Fast and Accurate Incremental Entity Resolution Relative to an Entity Knowledge Base

Michael Welch∗
Barnes & Noble
Palo Alto, CA 94301
mjwelch@cs.ucla.edu

Aamod Sane
Yahoo! Inc.
Sunnyvale, CA 94089
aamod@yahoo-inc.com

Chris Drome
Yahoo! Inc.
Sunnyvale, CA 94089
cdrome@yahoo-inc.com

ABSTRACT
User facing topical web applications such as events or shopping sites rely on large collections of data records about real world entities that are updated at varying latencies ranging from days to seconds. For example, event venue details are changed relatively infrequently whereas ticket pricing and availability for an event is often updated in near-realtime. Users regard these sites as high quality if they seldom show duplicates, the URLs are stable, and their content is fresh, so it is important to resolve duplicate entity records with high quality and low latencies. High quality entity resolution typically evaluates the entire record corpus for similar record clusters at the cost of latency, while low latency resolution examines the least possible entities to keep time to a minimum, even at the cost of quality. In this paper we show how to keep low latency while achieving high quality, combining the best of both approaches: given an entity to be resolved, our incremental Fastpath system, in a matter of milliseconds, makes approximately the same decisions that the underlying batch system would have made. Our experiments show that the Fastpath system makes matching decisions for previously unseen entities with 90% precision and 98% recall relative to batch decisions, with latencies under 20ms on commodity hardware.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

Keywords
Entity Resolution, Knowledge Base

1. INTRODUCTION
The problem of deduplication and linking data records that refer to real world entities arises in several contexts. For example, topical sites that specialize in travel, real estate, or hyperlocal collect and organize information of interest so that users can browse as well as search. Users are more likely to frequent those sites that offer a wide variety of fresh information in a well organized fashion. The objectives of information providers, however, do not always coincide with these goals. In addition to inadvertent errors and noisy data, a business owner, for example, might prefer to have their listing shown multiple times and thus submit several variations of their information. The onus is on the sites to deduplicate, enrich, and link information from multiple such data sources for the users.

In principle, entity resolution over a corpus of \(n\) entities is possible using \(O(n^2)\) pairwise comparisons, but since this becomes impractical as \(n\) grows, blocking techniques [9, 11, 8] are used to partition entities into blocks of similar records using hashes called blocking keys. Pairwise comparison [3] is then limited to blocks of size \(b\) such that \(O(b^2) \ll O(n^2)\). Most such research has been directed toward improved resolution quality, without considering the issues of overall entity resolution latency. Low latency resolution using search techniques has been recently studied by [4], and researchers in linguistics [13] have considered streaming deduplication. However, this research has focused on per entity resolution latency, without considering resolution quality relative to full corpus processing.

We employ entity resolution to synthesize a knowledge base that is being used in several of Yahoo’s topical web sites. The knowledge base is created from data sources of varying quality, ranging from high quality proprietary data to very noisy sources such as user supplied information. In this setting, to achieve the best quality, we need a full corpus entity resolution that can account for effects such as resolution cascades. On the other hand, sites such as event listings and shopping deals are most useful when the data they provide is also fresh. We therefore also need low latency deduplication of new records. To achieve low latency, we may opt to examine only a portion of the entity graph.

In this paper we describe the Fastpath system, which leverages the high quality deduplication decisions of a batch system as a basis for deduplicating records incrementally in real-time with high quality. To the best of our knowledge, ours is the first entity resolution system that achieves both high quality and low latency.

2. THE FASTPATH SYSTEM
The Batchpath is a Hadoop based system built at Yahoo! designed to ingest records in different formats from multi-
ple sources and generate a single, unified semantic graph of deduplicated records called an entity knowledge base.

The key stage of the Batchpath system is the Builder, which deduplicates the input records in a two step process. The Builder first partitions the data set into blocks of similar records, so that instead of comparing $O(n^2)$ objects, we need only compare $O(b^2)$ objects [7], provided blocking functions separate dissimilar records. Blocking functions are data dependent, for instance using n-grams for large text fields, numeric comparisons for telephone numbers, and so on.

In the second Builder step, within each block we perform a more thorough similarity comparison between all pairs of objects. This pairwise similarity measure operates on the same parcel used for blocking, though it need not use the same set of attributes. The similarity score between each pair of objects in a block creates a connected component graph with weighted edges. Pairs whose similarity score is above a certain threshold are considered similar. Subsequent stages of the Builder refine these connected components, splitting or merging them to form a final set of distinct entities. Full details of the Batchpath system can be found in another paper [2].

Fastpath uses a search engine to select matching records for a deduplication decision. We allow for two common incremental operations: (1) either the query object is added to an existing cluster, or (2) it forms a new cluster. This eliminates some of the complexity seen in the full graph, such as a merge or split operation potentially cascading changes to neighboring objects, while still providing high accuracy deduplication in most cases.

Fastpath reuses several components of Batchpath, including (1) the same blocking functions to generate blocking keys for the query record, (2) the same entity blocking keys as terms in the search engine, (3) the same pairwise comparison functions, and (4) the same matching thresholds to select the best matching entity.

We first populate a search index for the knowledge base created by Batchpath, where each object is indexed by its blocking keys. At runtime, we apply the same blocking function to the incoming query record and find candidate matching entities in the index. We use multiple blocking functions in the Batchpath to ensure that we do not miss any entities that we should have compared [9], thus improving recall. Using these same keys as terms for a query record allows the search engine to use standard text ranking approaches to provide an initial candidate ranking.

We compute the pairwise similarity score between the query record and each of the top-k candidate entities, and select the entity with the highest score $S_{\text{max}}$. If $S_{\text{max}}$ is above a configurable threshold (typically, the same threshold value used during entity cluster refinement in the Batchpath), we consider the entity as a match and return it as the result.

3. EXPERIMENTAL RESULTS

In our experiments we use a subset of data which is currently serving the local event listings\(^1\) for Yahoo. Our experiments were performed on a dual-core 2.4 Ghz Intel Core2 machine with 4GB of memory running Linux. This machine hosts both the Fastpath logic and the text retrieval engine, using a proprietary text search engine framework. The search engine allows us to plug in ranking and matching components written in Java that we share in the Hadoop-based Batchpath system.

Our input data consists of approximately 750k records with attributes and relationships that represent knowledge about local events, such as details of the event venues, performers or participants, event genre, ticket prices, starting and ending times, and so on. The data is collected from both curated data providers as well as crawled and extracted from the Web using modern information extraction techniques.

3.1 Experimental dataset

Our input data consists of approximately 750k records with attributes and relationships that represent knowledge about local events, such as details of the event venues, performers or participants, event genre, ticket prices, starting and ending times, and so on. The data is collected from both curated data providers as well as crawled and extracted from the Web using modern information extraction techniques.

3.2 Querying a complete knowledge base

In our first experiment, we evaluate the effect of varying the number of candidate objects $k$ to retrieve using the search engine. We use a weighted term matching function similar to TF-IDF, commonly found in text retrieval applications, as the candidate ranking strategy.

After constructing a knowledge base from the full 750,000 record input data set, we randomly selected 20,000 of those records to submit as queries and evaluate the accuracy of the top result after re-ranking. The result is considered a true positive match if the query object is one of the objects in the top ranked object cluster returned. The other two possibilities are a false positive match (matching the incorrect

\(^{1}\)An event may be a professional sports game, theatrical performance, a city council meeting, farmers market, etc.
cluster) and a false negative match (returning no result).
There are no “true negative” results, as each query should
match an entity. We expect high but not necessarily perfect
precision or recall, as the Fastpath system only has a partial
view of the entity graph.

Figures 1 and 2 shows how varying values of \( k \) effect precision,
recall, F-measure, and average query execution time
of the system. We use the standard definition of precision
and recall for classification tasks. As we consider additional
candidate entities for deeper evaluation, the recall and query
execution times increase, while the precision remains
effectively constant. This suggests that the re-ranking applied
to the retrieved entities nearly always finds the “right” can-
didate out of the available choices. Retrieving additional
candidates generally improves the accuracy of the system
up until approximately \( k = 10 \), above which we see only
insignificant improvements in recall and F-measure. This
shows the initial search engine ranking generally captures
the correct result in the top 10.

When considering the top \( k = 10 \) candidate entities se-
lected by term ranking, false positives, or queries where the
Fastpath matches an object to the incorrect cluster, occur
less than 1% of the time. The false negative rate is also
low at 1.33%. Collectively this means customers using the
Fastpath system will experience minimal overhead in their
system as they process additional full knowledge bases pro-
duced by the grid system.

### 3.3 Querying with new records

Next, we evaluate the ability to match updates to exist-
ing entities or previously unseen records to the appropriate
entity cluster, as would be the case when new records are
discovered by the search engine, but have not yet been seen
by the Batchpath. As we have noted earlier, when exam-
ined by Fastpath alone, such records cannot cause existing
clusters to split or merge. We simulate this behavior by
first constructing a complete knowledge base as before. We
uniformly at random sample 1% of the input records into
a subset \( S_{1\%} \). For each record \( O_i \) in \( S_{1\%} \), we identify the
cluster in the knowledge base which contains \( O_i \) as a source
and remove all attribute values, relations, and blocking keys
provided to that entity by \( O_i \). We then create an inverted
index from the modified knowledge base and use the records
in \( S_{1\%} \) to query the Fastpath. For comparison, we have the
same number of entity clusters as the previous experiment,
but now the attribute values from those excluded records
are not present in any of the clusters and therefore cannot
influence candidate retrieval or pairwise matching.

We evaluate the precision, recall, and F-measure for the
top \( 1 \leq k \leq 10 \) entities. We consider the returned result
a positive match if the query record was matched with the
suggested cluster in the full knowledge base. False positives
and false negatives are defined as before, and true negatives
are now possible as some of the excluded records were the
only source record for their particular cluster.

Figures 3 and 4 shows the accuracy and running time
characteristics of introducing new records to the system via
the Fastpath. Figure 3 shows the precision decreases slightly
from 0.904 to 0.896 as we increase the number of candidate
entities we retrieve from 1 to 10, while at the same time
recall improves from 0.951 to 0.979. For \( k \geq 3 \), the F-
measure remains effectively constant. Average running time
for a query with \( k \leq 10 \) remains at approximately 20ms on
a single machine.

These results show that the Fastpath system enables a
natural tradeoff between precision and recall. For applica-
tions that prefer more aggressive de-duplication at the poten-
tial cost of incorrectly merging records, retrieving more
candidate entities is preferable. Other applications which
require more certainty in disambiguation decisions, possibly
at the expense of maintaining duplicate entities, can opt to
retrieve fewer candidate entities. Table 1 shows the break-
down of misclassification rates for \( k = 1, k = 5, \) and \( k = 10, \)
highlighting this tradeoff. In either scenario, the Fastpath achieves approximately 90% precision while also finding over 95% of the matching entities in the knowledge base.

<table>
<thead>
<tr>
<th>k</th>
<th>Incorrectly resolved</th>
<th>Incorrectly non-resolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.19%</td>
<td>0.57%</td>
</tr>
<tr>
<td>5</td>
<td>1.31%</td>
<td>0.31%</td>
</tr>
<tr>
<td>10</td>
<td>1.34%</td>
<td>0.25%</td>
</tr>
</tbody>
</table>

Table 1: New record misclassification rates

4. RELATED WORK

Developments in entity resolution have been recently surveyed in [7, 17, 5]. Since the early work on census [12] researchers have sought to reduce the number of entities that need comparison by improving blocking techniques. Improvements include the development of effective blocking functions [6], minimizing number of comparisons [3], scalable blocking approaches such as Canopy Clustering [11], MultiBlock, a way to avoid missing any comparisons [9], and frameworks for scaling blocking [14]. We have adopted Canopy Clustering and MultiBlock approaches in our batch system, and adapted the ideas of [1] for better quality.

Two papers propose algorithms that trade off execution time for quality of results to complete computations in a reasonable amount of time. [18] approaches the problem from a database view and introduces an anytime algorithm capable of clustering massive data sets with a large number of features. Similarly, [16] uses “hints” to better guide a pay-as-you-go clustering algorithm. Both approaches generate high quality results without performing a complete comparison.

Closer to our approach, [4] proposes the combination of blocking and information retrieval techniques to first select similar entities at low latency, followed by further analysis of those entities in ways that avoid having to exhaustively compare all their features. A similar approach is taken in [13] for discovering co-occurrences of entities (phrases) in a document collection. Word sets, called “entity chains”, in input documents are compared with existing clusters of entity chains using several similarity scores, and top-k clusters are returned as likely matches. New clusters are created if an existing cluster is not found. Our indexing system is similar to that of [4, 13], but we use blocking keys as terms in our search engine. Like [13], our system can also create new entities. However, to the best of our knowledge, our approach of combining the indexing and batch approach, with continuity between the two systems, appears to be unique.

5. CONCLUSION

In this paper we presented the unique requirements and design of an entity resolution system suitable for topical Web sites. Our system began as a purely batch processing entity resolution platform, Batchpath, focused on building a high quality knowledge base. Additional requirements of topical Web applications, such as latency, necessitated designing an alternative path which supports fast entity resolution while maintaining accuracy. The Fastpath system achieves these goals, operating as a low latency, incremental knowledge base, seeded by the high quality Batchpath output.

Our experiments on large scale, real-world data have demonstrated the Fastpath meets all the requirements of a topical Web site: it performs deduplication operations for entities in under 20ms while running on commodity hardware, it achieves nearly 90% precision and 98% recall when resolving newly discovered or updates to existing entities. The experiments have also shown that customers of the Fastpath can adjust the system’s bias between conservative (more duplicates, fewer incorrectly resolved entities) and aggressive (fewer duplicates, but possibly more incorrectly merged entities) resolution by simply tuning how many of the top-k candidate entities to consider.

Additional details about the system and experiments can be found in our full paper [15].

6. REFERENCES