Hadoop/Mapreduce in NLP and machine learning

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center

Machine learning with Hadoop

Map-Reduce for Machine Learning on Multicore

Cheng-Tao Chu *
chengtao@stanford.edu

Sang Kyun Kim *
skkim38@stanford.edu

Yi-An Lin *
ianl@stanford.edu

YuanYuan Yu *
yuanyuan@stanford.edu

Gary Bradski *†
garybradski@gmail

Andrew Y. Ng *
ang@cs.stanford.edu

Kunle Olukotun *
kunle@cs.stanford.edu

* CS. Department, Stanford University 353 Serra Mall,
Stanford University, Stanford CA 94305-9025.
† Rexee Inc.

Abstract

We are at the beginning of the multicore era. Computers will have increasingly many cores (processors), but there is still no good programming framework for these architectures, and thus no simple and unified way for machine learning to take advantage of the potential speed up. In this paper, we develop a broadly applicable parallel programming method, one that is easily applied to *many* different learning algorithms. Our work is in distinct contrast to the tradition in machine

Outline

Many machine learning algorithms fit Kearns'
Statistical Query Model:
Linear regression, k-means, Naive Bayes, SVM, EM, PCA, backprop

These can all be written (exactly) in a **summation form**

This summation form can be **easily parallelized**,
leading to a linear speedup in the number of processors
(although specialized solutions may be faster for specific cases)

Kearns' Statistical Query Model

Given a function *f(x,y)* over instances (data points *x* and labels *y*), a **statistical oracle** will return an estimate of the expectation of *f(x,y)*

Any model that computes gradients or sufficient statistics over *f(x,y)* fits this model

Typically this is achieved by summing over the data.

Linear Regression

Each data point is an n-dimensional vector $x_i = (x_{i1}, ..x_{in})$, associated with a real valued target label y_i.
A data set $D = \{(x, y)\}$ of m such data points defines a $m \times n$ dimensional matrix X and an m-dimensional vector $\vec{y}$.

Linear Regression: Find the parameter vector θ^* such that:

$$\begin{aligned} \vec{y} &= \theta^T X \\ \text{i.e.} \quad \theta^* &= \frac{X^T \vec{y}}{X^T X} \end{aligned}$$

$$\begin{aligned} \text{Compute} \quad X^T X &= \sum_{i=1}^{m} (x_i x_i^T) \\ \text{and} \quad X^T \vec{y} &= \sum_{i=1}^{m} (x_i y_i) \end{aligned}$$

Summation Form

Naive Bayes

Each data point is an n-dimensional vector $x_i = (x_{i1}, ..x_{in})$, associated with a binary target label $y_i \in \{0.1\}$.
A data set $D = \{(x, y)\}$ of m such data points defines a $m \times n$ dimensional matrix X and an m-dimensional vector $\vec{y}$.

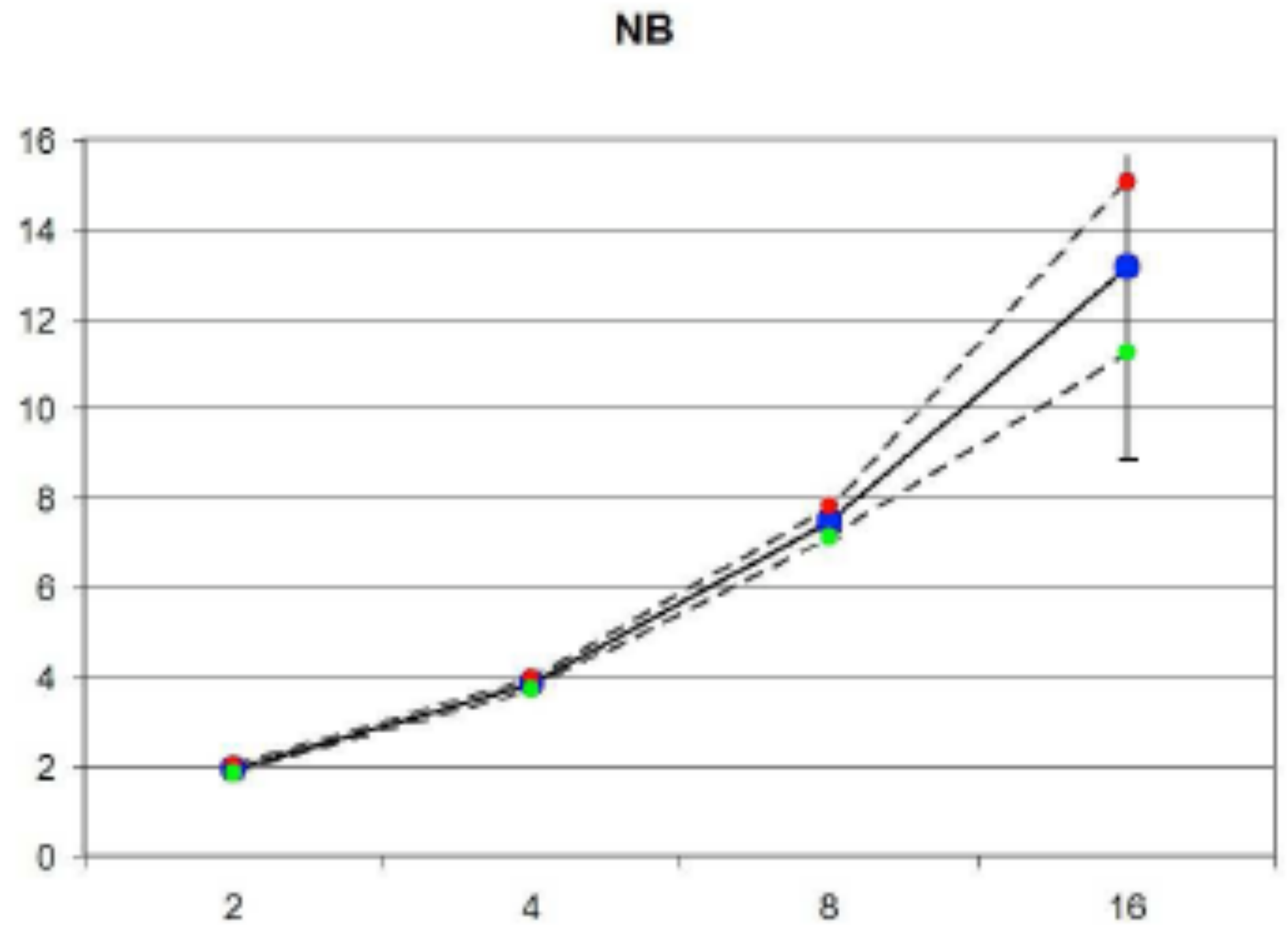
Naive Bayes:

$$\begin{aligned} y^* &= \operatorname*{argmax}_y P(y|x_i) \\ &= \operatorname*{argmax}_y \frac{P(y, x_i)}{P(x_i)} \\ &= \operatorname*{argmax}_y P(y, x_i) \\ &= \operatorname*{argmax}_y P(y)P(x_i|y) \\ &=_{def} \operatorname*{argmax}_y P(y) \prod_j P(x_{ij}|y) \end{aligned}$$

Speedups

On a dual-core machine: >1.9 for most data sets/algorithms

Multiprocessor -- here Naive Bayes example:
(slightly more speedup with multicore)



MapReduce: Distributed Computing for Machine Learning

Dan Gillick, Arlo Faria, John DeNero

December 18, 2006

Abstract

We use Hadoop, an open-source implementation of Google's distributed file system and the MapReduce framework for distributed data processing, on modestly-sized compute clusters to evaluate its efficacy for standard machine learning tasks. We show benchmark performance on searching and sorting tasks to investigate the effects of various system configurations. We also distinguish classes of machine-learning problems that are reasonable to address within the MapReduce framework, and offer improvements to the Hadoop implementation. We conclude that MapReduce is a good choice for basic operations on large datasets, although there are complications to be addressed for more complex machine learning tasks.

Good news and bad news

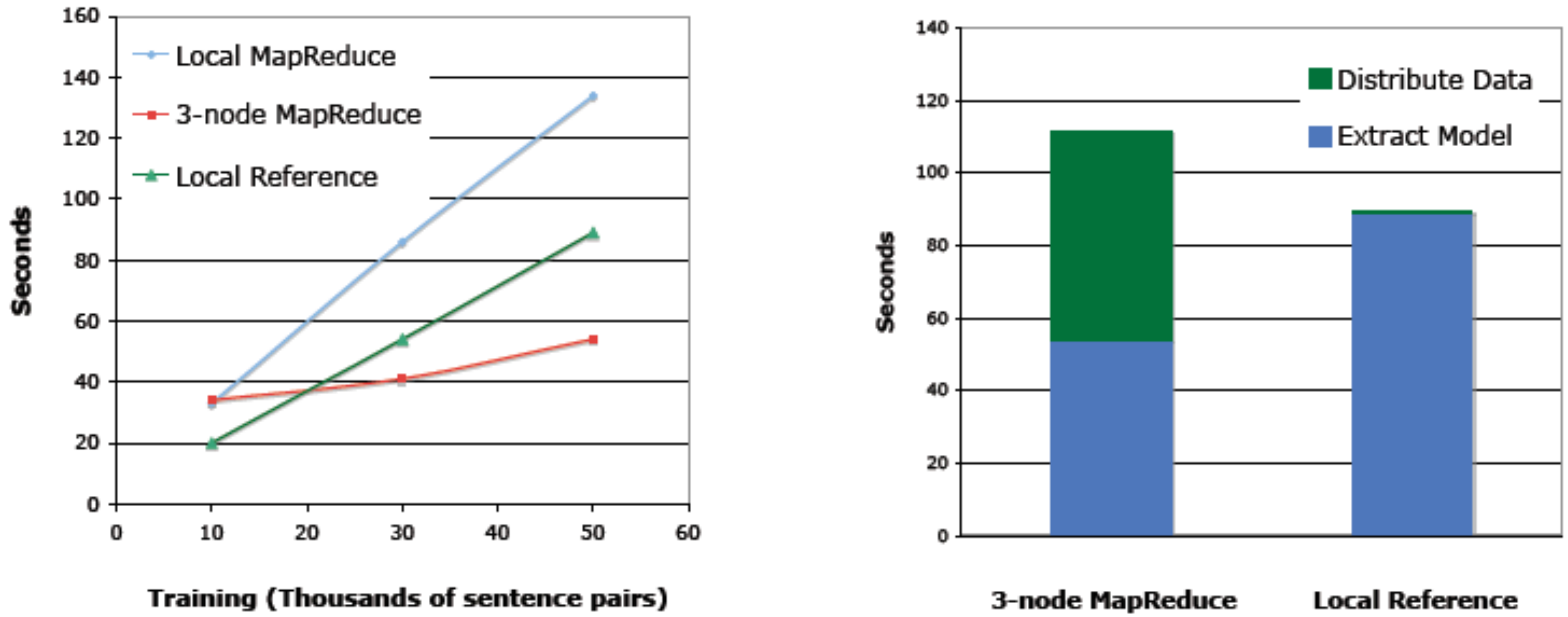


Figure 3: Generating syntactic translation models with Hadoop: (left) The benefit of distributed computation quickly outweighs the overhead of a MapReduce implementation on a 3-node cluster. (right) Exporting the data to the distributed file system incurs cost nearly equal to that of the computation itself.

Bad news for complex NLP applications:
Each data point (sentence) is represented in an application specific manner as a complex object (with parse tree etc.).
Transfer to DFS results in significant slowdown.

Wishlist

- Shared space for map tasks on name node for efficient distribution and referencing of common data (e.g. current model parameters)

- Parallel files (e.g. different representations of the same input) need to be easily tied together in DFS. Data sets should reside permanently on the DFS to be combined/ used in an ad-hoc manner.

NLP applications:
Machine translation

Large Language Models in Machine Translation

Thorsten Brants **Ashok C. Popat** **Peng Xu** **Franz J. Och** **Jeffrey Dean**

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94303, USA
{brants,popat,xp,och,jeff}@google.com

Google Translate

Very large statistical machine translation system.
Can be accessed in real time.

Implemented on DFS (both for estimation and decoding)

Challenge: network latencies result in constant overhead on the order of milliseconds per query.
Solution: fast decoding with batch processing.

N-gram language models

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i | w_1^{i-1}) \approx \prod_{i=1}^{L} \hat{P}(w_i | w_{i-n+1}^{i-1})$$

$$r(w_i | w_{i-n+1}^{i-1}) \quad = \quad \frac{f(w_{i-n+1}^{i})}{f(w_{i-n+1}^{i-1})}.$$

How many n-grams are on the web?

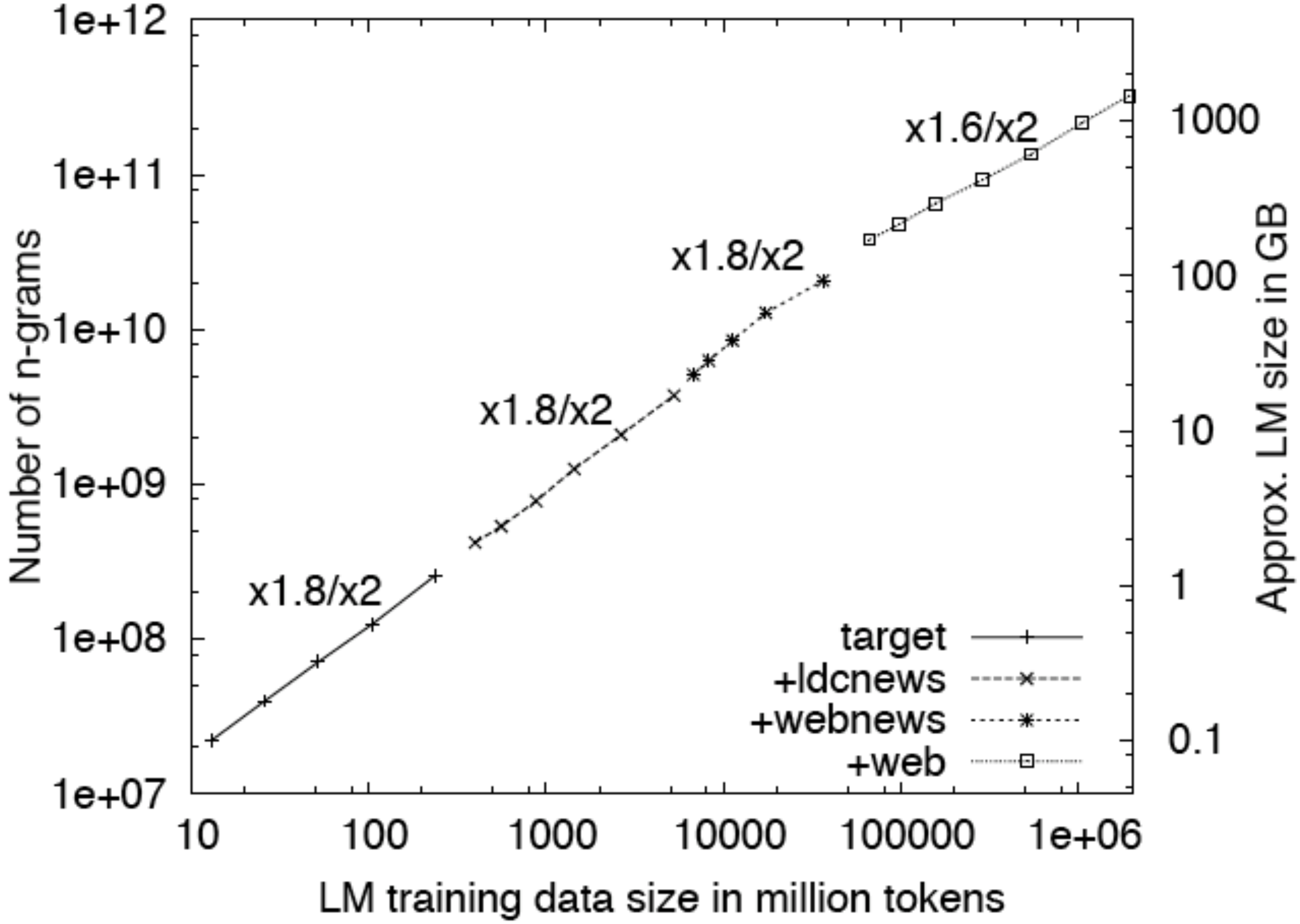


Figure 3: Number of n-grams (sum of unigrams to 5-grams) for varying amounts of training data.

Training times

	target	*webnews*	*web*
# tokens	237M	31G	1.8T
vocab size	200k	5M	16M
# n-grams	257M	21G	300G
LM size (SB)	2G	89G	1.8T
time (SB)	20 min	8 hours	1 day
time (KN)	2.5 hours	2 days	–
# machines	100	400	1500

Table 2: Sizes and approximate training times for 3 language models with Stupid Backoff (SB) and Kneser-Ney Smoothing (KN).

Does this data help?

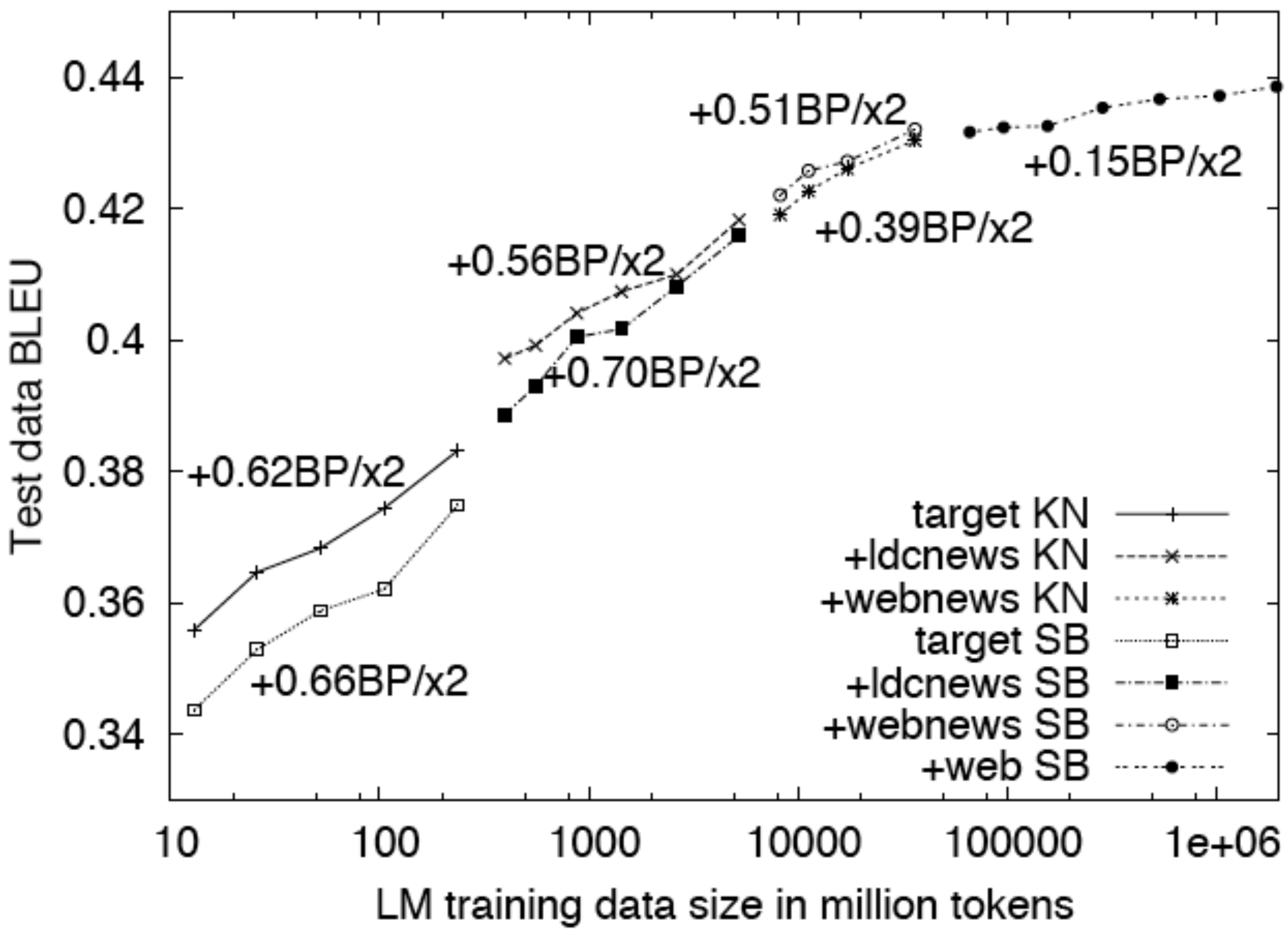
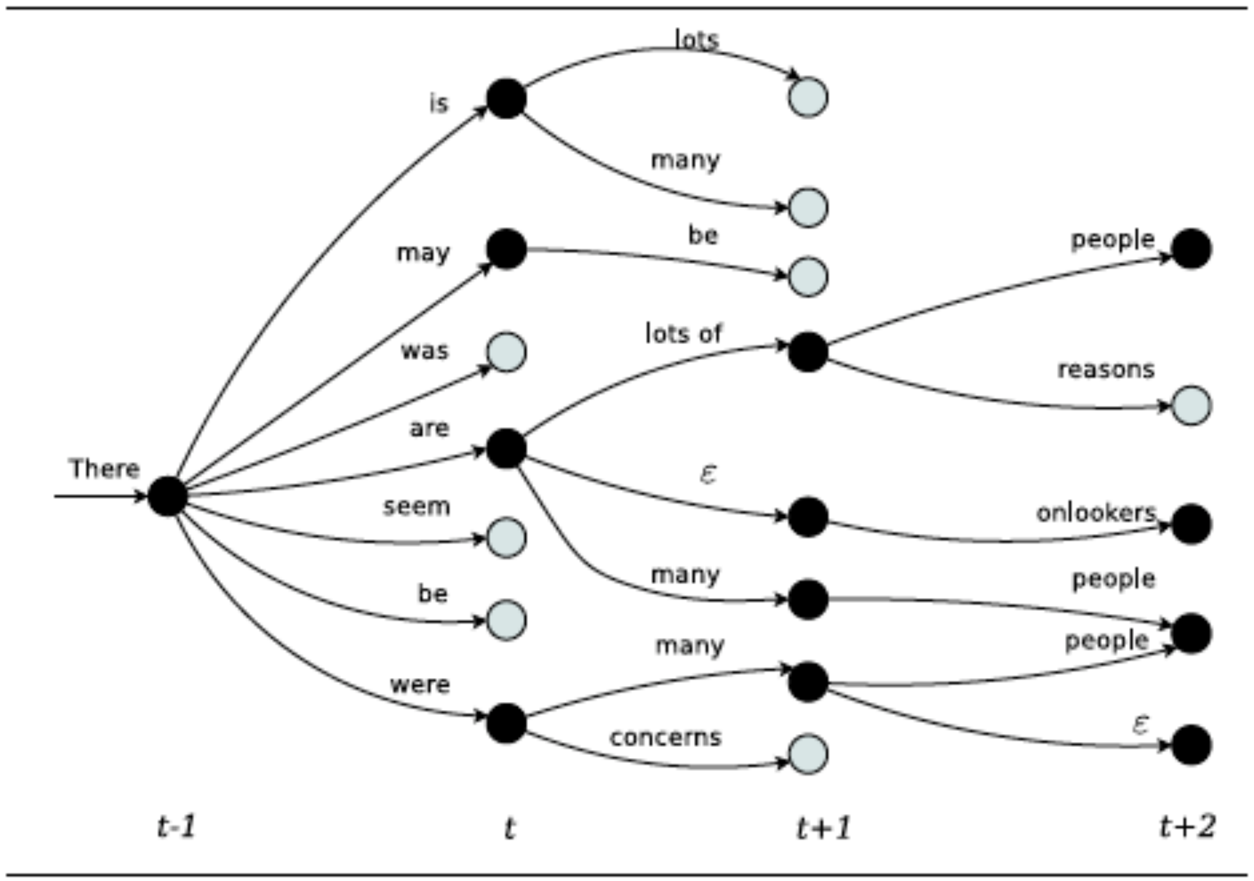


Figure 5: BLEU scores for varying amounts of data using Kneser-Ney (KN) and Stupid Backoff (SB).

Batch decoding

Maintain fixed list of **active hypothesis** (black) at each time step *t*.

Request *all* required n-gram probabilities at time t+1 in batch mode, then update the probabilities, and prune.



Fast, Easy, and Cheap: Construction of Statistical Machine Translation Models with MapReduce

Christopher Dyer, Aaron Cordova, Alex Mont, Jimmy Lin
Laboratory for Computational Linguistics and Information Processing
University of Maryland
College Park, MD 20742, USA
redpony@umd.edu

Parallel Implementations of Word Alignment Tool

Qin Gao and **Stephan Vogel**
Language Technology Institution
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
{qing, stephan.vogel}@cs.cmu.edu

Information extraction/ retrieval applications

Inducing Gazetteers for Named Entity Recognition by Large-scale Clustering of Dependency Relations

Jun'ichi Kazama
Japan Advanced Institute of Science and Technology (JAIST),
Asahidai 1-1, Nomi,
Ishikawa, 923-1292 Japan
kazama@jaist.ac.jp

Kentaro Torisawa
National Institute of Information and Communications Technology (NICT),
3-5 Hikaridai, Seika-cho, Soraku-gun,
Kyoto, 619-0289 Japan
torisawa@nict.go.jp

Gazetteers for Named Entity Recognition

Gazetteer: a list of names (place names, family names)
Can we learn such lists automatically from the web?

Class 791	Class 2760
ウィンダム	マリン/スタジアム
(WINDOM)	(Chiba Marine Stadium [abb.])
カムリ	大阪/ドーム
(CAMRY)	(Osaka Dome)
ディアマンテ	ナゴ/ド
(DIAMANTE)	(Nagoya Dome [abb.])
オデッセイ	福岡/ドーム
(ODYSSEY)	(Fukuoka Dome)
インスパイア	大阪/球場
(INSPIRE)	(Osaka Stadium)
スイフト	ハマ/スタ
(SWIFT)	(Yokohama Stadium [abb.])

Figure 2: Clean MN clusters with named entity entries (Left: car brand names. Right: stadium names). Names are sorted on the basis of $p(c|n)$. Stadium names are examples of multi-word nouns (word boundaries are indicated by "/") and also include abbreviated expressions (marked by [abb.]) .

Clustering by dependency relations

Verbs (*eat, read*) tend to co-occur with specific kinds of arguments:
PEOPLE/ANIMALS *eat* FOOD

Can we cluster nouns based on which verbs they occur with?
This requires parsing.

Here: parse 78M web documents, then cluster with MapReduce.

Pairwise Document Similarity in Large Collections with MapReduce

Tamer Elsayed,[*] Jimmy Lin,[†] and Douglas W. Oard[†]
Human Language Technology Center of Excellence and
UMIACS Laboratory for Computational Linguistics and Information Processing
University of Maryland, College Park, MD 20742
{telsayed, jimmylin, oard}@umd.edu

Aligning Needles in a Haystack: Paraphrase Acquisition Across the Web

Marius Paşca and Péter Dienes

Google Inc.,
1600 Amphitheatre Parkway,
Mountain View, California, 94043, USA
{mars, dienes}@google.com

Abstract. This paper presents a lightweight method for unsupervised extraction of paraphrases from arbitrary textual Web documents. The

Google News Personalization: Scalable Online Collaborative Filtering

Abhinandan Das
Google Inc.
1600 Amphitheatre Pkwy,
Mountain View, CA 94043
abhinandan@google.com

Mayur Datar
Google Inc.
1600 Amphitheatre Pkwy,
Mountain View, CA 94043
mayur@google.com

Ashutosh Garg
Google Inc.
1600 Amphitheatre Pkwy,
Mountain View, CA 94043
ashutosh@google.com

Shyam Rajaram
University of Illinois at Urbana Champaign
Urbana, IL 61801
rajaram1@ifp.uiuc.edu

The recommender problem

Given: the click history for N users over M news items, and a specific user u_i with click history, recommend K news stories

Content-based recommendation: topic alone is not sufficient (and does not generalize, e.g. to video etc.)

Recommender systems: use the ratings of other users (here, binary: click/no click)

Google News Recommendation

Assign users to clusters, and assign weights to stories based on the ratings of the users in that cluster.

$$r_{u_a, s_k} \propto \sum_{c_i : u_a \in c_i} w(u_a, c_i) \sum_{u_j : u_j \in c_i} I_{(u_j, s_k)}$$

News

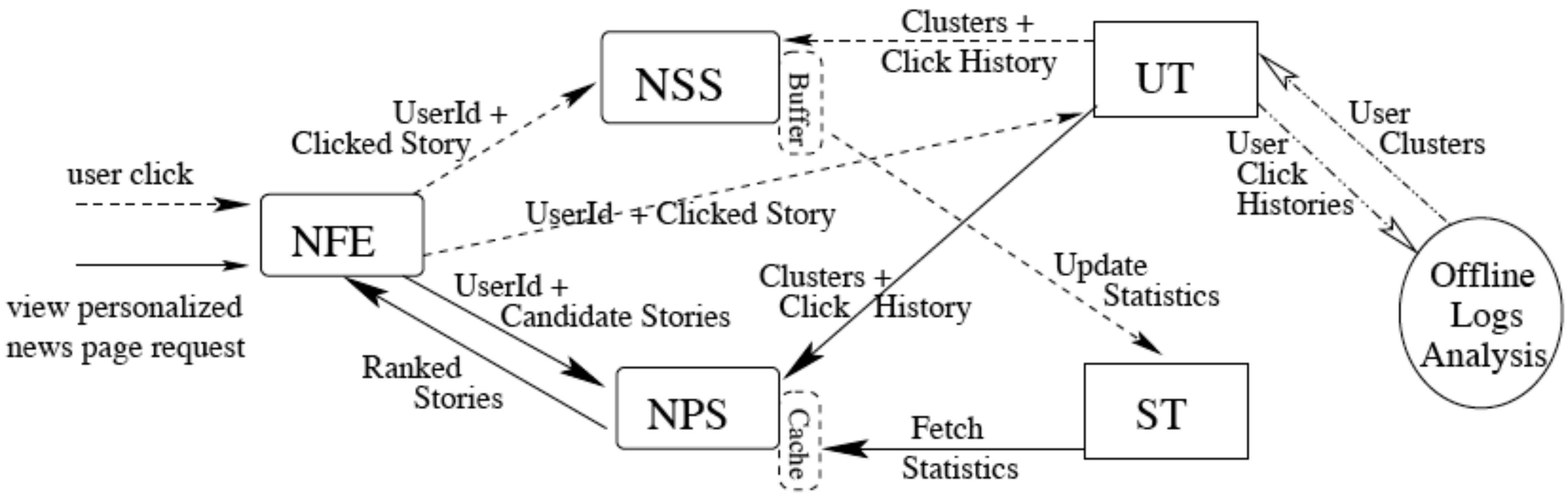


Figure 2: System Components

NFE: News Front End

NSS: News Statistics Server

NPS: News Personalization Server

ST: Story Table

UT: User Table