Linear-chain CRF for NLP in MADlib

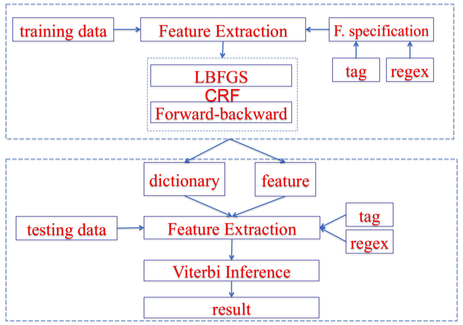
KUN LI[§], CHRISTAN GRANT[§], DAISY ZHE WANG[§]
[§]*Database Research Center, CISE, University of Florida*
{kli, cgrant, daisyw} @cise.ufl.edu

UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

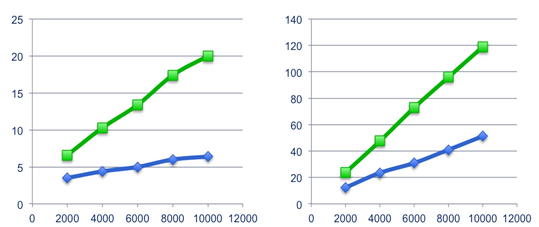
ABSTRACT

MADlib is an open-source library for scalable in-database analytics developed by EMC, UC Berkeley, University of Wisconsin and our group in University of Florida. It provides an evolving suite of data-parallel implementations of mathematical, statistical and machine-learning methods for structured and unstructured data. We are contributing a linear-chain conditional random field for natural language processing such as part of speech tagging(POS), named entity resolution(NER). Motivation:in database analysis.support interactive query. time critical, leverage parallel database,e,g Greenplum. Instead of extracting features for each token on fly, we extract features for each distinct token and materlized it in the database.

SYSTEM OVERVIEW



PERFORMANCE EVALUATION



ACKNOWLEDGEMENTS

This work is supported by EMC/Greenplum. Thanks to EMC/Greenplum employee Caleb Welton and Hitoshi Harada s precious feedback and code review.

PARALLEL TEXT FEATURE EXTRACTION

Text feature extraction is a step in most statistical text analysis methods, and it can be an expensive operation. To achieve high quality, CRF methods often assign hundreds of features to each token in the document. Examples of such features include: (1) dictionary features: does this token exist in a provided dictionary? (2) regex features: does this token match a provided regular expression? (3) edge features: is the label of a token correlated with the label of a previous token? (4) word features: does this the token appear in the training data? and (5) position features: is this token the ïňĄrst or last in the token sequence? The right combination of features depends on the application, and so our support for feature extraction heavily leverages the microprogramming interface provided by MADlib.

PARALLEL LINEAR-CHAIN CRF TRAINING

CRF training is time consuming due to the expensive forward-backward computation to evaluate the log-likelihood function and its gradient vector for each interation.
$\ell'_\lambda = \sum_k [\lambda F(y_k, x_k) - \log Z_\lambda(x_k)] - \frac{\lambda^2}{2\sigma^2} + const$
with gradient
$\nabla \ell'_\lambda = \sum_k [\lambda F(y_k, x_k) - E_{p\lambda(Y|x_k)} F(Y, x_k)] - \frac{\lambda}{\sigma^2}$
parallel forward-backward computation

PARALLEL VITERBI INFERENCE

The Viterbi dynamic programming algorithm [14] is the popular algorithm to ïňĄnd the top-k most likely labelings of a document for (linear chain) CRF models. Like any dynamic programming algorithm, the Viterbi algorithm is recursive. We experimented with two direct implementations of macro-coordination over time. First, we chose to implement it using a combination of recursive SQL and window aggregate functions. We discussed this implementation at some length in earlier work [42]. Our initial recursive SQL implementation only runs over PostgreSQL versions 8.4 and later; it does not run in Greenplum. Second, we chose to implement a Python UDF that uses iterations to drive the recursion in Viterbi. This iterative implementation runs over both PostgreSQL and Greenplum. In Greenplum, Viterbi can be run in parallel over different subsets of the document on a multi-core machine.We want to find the sequence of tags that maximizes the formula.
$P(T_1, T_2, ..., T_n | W_1, W_2, ..., W_n)$ which can be estimated as $\prod_{1 \le i \le n} (P(T_i|T_{i-1}) * P(W_i|T_i))$
$P(T_i|T_{i-1}$ is the transition weight
$P(W_i|T_i)$ is the emissing weight
-Create tables and import model data to the database $SELECT madlib.load\_crf\_model('/path/to/modeldata')$
–Feature extraction $SELECT madlib.text\_feature\_extraction$
–Viterbi inference
$SELECT madlib.vcrf\_label(...)$ Sample queries
$Q_1 : Tagging for setence with id = 501$
$SELECT doc\_id, madlib.vcrf\_top1\_label(mfactors.score, rfactors.score)$
$FROM m\_factors mfactors, r_factors rfactor;$
$Q_2 : Tagging for all sentences containing the word 'professor'$
$SELECT DISTINCT ON doc\_id, madlib.vcrf\_top1\_label(mfactors.score, rfactors.score)$
$FROM mfactor, rfactori, seg\_tbl, segmenttbl$
$WHERE segtbl.doc\_id = segmenttbl.doc\_id AND segmenttbl.seg\_text = ' professor'$

REFERENCE

[1] http://crf.sourceforge.net/
http://www.cs.berkeley.edu/ daisyw/ViterbiCRF.html
Xuan-Hieu Phan, Le-Minh Nguyen, and Cam-Tu Nguyen. FlexCRFs: Flexible Conditional Random Fields, 2004.