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 [1].

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 each other 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 [2].

... 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 [3]. For example, Figure 1 is a generated data set containing 40 million mentions and 3 million entities over 11 million web pages.

The mentions on disk can be represented as a large array of identifiers. Entities are a collection of mentions and can be represented as such. In the worst case there is an equal number of entities and mentions. This means each mention is its own individual entity. In the other extreme, all the mentions may be a part of the same entity.

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 [2]

Wick et al perform hierarchal compression of entities [5].

As we see, in Section 7.1 the entity sizes have a large effect on the factor computation size.

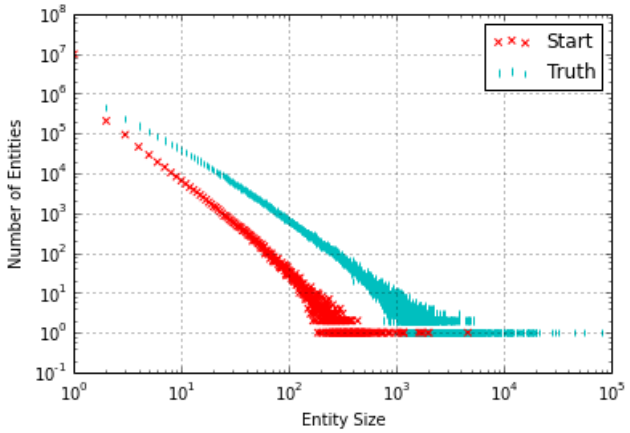

Figure 1: A distribution of entity sizes from the wiki-links corpus [3] with an initial start and the truth.

4. EXAMPLE

In this section we discuss define the acceleration in MCMC-MH sampling for entity resolution. We then motivate how we believe gains can be achieved given using compression, sampling acceleration methods and optimizers.

The issues we are investigating are as follows. First, Given an Entity e_i how can we create a structure with *minimal total size*. Next, how can we compute the features over a source and destination entity (e_s, e_d) in the *smallest amount of time*. Lastly, how do we *decide* when to use each technique.

The total size of all entities in the traditional representation is:

$$\text{sizeof}(\mathcal{E}) = \sum_i c + (\text{sizeof}(\text{int}) * |e_i|), \quad (1)$$

where *sizeof* is an abstract function to compute the size of the containing object, c is a class constant and $|e_i|$ is number of mentions in the entity.

There are many compression techniques, one being we only keep the mentions an entity that have a unique representation. That is, if any mention token is a duplicate, we remove it. This compressed total entity size size is:

$$\text{sizeof}(\mathcal{E}_{\text{compressed}}) = \sum_i c + (\text{sizeof}(int) * \#(e_i)), \quad (2)$$

where $\#(e_i)$ is the cardinality of the mention tokens in entity e_i. We note that when the $\#(e_i) \ll |e_i|$ it may be worth compressing this entity.

In the example data set in Figure 1 we note that 45% percent of mentions are smaller that 100 mentions in size. Additionally, 82% percent of entities contain less than 1000 mentions. These number suggest that at times we we can take advantage of the redundancy within large entities and compress the entities. (We investigate the WikiLinks corpus further in Section 7.1).

In addition, Figure 1 show that there is about an order of magnitude difference between the sizes of initial entities and the true entity sizes. This suggest that there are several unique representations of entities that during entity resolution entities sizes can expect to increase by an order of magnitude in size and smaller entities will decrease. This is to be expected but we can use this property to track the grow and change of entity sizes over time to understand which camp a particular entity cluster belongs. Either it will grow or it will probably shrink.

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 comparison but it performs 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 tf-idf scores. Cardinality of tokens in both entities. Generalize the feature explanation and move it to the implementation section — CEG

Using this information we train a decision tree classifier 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 empirically study both of these outcomes.

Technique	Reduces size	Lossless	locatio
Full pairwise	No	Yes	Feature
Early Stopping [4]	No	No	Feature
Run-Length Encoding	Yes	Yes	Storage

6. OPTIMIZER

When before calculating the MCMC-MH proposal there are several decision we can make that will affect the runtime and accuracy of the algorithm. We could (1) Update the entity structure to or from a compressed format; (2) Select a new way of calculating the pairwise features; (3) Skip the calculation of the proposal and directly accept or reject.

These decision are made by observing several features of a source entity, destination entity and a source mention. These features include the source and destination size, the source and destination cardinality, the number of samples the algorithm has made.

At each proposal step the decision made should maximize the *utility*. Utility of the decision is a numeric score to represent the gain performing the proposal calculation. The utility value is a real number ranged from $(\inf, \int)$.

A formal model for utility is as follows:

$$U = C_{\text{Entity trend}} + C_{\text{pw calculation}} + C_{\text{Entity update}} + C_{\text{Undoing the change}} \quad (3)$$

You should only update the entity structure if it will improve the pairwise feature calculation cost enough.

7. 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.

7.1 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 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.

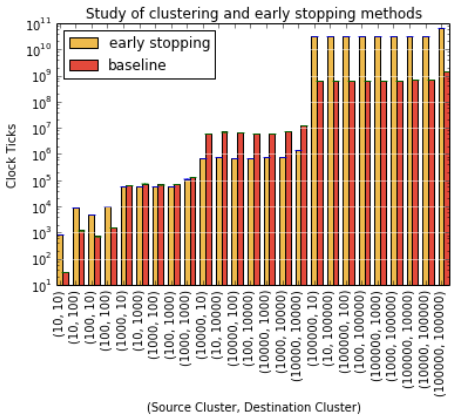

Figure 2: Comparison of baseline verses early stopping methods.

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

The optimization found in predictable code paths make simple implementations like the baseline method attractive for small cluster sizes and very large clusters sizes. In addition, 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.

7.2 Wiki Link Corpus

The Wikilinks corpus is the largest fully labeled cross-document coreference resolution data set to date [3]. 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.

8. EVALUATION

9. CONCLUSION

10. REFERENCES

[1] D. Sculley and C. E. Brodley. Compression and machine learning: A new perspective on feature space vectors. In *Data Compression Conference, 2006. DCC 2006. Proceedings*, pages 332–341. IEEE, 2006.

[2] S. Singh, A. Subramanya, F. Pereira, and A. McCallum. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 793–803. Association for Computational Linguistics, 2011.

[3] S. Singh, A. Subramanya, F. Pereira, and A. McCallum. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst, 2012.

[4] S. Singh, M. Wick, and A. McCallum. Monte carlo mcmc: efficient inference by approximate sampling. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1104–1113. Association for Computational Linguistics, 2012.

[5] M. Wick, S. Singh, and A. McCallum. A discriminative hierarchical model for fast coreference at large scale. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 379–388, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.