A Proposal Optimizer for Sampling-based Entity Resolution

Christan Earl Grant
University of Florida
Dept. of Computer Science
Gainesville, Florida, USA
cgrant@cise.ufl.edu

Daisy Zhe Wang
University of Florida
Dept. of Computer Science
Gainesville, Florida, USA
daisyw@cise.ufl.edu

ABSTRACT

1. INTRODUCTION

The storage of user generated content within systems has introduced vast amounts of data. To correctly process this data must be cleaned. Entity resolution is an important part of the ubiquitous cleaning task. Entity Resolution (ER) is the problem of resolving records in a data set that correspond the same real world entity.

Entity resolution is a notoriously computationally difficult problem. Several efforts in different domains have made outstanding progress []. The main issues still affecting runtimes of ER systems are twofold, first, the computation of large entities and second, excessive computation spent resolving unambiguous entities. Optimization that touches these critical portions is wholly understudied. We argue that compression and approximation techniques can efficiently decrease the runtimes of traditional ER systems.

There is not one size fits all techniques even inside sampling algorithms [**?**].

Some recently, researchers have suggest methods of compressing entities. Wick et al Heirchical ...

Singh et all efficient factoring

Each of these methods has drawbacks ...

In this paper, we train a multi-class classifier to optimize the decision of the sampling inference technique to apply.

We make the following contributions

- We identify several techniques to speed up sampling past the baseline **??**.
- We create an optimizer to choose parameters and methods at run time **??**.
- We empirically evaluate these methods over a large data set **??**.

2. BACKGROUND

Factor Graphs. Factor graphs are a pairwise formalism for expressing arbitrarily complex relationships between random variables. A factor graph $\mathcal{F}$, will contain a set of random variables, r and a set of factors f. Random variables are connected to eachother through factors. Factors represent a mapping of the relationship to a real valued number.

Markov Chain Monte Carlo Metropolis Hastings. Inference over complex factors graphs is computationally prohibitive. Therefore, it is popular for researchers to use Markov Chain Monte Carlo (MCMC) approximation techniques to estimate probability values. In particular, for large dense factor graphs MCMC Metropolis Hastings has been shown to be a scalable and efficive technique for inference calculation [**?**].

... details of mcmc mh ...

Cross-Document Coreference.

Cross-Document coreference is resolving entities across document borders. This problem is usually several orders of magnitude smaller when compared to within document entity resolution. Solution to coreference comes in a variety of techniques. We model entity resolution as a factor graph and use MCMC-MH for flexibility and the ability to generalize the mathematically for other operations.

The distribution of entity sizes In large text corpora, the sizes of entities follows the power law [**?**]. For examples, Figure 1 is a generated data set containing 40 million mentions and 3 million entities over 11 million web pages.

In this data set XX percent of mentions are smaller that 100 mentions in size. Additionally, YY percent of entities contain more than 1000 mentions.

Doing pairwise comparison over clusters is $O(n^2)$. For clusters larger than 1000 mentions calculating scores of the model becomes extremely expensive. Performing sophisticated techniques over smaller clusters could also add extra over head.

In this paper, we examine the trade-off of selecting techniques to accelerate the feature computation process.

3. RELATED WORK

In this paper use use sampling based ER over factor graphs.

Sameer Singh [**?**]

Wick et al perform hierarchal compression of entities [**?**].

As we see, in Section 6.1 the entity sizes have a large effect on the factor computation size.

4. PROBLEM STATEMENT

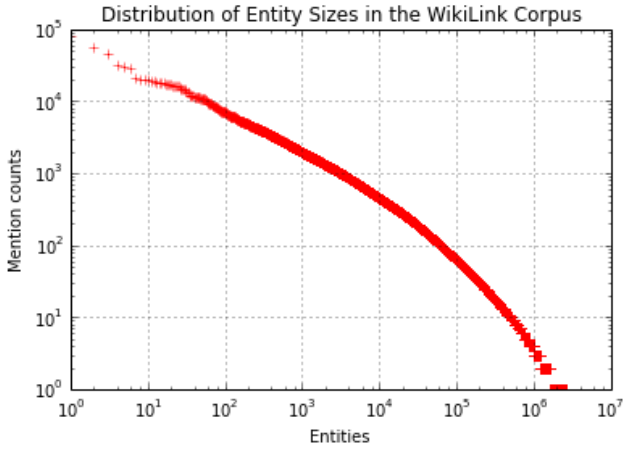

Figure 1: A distribution of entity sizes from the wiki-links corpus [**?**].

In this section we formally define the compression problem for entity resolution.

We have two problems. First, Given an Entity $\mathcal{E}$ how can we create a a structure with minimal total size. Second, How can we incrementally add and remove items from the compressed structure. Third, how can we iterate over or sample from the compressed data structure to perform uncompression.

The total entity size entity is

$$\sum_i C + (sizeof(int) * |m_i|)$$

The smallest compression we can do per entitiy is to only unique entity tokens.

The compressed total entity size size is

$$\sum_i c + (sizeof(int) * \#(m_i))$$

Show Distribution of Entity sizes in the data set

We can take advantage of the redundancy within large entities and compress the entities.

We look to perform add, remove and iterate operations over the compressed entity sets.

5. ALGORITHMS

In this section we will describe XXX algorithms for entity sampling. The main method we use to improve the computation speed is to reduce the number of comparisons we make for large entity clusters. We additionally discuss methods to improve the performance of algorithms over time by collecting statistics from the processes.

Naive Iterator. This method performs pairwise comparisons by iterating over the mentions using the order on disk. This is the traditional method of computing the pairwise similarity of two clusters.

SubSample Iterator. This method performs uniform samples of the mentions from the source and destination entities clusters. This method measures the confidence of the calculated pairwise samples and stops when the confidence exceeds a threshold of 0.95.

Top-K SubSample Iterator. This method uses a priority queue to sort the mentions and only performs comparisons between the top M mentions.

Blocked Iterator. This method performs blocking on the mentions in the source and destination entities and takes the average gain from pairwise comparisons inside each block. Blocks are formed arbitrarily and are of fixed size. In database terms this method is a *block nested loop* comparison.

Blocked Subsample Iterator. This performs a block nested loop comparisom but it performes book keeping to only perform pairwise sampling until we reach a confidence threshold of 0.95.

Blocked Top-k Iterator. This method performs a block nested loop comparison except blocks are created by reading mentions from a priority queue.

Blocked Top-K Subsample Iterator. This method is a combination of the Block subsample iterator and the blocked top-k iterator.

Further, We examine active learning techniques to adjust threshold sizes based on statistics collected while running the algorithms.

We collect the following statistics from each training run:

Success and failure Data set size. Number of uniques tokens. Average pairwise score between mentions. Maximum pairwise score between mentions. Minimum pairwise score between mentions. Variance of the pairwise score between mentions. Mention Token tfidf scores. Cardinality of tokens in both entities. Generailze the feature explaination and move it to the implementation section — CEG

Using this information we train a decision tree classsifier to minimize the pairwise comparisons in the score function. The classifier is also constrained by the accuracy of the decision tree as approximate methods are less accurate. More formally ...

The result of this classifier is two-fold. First, we have a classifier to choose the optimal algorithm for each proposal. Second, we have an active learning feed-back loop for algorithms with thresholds. We emperically study both of these outcomes.

Technique	**Reduces size**	**Lossless**	**location**
Full pairwise	No	Yes	Feature
Early Stopping [**?**]	No	No	Feature
Run-Length Encoding	Yes	Yes	Storage

6. IMPLEMENTATION

In this section we first present a microbenchmark to validate our invsitgation of entity approximation and compression. We then discuss the implementation of the compression and approximation techniques over a large real-world cross-document coreference corpus.

6.1 Microbenchmark

To increase our intuition of early stopping techniques we simulated the MCMC proposal processes. We hypothesis that there was bould a clear range values where performing the baseline cluster sampling would be faster when compared to early stopping methods. We arrange entity clusters of increasing time and we compute the time (in clock ticks) each proposal takes to compute the arrangement of the clusters. The data in the clusters are disctributed uniformly and for this experiment each cluster point was 5 dimensional. For the baseline cluster score computation we used a pairwise calculated of the average cosine distance

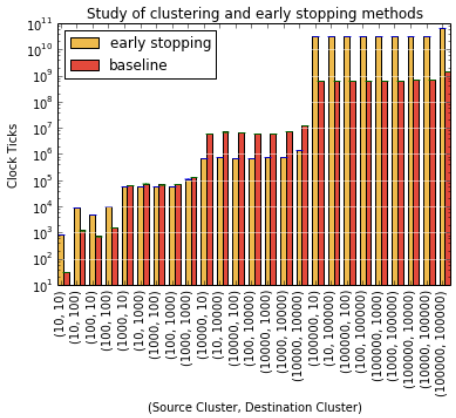

Figure 2: Comparison of baseline verses early stopping methods.

with and without the mention. To compute early stopping we set a confidence threshold to 0.8 and the early stopping code stopped computation when the error predition was under 20%. There was no difference in the proposal choices between the baseline and the early sorting method.

The simulations were developped in `GNU C++11` and compiled with `g++ -O3`. The CPU was an 8 core intel i7 with 3.2 HGz and 12 GBs of Memory. Each arrangement was run 5 times and results averages.

Experiment Results... The result of this experiment is summarized in Figure 2. On the x-axis is the number of mentions in the source and destination clusteris for each proposal. The y-axsis is the number of clock ticks on a log scale.

We observe that for proposals with less than 100 and 1000 source and destination mentions, the performance of the baseline proposer is better than or almost equal to that of the more sorted early stopping method. For proposals that contain an entity cluster with 10000 mentions the early stopping method performs significantly better than the baseline method.

Surprisingly, the baseline proposals for for entities clusters containing $100K$ mentions performed over an order of magniture better than the early stopping method.

The optimization found in predictable code paths make simple implementations like the baseline method attactive for small cluster sizes and very large clusters sizes. In addtion, 82% of the entities in the truthed Wiki-Links data sets are less that 1000 mentions in size and 45% of the entities contain less than 100 mentions.

The results of the microbenchmark suggests that different proposal estimation techniques are useful at different times.

6.2 Wiki Link Corpus

The Wikilinks corpus is the largest fully labeled cross-document coreference resolution data set to date [**?**]. When downloaded, the data set contains 40 million mentions and almost three million entities — it is a compressed 180 GBs of data. The wikilink corpus was created by crawling pages across the web and extracting anchor tags that referenced wikipedia articles. Each page contains multiple multiple mentions of different types. The wikipedia articles act as the truth for each mention.

7. EVALUATION

8. CONCLUSION