



# Fast Logistic Regression for Text Categorization with Variable-Length N-grams

Georgiana Ifrim<sup>\*</sup>, Gökhan Bakır<sup>+</sup>, Gerhard Weikum<sup>\*</sup>

<sup>\*</sup> Max-Planck Institute for Informatics  
Saarbrücken, Germany

<sup>+</sup>Google Switzerland GmbH  
Zürich, Switzerland

# Outline

---

- Background
- Related Work
- Our Approach
  - **Structured Logistic Regression**
- Experiments
- Conclusion

# Background

---

## □ Text Categorization using Machine Learning

□ Typical steps: Training docs -> Feature selection -> Build classifier -> Classify test docs

□ Typical document representation: **bag-of-words**

□ Document: “This phone is good.”

*document,  $x = (\dots, is, phone, good, this, \dots)$*

*classifier weights,  $\beta = (\dots, -0.1, 0.3, 0.7, -0.3, \dots)$*

□ By this representation we lose: structure, order, context.

□ Should we care?

# Background - Motivation

---

- ❑ **+1 This product is not bad, it's actually quite good.**
- ❑ **-1 This product is not good, it's actually quite bad.**
  
- ❑ Bag-of-words representation for both samples :
  - ❑ (actually, **bad**, it's, **good**, not, product, quite, This)
  
- ❑ N-gram representation:
  - ❑ +1 (This, product, is, not, bad, it's, actually, quite, good, This product, product is, **not bad**, ..., **quite good**, This product is, product is not, **is not bad**, ..., **it's actually quite good**,.....)
  - ❑ -1 (This, product, is, not, good, it's, actually, quite, bad, This product, product is, **not good**, ..., **quite bad**, This product is, product is not, **is not good**, ..., **it's actually quite bad**, ...)

# Background - Motivation

---

- ❑ Simple solution to this problem: use syntax clues, use n-grams for **fixed**  $n$  (e.g. all bigrams), e.g. feature engineering
- ❑ Problem with this idea: feature selection depends on
  - ❑ application, language, domain, corpus size
- ❑ Ideally:
  - ❑ Regard text as a sequence of word or character tokens
  - ❑ Consider as features all the possible subsequences (n-grams) in the training set (But:  **$u = \#$  unigrams,  $d = O(u^n)$  n-grams**)

# Our Contribution

---

- ❑ A new (**S**tructured) **L**ogistic **R**egression algorithm able to select a compact set of discriminative variable-length n-grams
- ❑ **SLR** exploits the structure of the n-gram space in order to transform the **learning** problem into a **search** problem
- ❑ **SLR learns:** ..., not bad, quite good, ...  
..., not good, quite bad, ...

# Related Work

---

## ❑ Efficient, regularized learning algorithms

- ❑ SVM<sup>perf</sup> (Joachims06)
- ❑ Sparse logistic regression (Genkin07: BBR, Komarek03: LR-TRIRLS, Efron04, Zhang01)

## ❑ Text Categorization using word/char n-grams

- ❑ Markov chain models
  - ❑ **fixed order** (n-gram language models, Peng2004), **variable order** (PST - Prediction Suffix Trees, Dekel04), (PPM, PPM\* - Prediction by Partial Matching, Cleary97)
- ❑ SVM with string kernel (Lodhi01), efficient feature selection + SVM (Zhang06)

## ❑ Optimization approaches to solving logistic regression

- ❑ Newton algorithms ( $O(d^2)$  memory)
- ❑ Limited memory BFGS ( $O(d)$ ), conjugate gradient ( $O(d)$ , Nocedal06), iterative scaling ( $O(d)$ , Jin03), cyclic coordinate descent ( $O(d)$ , Shevade03, Zhang01)

# The Problem

---

## □ Given:

□ X set of samples

□  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$  training set,  $y_i \in \{0, 1\}$

□ Features: all n-grams in the training set

□ **d = # distinct n-grams in training set**

□  $\beta = (\beta_1, \beta_2, \dots, \beta_d)$  parameter vector

□ Log-likelihood of training set for logistic regression

$$P(y_i = 1 | x_i, \beta) = \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}}$$

$$l(\beta) = \sum_{i=1}^N [y_i \cdot \beta^T \cdot x_i - \log(1 + e^{y_i \cdot \beta^T \cdot x_i})]$$

## □ Goal:

□ Learn a mapping  $f : X \rightarrow \{0, 1\}$  from given training set D

□ Solved by: Find parameter vector  $\beta$  that maximizes  $l(\beta)$

# Our Approach

- Solve logistic regression by **coordinate-wise gradient ascent**

$$\beta^{new} = \beta^{old} + \varepsilon \cdot \frac{\partial l}{\partial \beta}(\beta^{old}), \varepsilon = \text{step size (line search)}$$

- In each iteration: estimate ascent direction and step size.
- Computing full gradient vector not feasible for the space of all possible n-grams in the training set ( $u = \#$  unigrams,  $d = O(u^n)$ )

$$\frac{\partial l}{\partial \beta}(\beta) = \left( \frac{\partial l}{\partial \beta_1}(\beta), \frac{\partial l}{\partial \beta_2}(\beta), \dots, \frac{\partial l}{\partial \beta_d}(\beta) \right)$$

- **Our ascent direction:**

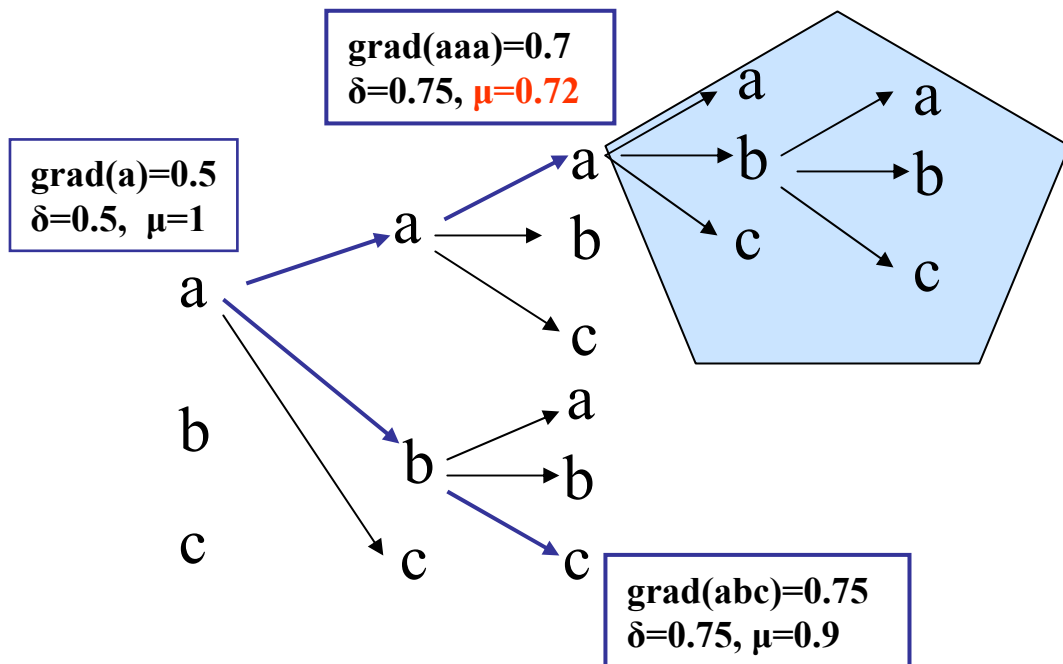
$$\left( \frac{\partial l}{\partial \beta}(\beta) \right)^{sparse} = \left( 0, 0, \dots, 0, \frac{\partial l}{\partial \beta_j}(\beta), 0, \dots, 0, 0 \right), \text{ where } \beta_j = \arg \max_{\beta_j, j \in \overline{1, d}} \left| \frac{\partial l}{\partial \beta_j}(\beta) \right|$$

# Searching & Pruning

---

- Goal: Find coordinate  $\mathbf{j}$  s.t.  $\beta_{\mathbf{j}} = \arg \max_{\beta_{\mathbf{j}}, \mathbf{j} \in \overline{1, d}} \left| \frac{\partial l}{\partial \beta_{\mathbf{j}}}(\beta) \right|$
- For super sequence  $s_{j'} \supseteq s_j$ , find upper bound on gradient value as a function of subsequence  $s_j$ :  $\text{gradient}(s_{j'}) \leq \mu(s_j)$
- Can prune the search space starting at  $s_j$  if  $\mu(s_j) \leq \delta$ , where  $\delta$  is the current suboptimal gradient value

# Searching & Pruning



## Branch-and-bound strategy

If  $\mu(\text{aaa}) = 0.72$  then the gradient of any super-sequence of **aaa** is not better than 0.72

$$s_{j'} \supseteq s_j \quad \text{gradient}(s_{j'}) \leq \mu(s_j)$$

□ Goal: Find coordinate  $j$  s.t.  $\beta_j = \arg \max_{\beta_j, j \in \overline{1, d}} \left| \frac{\partial l}{\partial \beta_j}(\beta) \right|$

# Upper Bound

---

## □ Theorem:

For any super sequence  $s_{j'} \supseteq s_j$  it holds that  $\left| \frac{\partial l}{\partial \beta_{j'}}(\beta) \right| \leq \mu(\beta_j)$

where

$$\mu(\beta_j) = \max \left\{ \sum_{\{i|y_i=1, s_j \in x_i\}} x_{ij} \left( 1 - \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right), \sum_{\{i|y_i=0, s_j \in x_i\}} x_{ij} \left( \frac{e^{\beta^T \cdot x_i}}{1 + e^{\beta^T \cdot x_i}} \right) \right\}$$

# Experiments

---

## □ Test collections

### □ IMDB: movie genre classification

- Categories: Crime vs Drama; 7,440 documents; 63,623 word unigrams, 800,000 up to trigrams, 1.9 million up to 5-grams
- Task: Given movie plots, learn the genre of movies

### □ CHINESE: topic detection for Chinese text

- Corpus: TREC-5 People's Daily News (3600 documents, 4,961 characters)
- Categories: (1) Politics, Law and Society; (2) Literature and Arts; (3) Education, Science and Culture; (4) Sports; (5) Theory and Academy; (6) Economics.

# Experiments

---

## ❑ Compared Methods

- ❑ Linear training time SVM: **SVM<sup>perf</sup>** (Joachims, KDD06)
- ❑ Bayesian Logistic Regression: **BBR** (Genkin et al, Technometrics07)
- ❑ Structured Logistic Regression: **SLR** (Ifrim et al, KDD08)

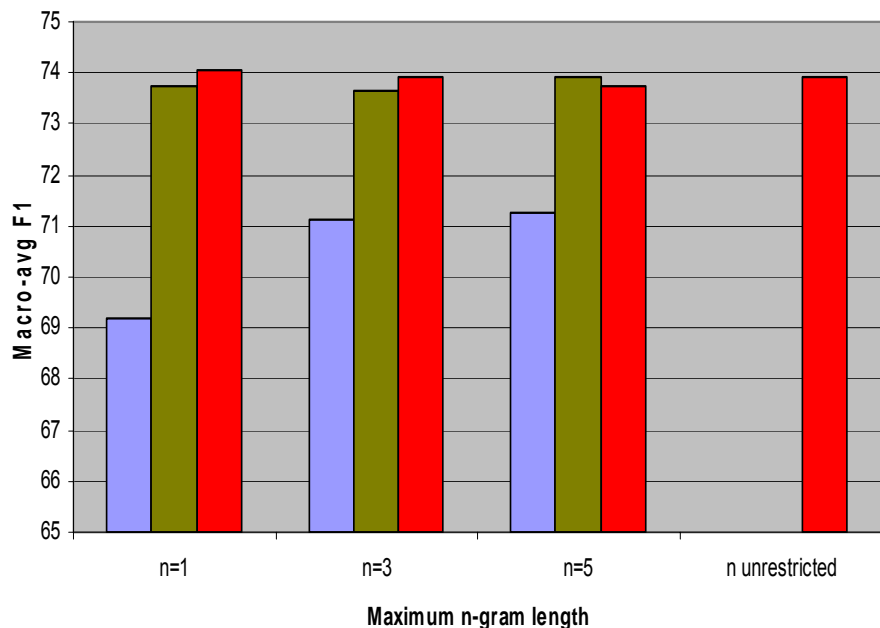
## ❑ Methodology

- ❑ SVM, BBR: generate space of variable-length n-grams explicitly, e.g. for n=3, generate all unigrams, bigrams and trigrams
- ❑ Evaluate all methods w.r.t. training run-times, micro/macro-avg F1
- ❑ Parameters:
  - ❑ SVM: C = 100 (fixed), C in {100,...,500} (tuning)
  - ❑ BBR: -p 1 -t 1 (fixed), -p 1 -t 1 -C 5 -autosearch (tuning)
  - ❑ SLR: # optimization iterations, set by fixed threshold (0.005) on the aggregated change in score predictions, or varied between 200 and 1,000 for tuning

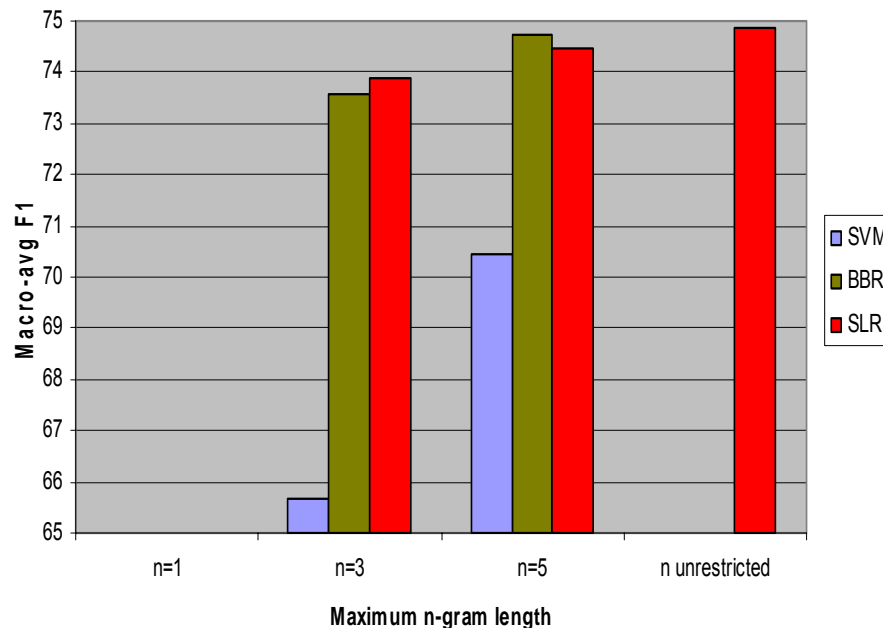
# IMDB – Macro-avg F1

- 5-fold CV; Plots show Macro-avg F1, Micro-avg F1 similar
- $n$  = max n-gram length, i.e. all n-grams up to length  $n$

Macro-avg F1 comparison (word n-grams)



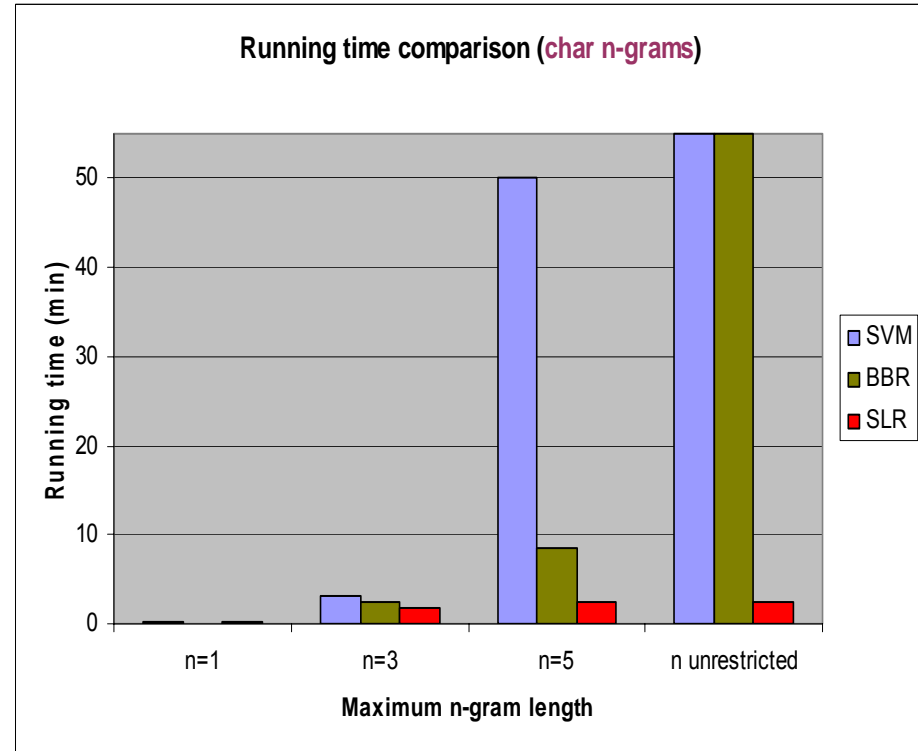
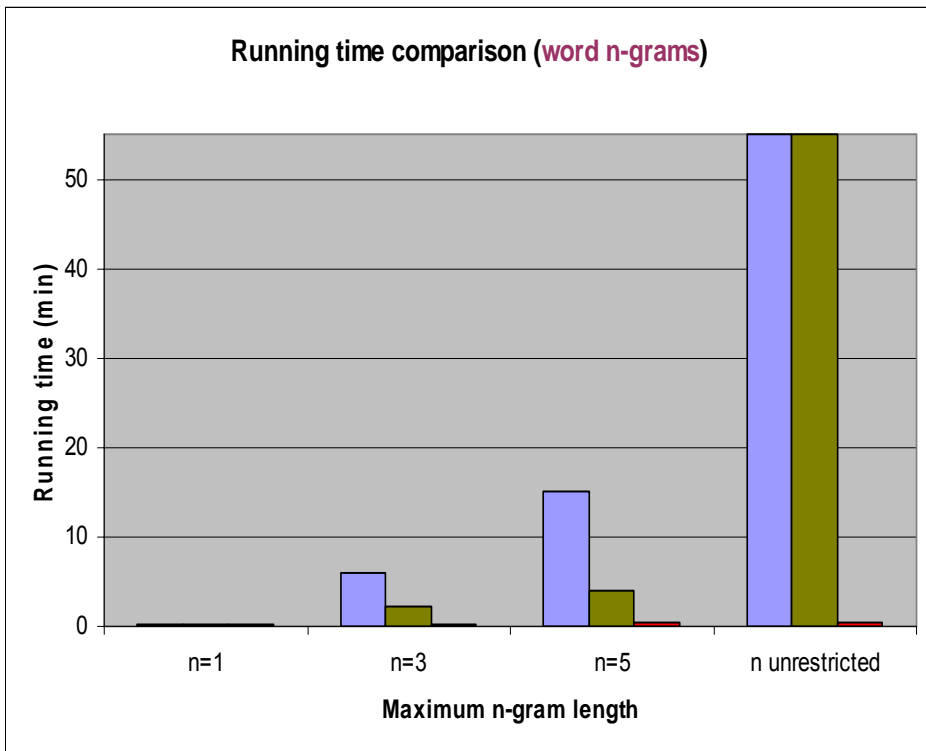
Macro-avg F1 comparison (char n-grams)



- SLR as good as the state-of-the-art in terms of generalization ability

# IMDB – Running time

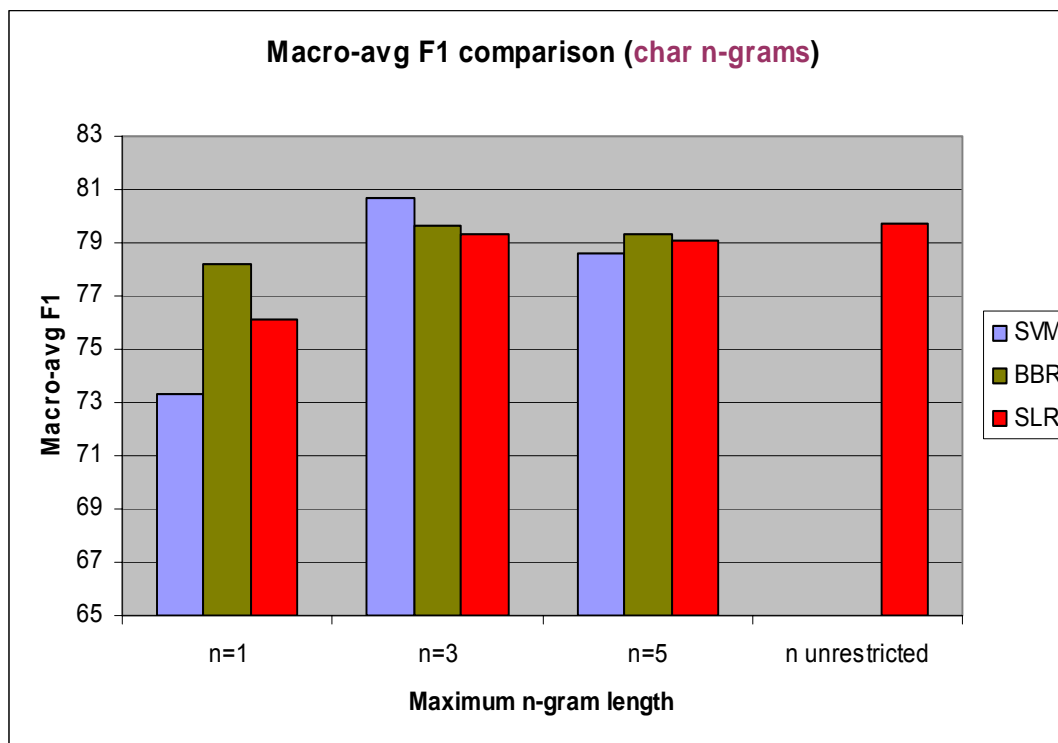
- 5-fold CV; average running time per CV split averaged over topics
- $n$  = max n-gram length, i.e. all n-grams up to length  $n$



- SLR more than one order of magnitude faster than both SVM and BBR

# CHINESE – Macro-avg F1

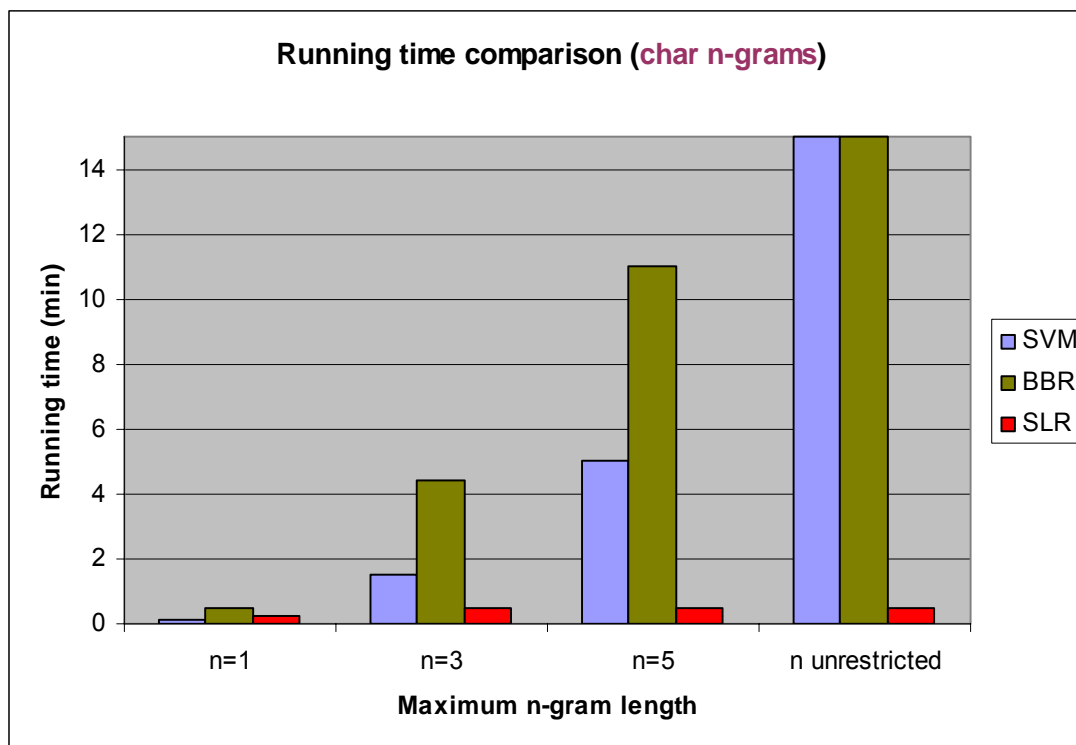
- $n$  = max n-gram length, i.e. all n-grams up to length  $n$



- SLR as good as the state-of-the-art in terms of generalization ability
- Char n-grams avoid the need for word segmentation in Asian text

# CHINESE – Running time

- Training running time
- $n$  = max n-gram length, i.e. all n-grams up to length  $n$



- SLR more than one order of magnitude faster than both SVM and BBR

# Interpretability – A look at the models

---

## IMDB: Crime vs Drama

### word n-grams

### char n-grams

#### Crime n-grams

0.0078 gang war

0.0067 in the middle of

0.0058 hired by

0.0048 from prison

0.040 urde

0.039 gang

0.038 olice

0.034 crime

#### Drama n-grams

-0.0069 they meet

-0.0066 group of people

-0.0015 finds himself

-0.0014 an affair with

-0.020 lov

-0.015 otion

-0.014 urney

-0.011 choo

- Char n-gram models select characteristic substrings (implicit stemming, robustness to morphological variations)

# Conclusion

---

- ❑ **S**tructured **L**ogistic **R**egression: a new algorithm for efficiently learning a classifier with unbounded n-grams
- ❑ No prior feature selection needed
- ❑ No restriction on the n-gram length
- ❑ Using n-grams for text classification is beneficial and can be done efficiently

# Future Work

---

- ❑ Looking at other text categorization applications (spam filtering, e-mail categorization, etc.)
- ❑ Applying this model to supervised information extraction
- ❑ Carrying the same gradient projection ideas to other learning problems

---

**Thank you!**

Open source software for SLR:  
<http://www.mpi-inf.mpg.de/~ifrim/slr>