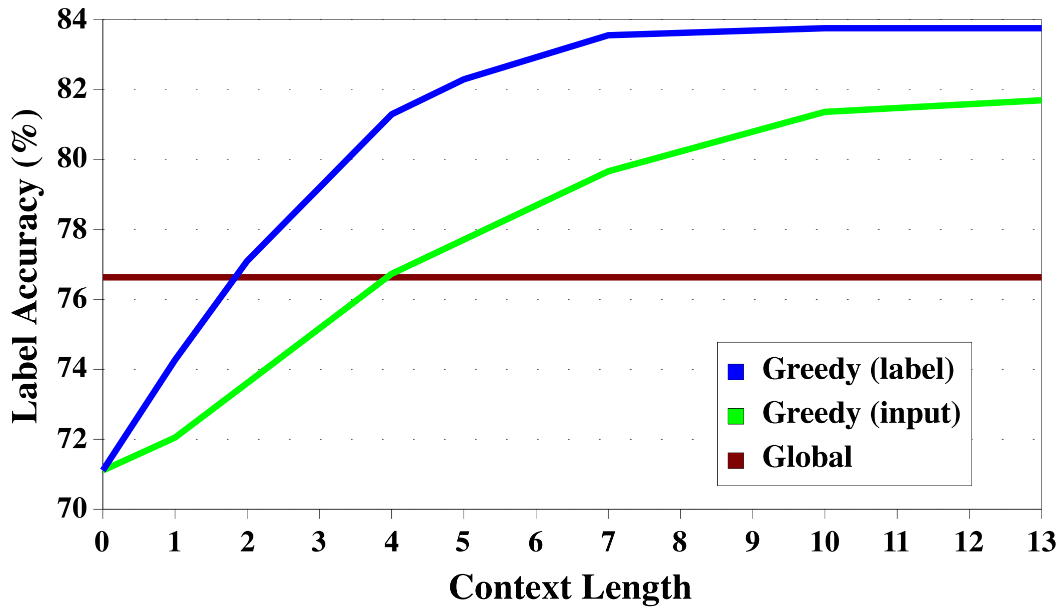
– **Greedy (inputs)**: a window of 25 input tokens.

– **Greedy (labels)**: 10 previous labels.

1^{st} task: Optical Character Recognition



Influence of the Context Length



- Inference complexity for a sequence of length l with N possible labels:
  - $\mathcal{O}(lN^2)$ with global inference (with 1^{st} order dependencies).
  - $\mathcal{O}(lN)$ with greedy inference.
- Global inference is restricted to 1^{st} order for tractability reasons.
- Using a larger context can lead to better generalization.

2^{nd} task: Part-Of-Speech Tagging

Task description:

– Label each word of a text with its Part-Of-Speech tag.

– Example:

PRP		VBD		DT		NN
He	/	opened	/	the	/	window

Dataset:

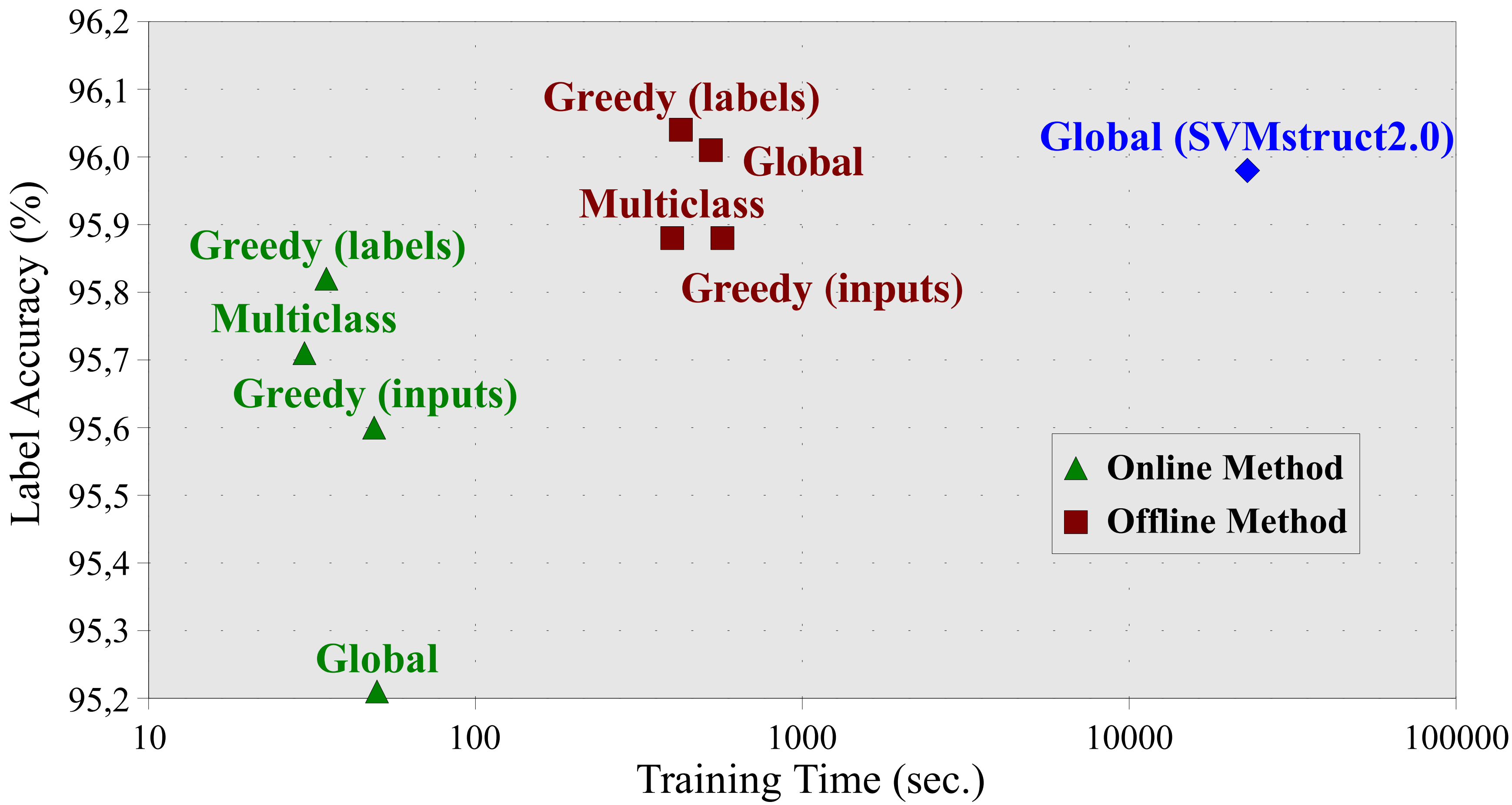
Classes	Training Seq.(Tokens)	Testing Seq.(Tokens)	Features
41	7,200 (172,000)	1,681 (40,000)	$>$400,000

Context Size:

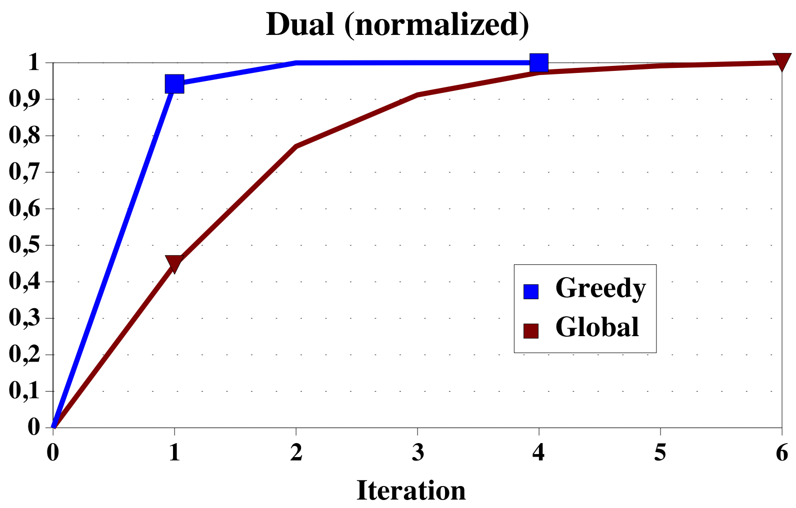
– **Greedy (inputs)**: a window of 3 input tokens.

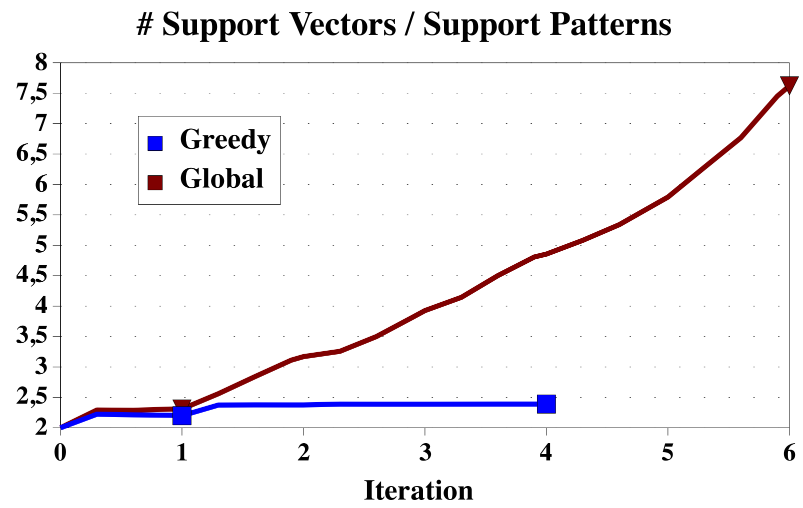
– **Greedy (labels)**: 2 previous labels.

2^{nd} task: Part-Of-Speech Tagging



Invariances





- Output space size for a sequence of length l with N possible labels:
  - N^l with global inference.
  - lN with greedy inference (l successive decisions).
- Support Vectors for the global model are complete sequences:
  - Local dependencies are not represented in an invariant fashion.
  - $\rightarrow$ More support vectors per support pattern.
- Greedy inference can deal with invariances.

3^{rd} task: Chunking

Task description:

– Divide a text in syntactically correlated parts.

– Example:

NP	VP	NP	VP	PP	NP
He /	reckons /	the current account deficit /	will narrow /	to /	only 1.8 billion

Dataset:

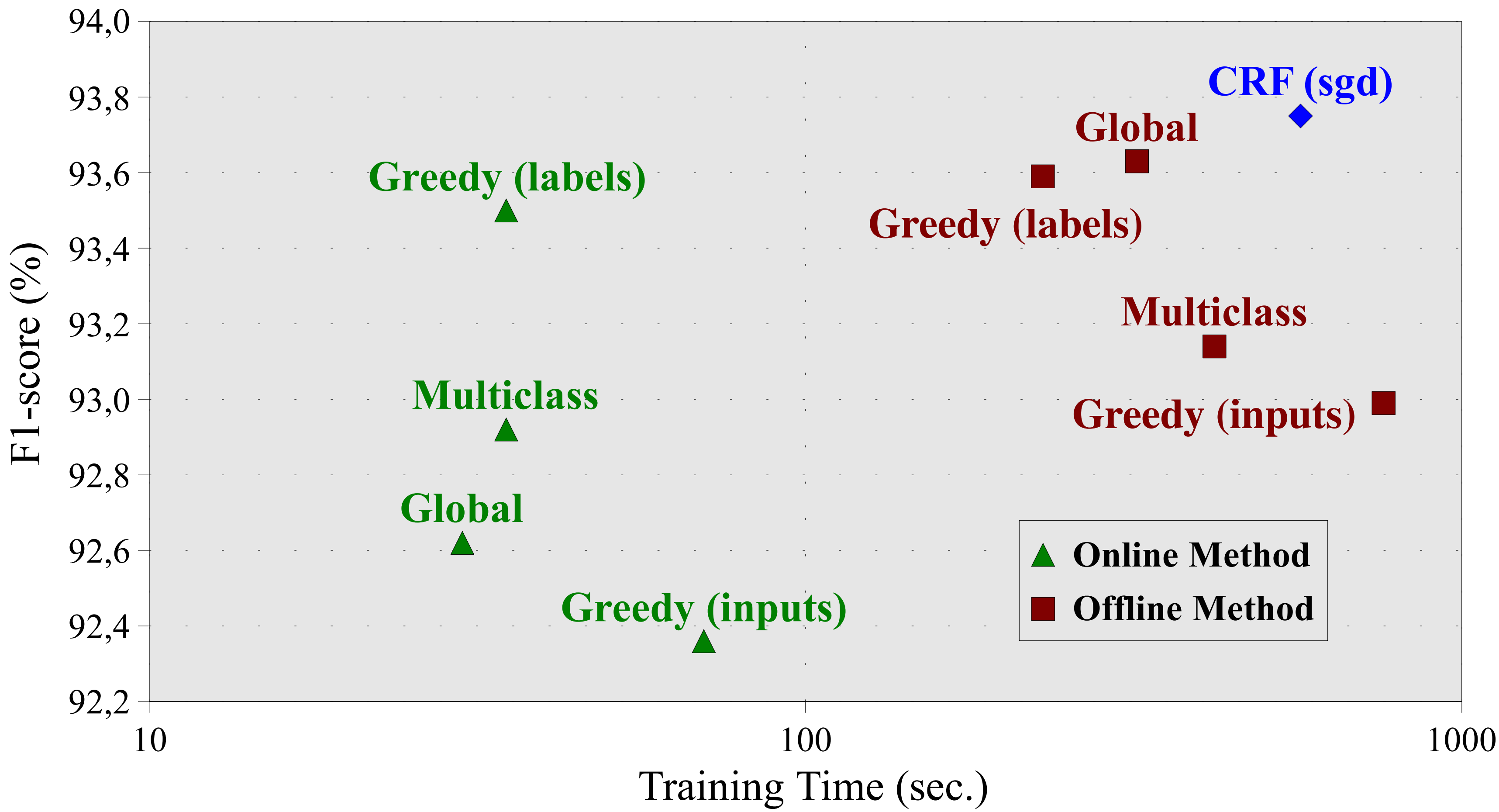
Classes	Training Seq.(Tokens)	Testing Seq.(Tokens)	Features
18	8,931 (212,000)	2,012 (47,000)	>76,000

Context Size:

– **Greedy (inputs)**: a window of 3 input tokens.

– **Greedy (labels)**: 2 previous labels.

3^{rd} task: Text Chunking



Partial Conclusion

- Greedy & global inference: similar generalization performances.
- Using LaRank as optimizer allows fast training.
  (5 min for POS or Chunking)
- But greedy inference $\longrightarrow$ local decisions:
  – Factorization of output space.
  – Handling of invariances.
  $\longrightarrow$ Efficient online learning.

- Online learning + Greedy inference = most efficient combination.
  (training in 30 sec. for POS or Chunking)
- Limitations of greedy inference:
  – Long-term dependencies.
  – Weak-supervision.

Part IV

Kernels

Large Scale Task: POS Tagging (bigger)

Task description:

– Label each word of a text with its Part-Of-Speech tag.

Dataset: the whole Wall Street Journal dataset.

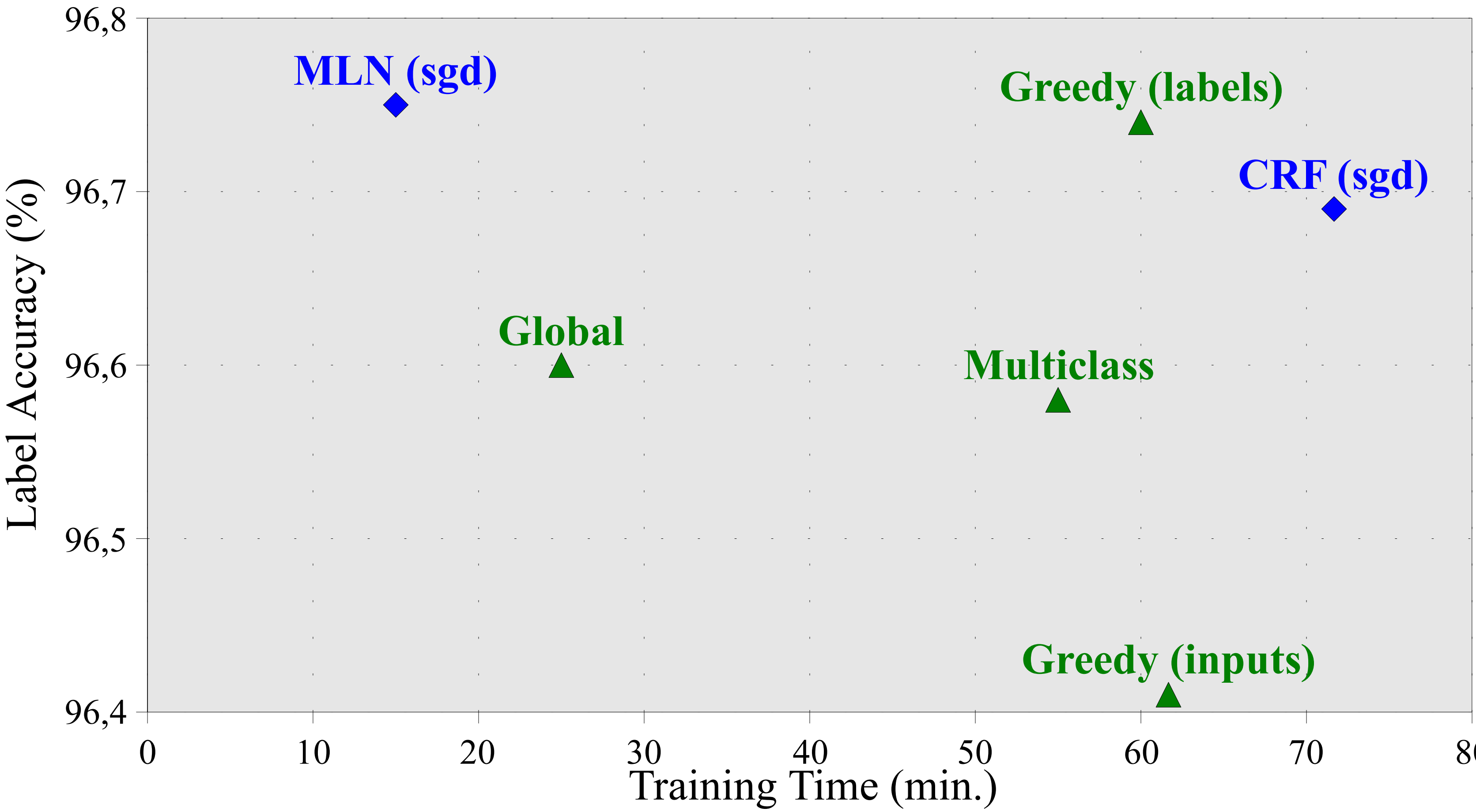
Classes	Training Seq.(Tokens)	Testing Seq.(Tokens)	Features
45	**107,633** (**3,072,872**)	5,284 (149,168)	>130,000

Context Size:

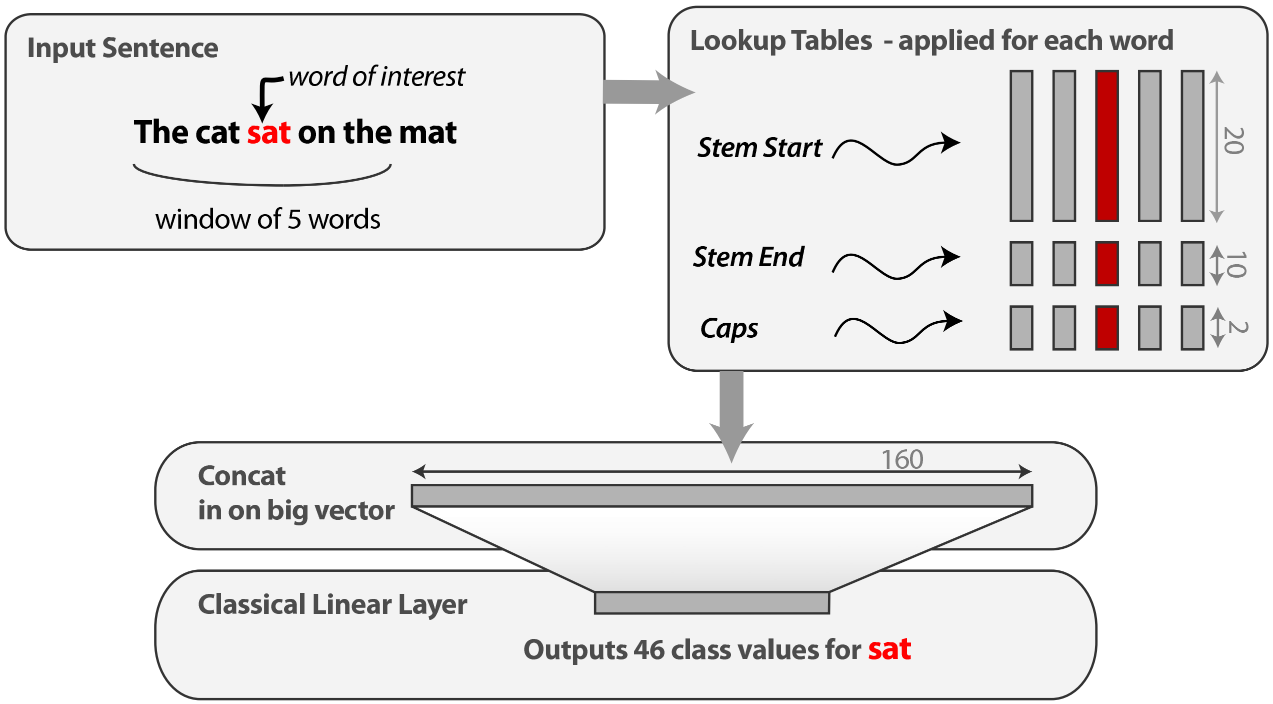
– **Greedy (inputs)**: a window of 3 input tokens.

– **Greedy (labels)**: 2 previous labels.

4^{th} task: Big Part-Of-Speech Tagging



Why the MLN is fast



The key: **Lookup tables**.

→ SGD on a vector of 160 attributes << 130,000!

→ Learned via backpropagation: hard to efficiently do with an SVM.

Conclusion

On simple tasks:

- Simple and sophisticated inference methods have similar performances and using LaRank can speed-up training.
- Greedy inference allows an efficient online training (faster).
- But all kernel-based methods are limited by their representation.
  $\longrightarrow$ Multi Layer Networks are not!