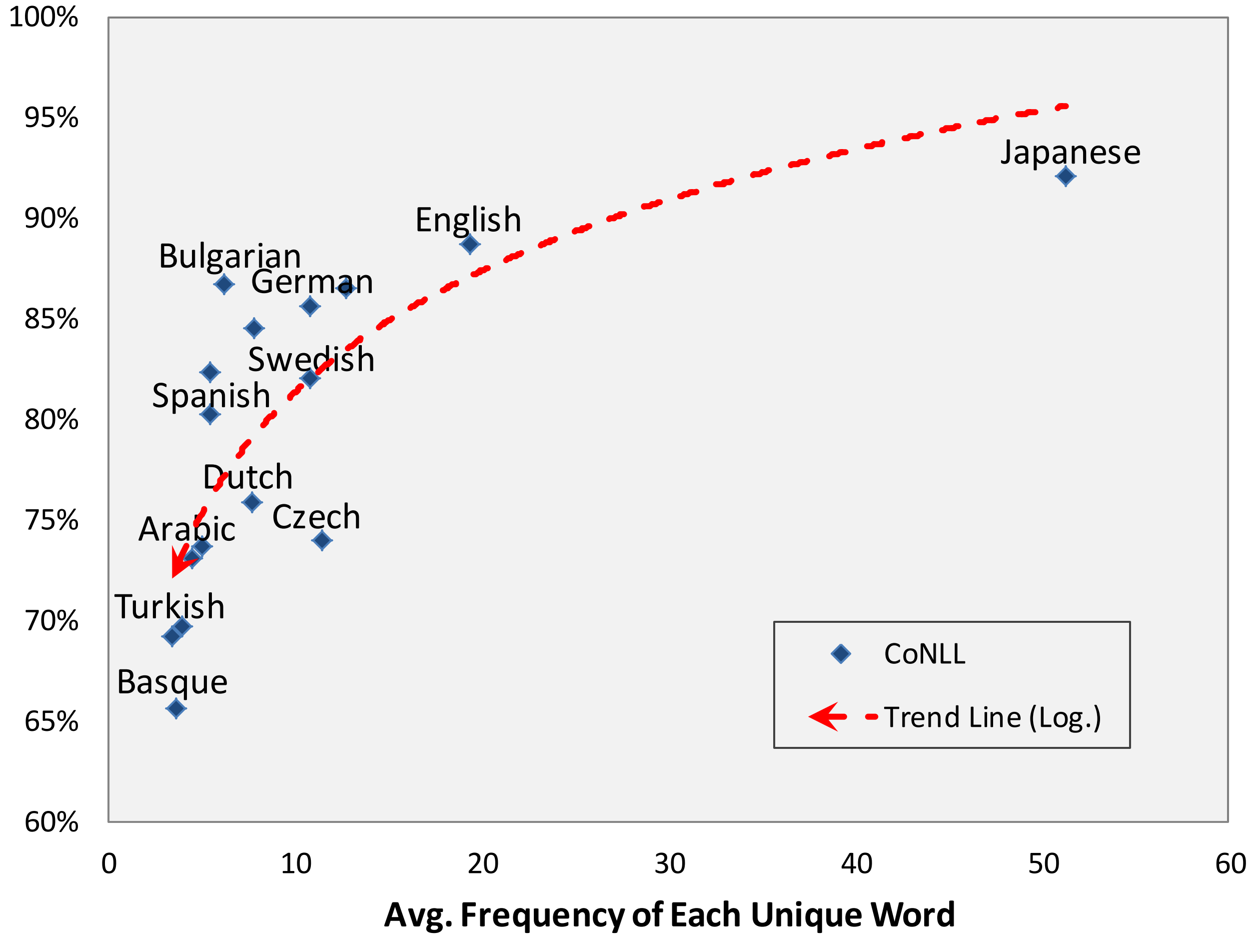
Learning to Parse Using a Tiny Corpus

Tao Lei, Yu Xin
Regina Barzilay, Tommi Jaakkola
CSAIL, MIT

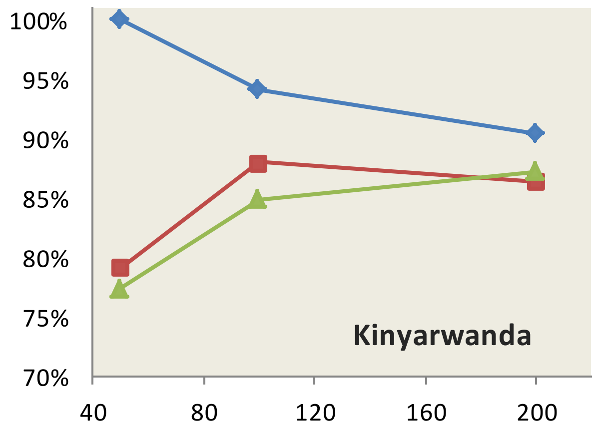
Sparsity Problem

- Data sparsity makes parsing harder
  - *due to less frequent/unseen words* and *dependency arcs* in data

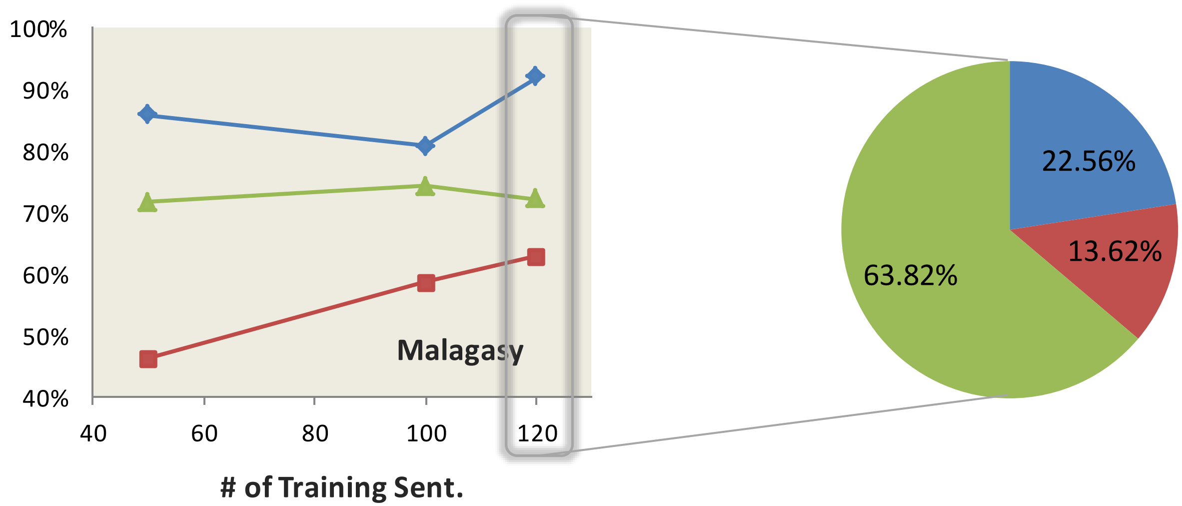


Sparsity Problem

- Prediction is worse when the arc is not seen in the training data



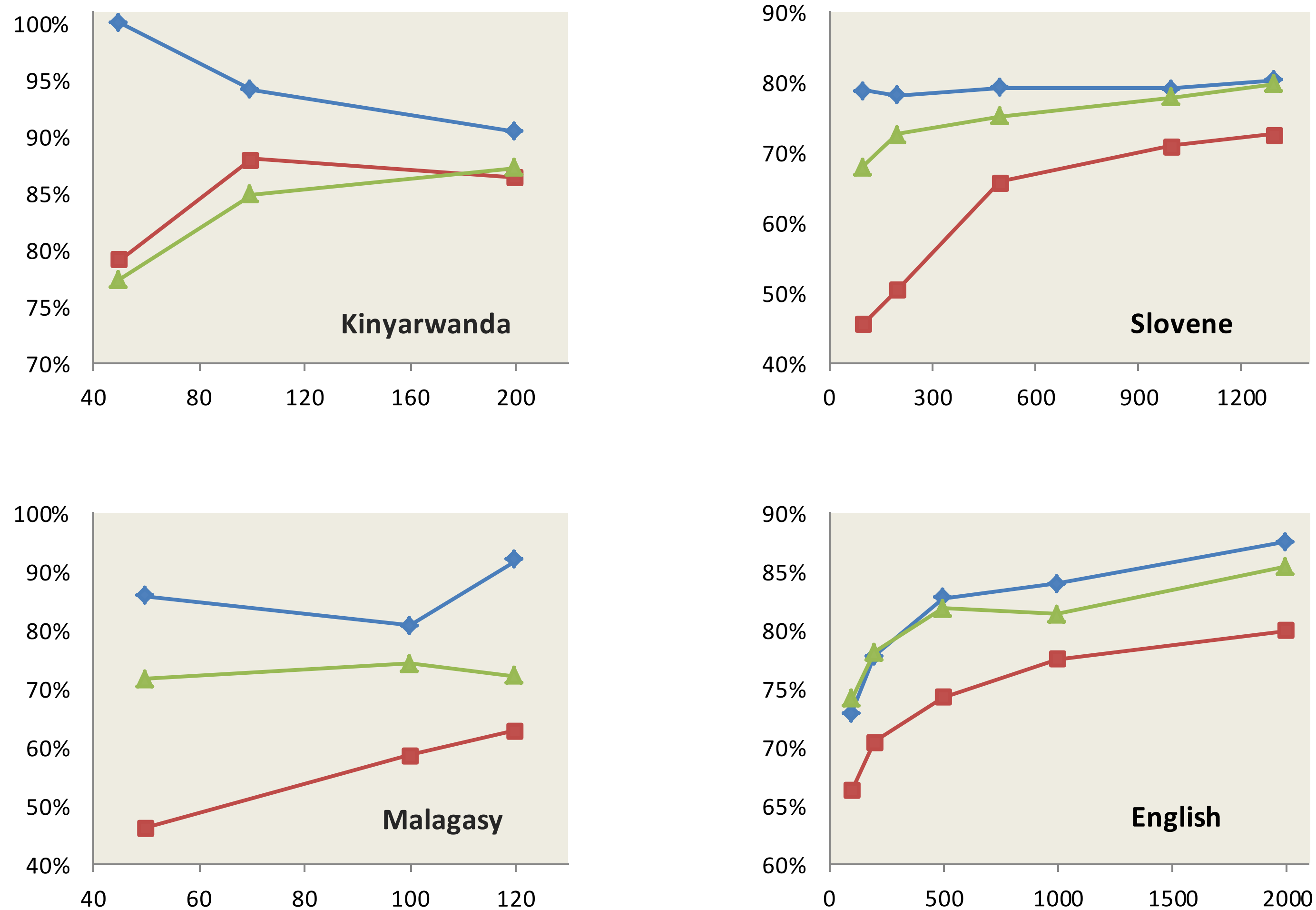
Seen Arc: See dependency arc in the training data
Seen Words: See the words in training but arc unseen
Unseen: At least one word not in training



a large portion of dependency arcs in test is unseen

Sparsity Problem

- Prediction is worse when the arc is not seen in the training data



Sparsity Problem

- Feature weights are zeros when the features are not seen

I	eat	a	cake	with	frosting
PRON	VB	DT	NN	IN	NN

eat⊕IN⊕frosting		VB⊕IN⊕NN		eat⊕IN⊕NN		…	
0	+	1.1	+	0.9	=	2.0	

unseen in training data

I	eat	a	cake	with	frosting
PRON	VB	DT	NN	IN	NN

cake⊕IN⊕frosting		NN⊕IN⊕NN		cake⊕IN⊕NN		…	
0	+	1.2	+	0.5	=	1.7	

Opportunity and Challenge

To deal with sparsity problem, we will

- Make the model flexible to add various rich features
  - E.g., words, coarse-to-fine POS tags and word embeddings
  - Feature selection: adjust complexity based on how much training data it has

- Model interactions between feature weights
  - E.g. propagating weights ~~from seen features to unseen features~~
  - E.g. propogating weights between features

Motivating Example: Matrix Completion

- Learn a matrix (or high-order tensor) that has a lot of unseen entries
  - Example: image

Input
image with missing values

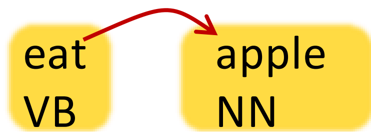
Output
re-constructed image

  - Example: Netflix problem

    Users give only a few movie ratings. Predict unseen ratings

Motivating Example

- In our case: learn a parameter matrix (or tensor) from sparse feature observations



Dependency Arc

$\{\, \text{eat} \oplus \text{apple}, \text{eat} \oplus \text{NN}, \text{VB} \oplus \text{apple}, \text{VB} \oplus \text{NN} \,\}$

Feature Strings

	eat	apple	banana	VB	NN
eat		1			1
apple					
banana					
VB		1			1
NN					

Feature Matrix

$\otimes$

...	**2**	...	...	**4**
...	0	0	...	...
...	0	0	...	
...	**1**	0.9	...	**5**
...	0.1	0.1	...	...

Parameter Matrix

$= 12$

Motivating Example

- In our case: learn a parameter matrix (or tensor) from sparse feature observations

Dependency Arc

$\{\, \text{eat} \oplus \text{banana}, \text{eat} \oplus \text{NN}, \text{VB} \oplus \text{banana}, \text{VB} \oplus \text{NN} \,\}$

Feature Strings

	eat	apple	banana	VB	NN
eat			1		1
apple					
banana					
VB			1		1
NN					

Feature Matrix

$\otimes$

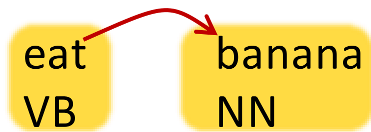
unseen entry

...	2	??	...	**4**
...	0	0	...	...
...	0	0	...	
...	1	**0.9**	...	**5**
...	0.1	0.1	...	...

Parameter Matrix

Motivating Example

- In our case: learn a parameter matrix (or tensor) from sparse feature observations



Dependency Arc

{ eat⊕banana, eat⊕NN, VB⊕banana, VB⊕NN }

Feature Strings

	eat	apple	banana	VB	NN
eat			1		1
apple					
banana					
VB			1		1
NN					

Feature Matrix

$\otimes$

similar columns because "apple" and "banana" have similar syntactic behavior

…	2	??	…	4
…	0	0	…	…
…	0	0	…	
…	1	0.9	…	5
…	0.1	0.1	…	…

Parameter Matrix

Motivating Example

- In our case: learn a parameter matrix (or tensor) from sparse feature observations

eat	banana
VB	NN

Dependency Arc

$\{\, eat \oplus banana,\ eat \oplus NN,\ VB \oplus banana,\ VB \oplus NN \,\}$

Feature Strings

	eat	apple	banana	VB	NN
eat			1		1
apple					
banana					
VB			1		1

$\otimes$

...	2	**2**	...	**4**
...	0	0	...	...
...	0	0	...	
...	1	**0.9**	...	**5**

$= 11.9$

Goal: learn a low rank parameter matrix

Preliminary (Matrix Norm)

- Goal: learn a low rank matrix $Z^{n\times n}$
  - Directly learning decomposition $Z = U^{n\times k}V^{k\times n}$ is hard -- non-convex
- Using ***matrix norm*** constraint instead

Vector Case v_i	**Matrix Case M_{ij}**
L1 norm: $\sum_i \lvert v_i \rvert$	Nuclear norm $\lVert \quad \rVert_*$: $\sum_k \sigma_k$
L2 norm: $\sum_i v_i^2$	Frobenous norm $\lVert \quad \rVert_F$: $\sum_{ij} M_{ij}^2 = \sum_k \sigma_k^2$
L∞ norm: $\max_i \lvert v_i \rvert$	Spectral norm $\lVert \quad \rVert_\infty$: $\max_k \sigma_k$

Formulation

- Recall first-order decoding objective:

$$\tilde{y}_i = \underset{y_i \in T(x_i)}{\operatorname{argmax}} \; S(y_i)$$
$$= \underset{y_i \in T(x_i)}{\operatorname{argmax}} \sum_{(h,c) \in y_i} \mathrm{s}(h,c)$$

- Define score as matrix (tensor) inner product:

$$s(h,c) = \boldsymbol{\theta} \otimes \phi(h,c)$$

$$\mathrm{s}(h,c) = {w_h}^T \boldsymbol{A}\, w_c + \boldsymbol{\eta}^{\mathrm{T}} d(h,c)$$

Model Parameters: $\boldsymbol{\theta} = \{\boldsymbol{A}, \boldsymbol{\eta}\}$

Feature Matrix/Vector: $w_h w_c^{\mathrm{T}}$ $d(h,c)$

Formulation

- Minimize the loss of training data:

$$\min_{A,\eta} \mathcal{L}(D; A, \eta) = \frac{1}{N} \ell(x_i, \hat{y}_i)$$

$$\text{s.t.} \quad \|A\|_* + \lambda \|\eta\| \leq C$$

Force A to be low-rank using nuclear norm constraint

  – online learning algorithm available
    (Jaggi & Sulovsky, 2010) (Hazan, 2008)

Initialize $Z^{(1)} := v_0 v_0^T$ for arbitrary unit vector v.
For $k = 1$ to T do
  Compute $v_k := \text{ApproxEV}\left(-\nabla f(Z^{(k)}), \frac{C_f}{k^2}\right)$.
  Set $\alpha_k := \frac{2}{k}$.
  Set $Z^{(k+1)} := Z^{(k)} + \alpha_k \left(v_k v_k^T - Z^{(k)}\right)$.
End for

Formulation

- Minimize the loss of training data:

$$\min_{A,\eta} \mathcal{L}(D; A, \eta) = \frac{1}{N} \ell(x_i, \hat{y}_i)$$

$$\text{s.t.} \quad \left\| \begin{pmatrix} A & 0 \\ 0 & \lambda\eta \end{pmatrix} \right\|_* \leq C$$

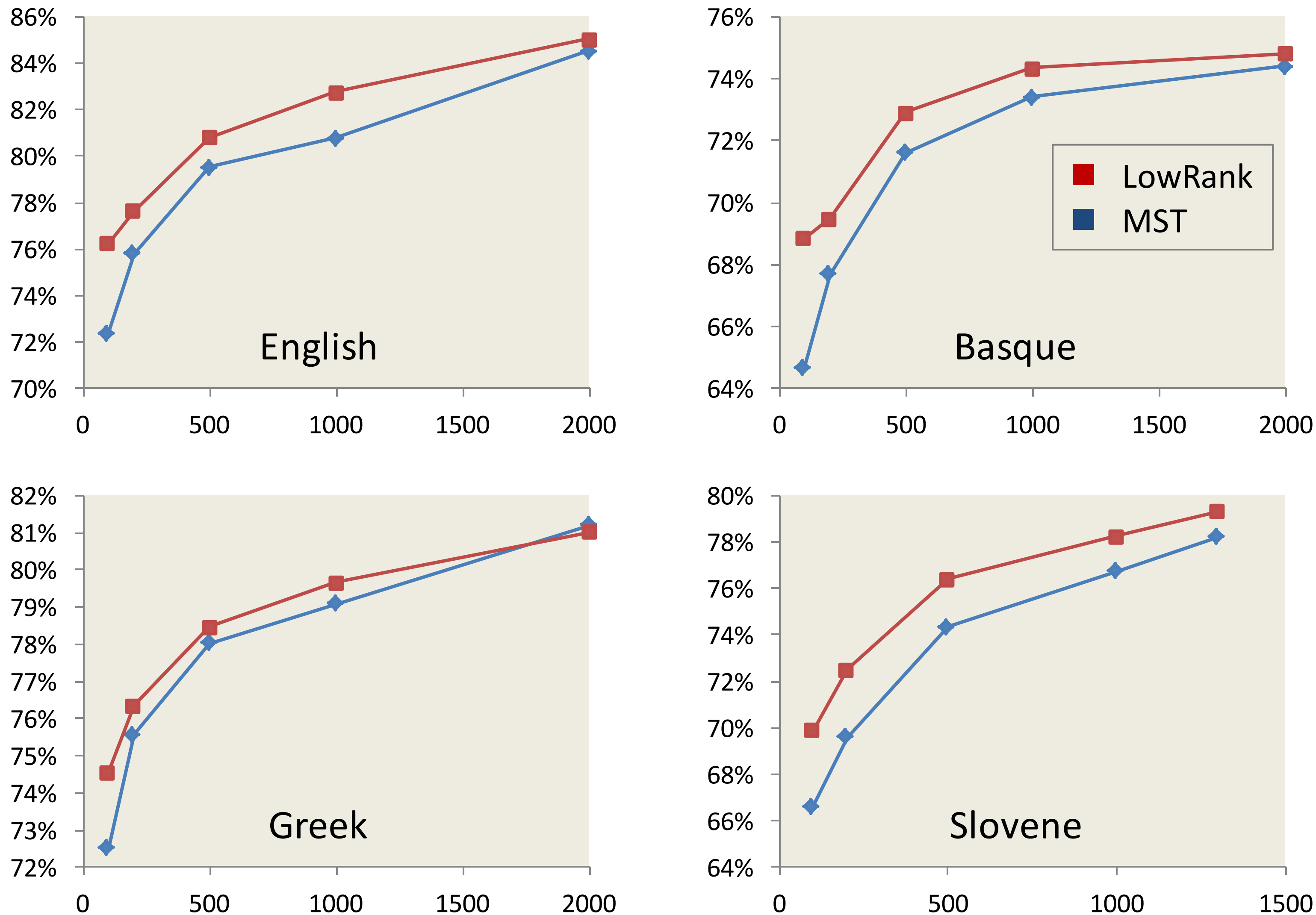
Force A to be low-rank using nuclear norm constraint

  - online learning algorithm available
    (Jaggi & Sulovsky, 2010) (Hazan, 2008)

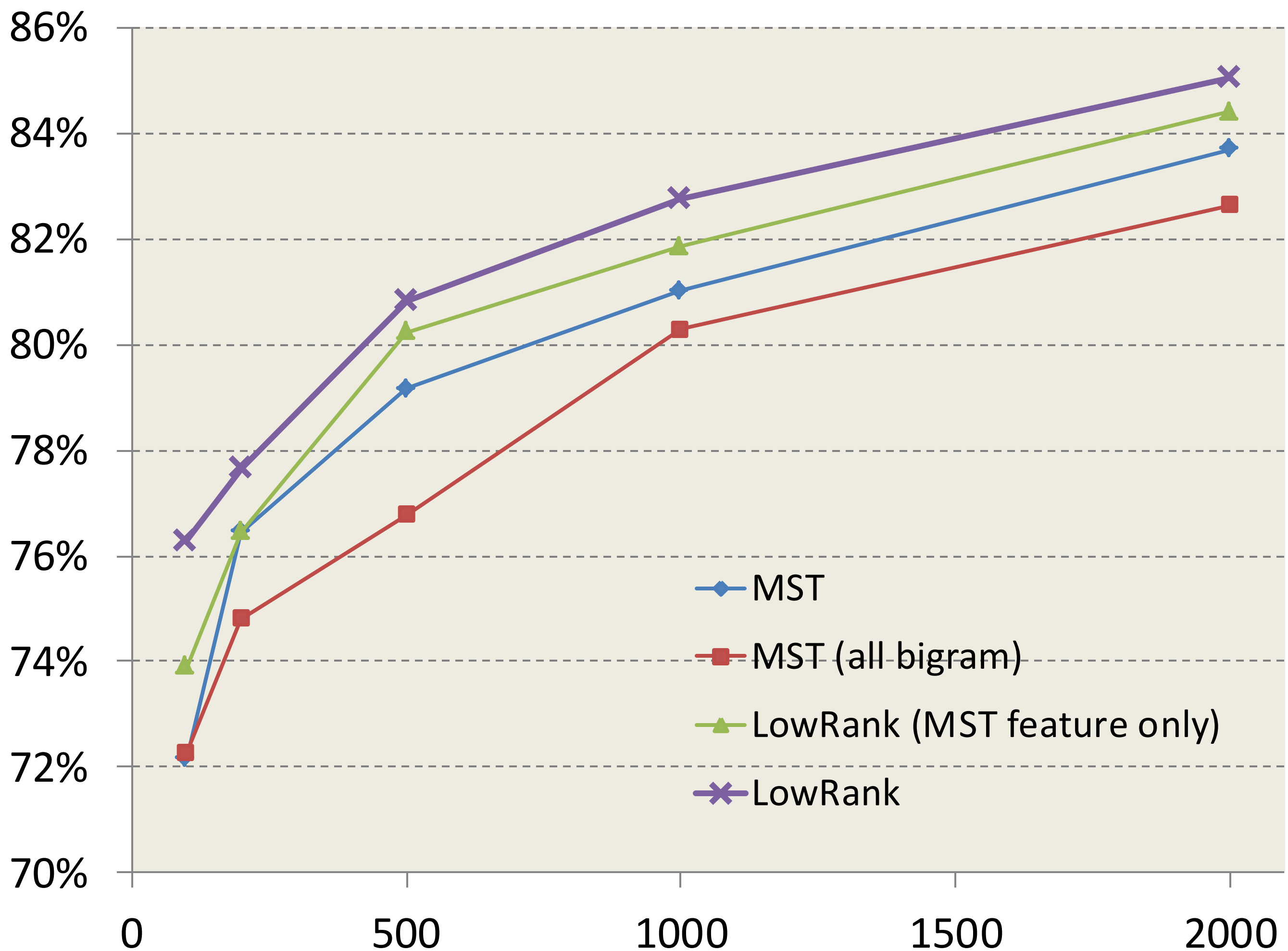
Initialize $Z^{(1)} := v_0 v_0^T$ for arbitrary unit vector v.
For $k = 1$ to T do
  Compute $v_k := \text{ApproxEV}\left(-\nabla f(Z^{(k)}), \frac{C_f}{k^2}\right)$.
  Set $\alpha_k := \frac{2}{k}$.
  Set $Z^{(k+1)} := Z^{(k)} + \alpha_k \left(v_k v_k^T - Z^{(k)}\right)$.
End for

Results

- Results on CoNLL shared task (up to 2000 sentences)



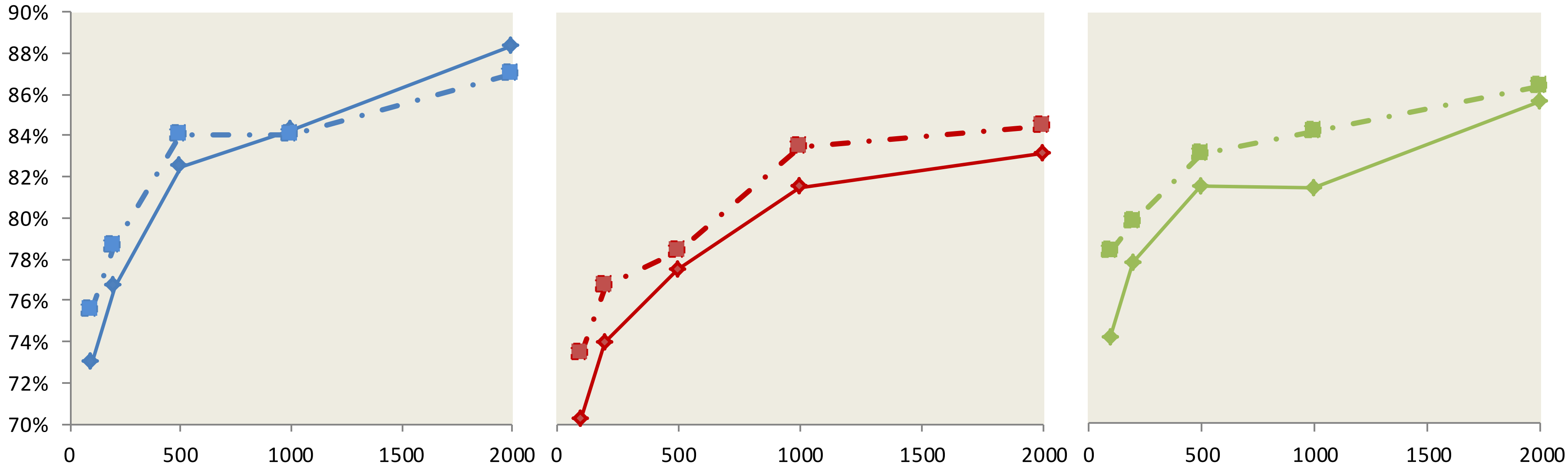
Results



86%
84%
82%
80%
78%
76%
74%
72%
70%
0
500
1000
1500
2000
MST
MST (all bigram)
LowRank (MST feature only)
LowRank

Results

- MST parser: solid lines
- Low-rank parser: doted lines



Seen Arc: See dependency in the training data
Seen Words: See the words in training but arc unseen
Unseen: At least one word not in training

Results

- Adding unsupervised word embeddings to English

	MST	MST+label	LowRank	LowRank+wv
100	72.5%	72.4%	76.3%	**76.6% (+0.3%)**
200	75.8%	75.8%	77.7%	**78.0% (+0.3%)**
500	79.4%	79.5%	80.8%	**81.4% (+0.6%)**
1000	80.9%	80.8%	**82.8%**	**82.8% (+0.0%)**
2000	83.7%	84.5%	85.1%	**85.8% (+0.7%)**

- **MST+label:** MST parser trained with labeled dependencies
- Other models are trained with only unlabeled dependencies.

Thanks

- Current implementation available at:

  http://people.csail.mit.edu/taolei/muri/lowrankparser.zip