Matching Titles with Cross Title Web-Search Enrichment and Community Detection

Nikhil Londhe, Vishrawas Gopalakrishnan
Aidong Zhang, Hung Q. Ngo, Rohini Srihari [*]
State University of New York at Buffalo
{nikhillo, vishrawa, azhang, hungngo, rohini}@buffalo.edu

ABSTRACT

Title matching refers roughly to the following problem. We are given two strings of text obtained from different data sources. The texts refer to some underlying physical entities and the problem is to report whether the two strings refer to the same physical entity or not. There are manifestations of this problem in a variety of domains, such as product or bibliography matching, and location or person disambiguation.

We propose a new approach to solving this problem, consisting of two main components. The first component uses Web searches to "enrich" the given pair of titles: making titles that refer to the same physical entity more similar, and those which do not, much less similar. A notion of similarity is then measured using the second component, where the tokens from the two titles are modelled as vertices of a "social" network graph. A "strength of ties" style of clustering algorithm is then applied on this to see whether they form one cohesive "community" (matching titles), or separately clustered communities (mismatching titles). Experimental results confirm the effectiveness of our approach over existing title matching methods across several input domains.

1. INTRODUCTION

Identifying duplicate patient records is an important and well-studied problem in the public health-care domain for the past 60 years [11, 13]. The essence of this problem has now blended itself in a variety of modern settings in the Internet-age, including aligning multilingual texts for machine translation, IP aliasing in networks [25], reconciling photographs with correct tags [21], or matching product titles across different online merchants [8, 16, 18]. The diversity in the target domains has led to a varied set of solutions tailored for individual domains.

[*]Arranged alphabetically by first name - students first

The two major goals of this paper are to (i) formulate a generic problem capturing the above "title/record matching" task, and (ii) devise a generic framework for solving this problem that can be used for a range of textual data drawn from several domains. For the sake of clarity, our solution discussion will first focus on one particular problem: matching titles of electronic products from heterogeneous online merchants. The requirements, data heterogeneity and richness from this domain are sufficient to illustrate the difficulty of the problem and the effectiveness of our approach. In a separate section, we then explain and experimentally confirm, how the same approach can be used to solve the title matching problem in other domains such as bibliography, locations and person names. There is another practical reason why product title matching is an important problem on its own - in UK alone, Web searches on this sector constitute a significant portion of Web traffic: as high as 6.06% (social networking - 7.3%) [10].

"Title matching" in product domain can be defined as the problem of consolidating item names that refer to the same underlying physical entity but are represented differently across sources. For example the product "*d200 10.2 megapixel digital slr camera body with lens kit - 18mm - 135mm (2.5"lcd - 7.5x optical zoom - 3872×2592 image)*" from `PriceGrabber.com` is represented simply as "*nikon d200*" at `CNET.com` [16]. This problem is very common, in part due to the general lack of standardisation in how a product is published, curated and managed [17].

At a concept level, instances to this problem can be categorised into the following, examples of which are listed in Table 1. Amongst them, some instances can be reasonably solved by existing methods while others need fresh ideas.

1. *Matching titles with high degree of token overlap*: The input titles represent the same physical entity, and their token sets have high overlap. This is arguably the easiest case, which can be solved using any reasonable similarity function such as the Jaccard coefficient.
2. *Matching titles with low degree of token overlap*: These include titles that are regarded as synonyms and need specific domain knowledge for disambiguation.
3. *"Softly-matched" titles*: The input titles represent variations of the same product (in physical characteristics or configuration). Distinguishing between an exactly-matched input and a softly-matched input using similarity measures can be very tricky.
4. *Unmatched titles with high degree of token overlap*: The titles do not represent the same physical entity, but their token sets have many elements in common.

Table 1: Examples of input instances to the Product Title Matching problem

Category	Title 1	Title 2	Jaccard Coefficient	TF-IDF + cosine	EN+IMP
1	sony dvd-r recordable camcorder media 3dmr30l1h	sony mini dvd-r media 3dmr30l1h	0.57	0.72	0.83
2	Rebel T3i	EOS 600D	0	0	0.33
3	speck seethru green hard shell case for 15' macbook - mb15grnseev2	speck products seethru case for apple 15' macbook pro - mb15-blu-see-v2	0.42	0.59	0.6
4	fe-140 digital camera battery charger replacement for 4 aa nimh 2800mah rechargeable batterie	olympus fe-140	0.07	0.47	0.25
5	canon 2gb sd secure digital card - 3505b001	sony alpha dslr-a350 digital slr camera - dslra350	0.083	0.1	0

This happens, for example, when one is a sub-category of the other (e.g., a product vs. its accessory). Titles under this category can be differentiated easily by humans but requires semantic understanding on the part of machines.

5. *Unmatched titles with low degree of token overlap*: This is symmetric to Category 1, and it is easily solved by a naïve similarity measure based approach.

Most of the existing methods can be broadly classified into two categories: based purely on a similarity measure [1, 7, 26, 8] or based on learning [20, 18, 24].

In a similarity-measure-based scheme, a score is assigned to each token in a title and the distance between two titles is calculated by a similarity function. After the distance function is normalised, a distance close to 0 indicates "mismatching" titles and a distance close to 1 leans toward the "matching" outcome. In a learning-based scheme, one uses training sets to learn to: (i) categorise the tokens into appropriate labels like brand, model number, etc. and (ii) identify the weight contribution of each label towards making the decision. In this work, we will be focusing on former and briefly describe towards the end how our method can be used to automatically generate a training set for the latter.

Expectedly, the performance of any similarity based technique depends heavily on how the tokens are weighed and the similarity measure used. It is easy to see that a basic similarity-measure scheme such as Jaccard coefficient would be effective with inputs from the "easy" categories - 1 and 5 (see examples in Table 1). However, these basic similarity functions cannot consistently perform well across the remaining three categories. Variations of the basic similarity approach, such as "Term Frequency $\times$ Inverse Document Frequency" (TF-IDF)[22] that has a global scope and stable ranking mechanism, can handle Category 3 fairly well, but fail in Category 4 for the reasons presented in [16].

There have been some recent works such as [8] and [16] (EN+IMP) that try to overcome the above limitations of these basic approaches. Although these fare better than most standard approaches - see Category 4, their efficacy is limited. This is largely due to their dependency on reducing the comparison to exact matches [8] or using a similarity measure that makes use of weights that are computed in a way which is agnostic to the compared title [16]. Furthermore, both these methods, as well as TF-IDF, fail in understanding the semantic relationships - see TF-IDF and EN+IMP for Category 2 in Table 1. These methods, especially TF-IDF, also fail in cases where there are no unique identifiers, refer to Section 5 for further details.

Our approach to overcome the aforementioned limitation of similarity-based approaches are two-fold.

Component 1. Using Web searches, we first transform a given pair of titles to a different pair of titles that magnifies the similarity or dissimilarity of the original pair. For example, this component helps transform a pair of titles belonging to Category 4 to a different pair with low overlap, effectively reducing the difficult Category 4 case into an easier Category 5 case. Symmetrically, we transform a Category 2 case into a new pair belonging to the easier Category 1 case.

We refer to this component as *Cross-Title Enrichment* (CTE). The motivation for this is drawn from "query rewriting", a classic IR technique. However, our approach and use-case is different from the traditional IR sense where query/user logs are used to perform this task [27, 6].

Component 2. To overcome the limitation of pure similarity -measure-based methods (even after title enrichment), we create a directed graph whose nodes are the private tokens of the two (enriched) titles. Each edge from token a to token b represents a degree of belief that a "can replace" b. A token is private if it belongs to one title but not the other; and a can replace b if in some sense a can be used to identify b. By modelling these private nodes as a "social" network and measuring the "community strength of ties" in a particular way, one can now quantify the similarity between the two sets of private tokens and thus, between the two given titles. Towards this end, we show experimentally the drawbacks of existing algorithms and develop a community detection algorithm (CDAM) that scores the cohesive strength of these nodes to quantify their similarity.

Due to varied domains of the datasets that we experiment upon and absence of a consolidated catalogue to query, we rely on Web search engine to query and the entire Web as the corpus. The novelty of our approach lies in breaking away from existing similarity based approaches and in utilising cues from search results to measure contextual similarity, whilst not losing our focus on the generality of our approach. Our framework not only yields better and more consistent F1-scores over existing methods across varied domains, but also has the power to identify phrases and logical segments.

Our contributions can be summarised as:

- We propose an end-to-end efficient system architecture for localised and cross context based disambiguation that is unsupervised and online.
- We develop a new community detection algorithm that serves as a similarity measure. The proposed measure is more suited to this problem setting and also has good accuracy as demonstrated in the experiments to disambiguate regular as well as soft-matches.
- Our proposed method has the ability to correlate the semantic similarity across the comparing entities, even under cases where there are no token overlaps.
- Our proposed method is not only suited to product domain but is also applicable to a wide range of domains involving textual data.
- On a diverse set of real-world datasets, we demonstrate

that the proposed method consistently performs better than many state-of-the-art techniques.

2. OVERVIEW OF OUR APPROACH

The key problem we address is the following. We are given two titles representing some physical entities such as commercial products, bib entries, locations, or people. Each title is a string of words. The two titles typically come from different sources, such as product titles from `Amazon` and `BestBuy`, or bib entries from `ACM` and `DBLP`, etc. Consider accessory vs. its corresponding product as an example:

t_1 = "Optio S60 camera Battery Charger Replacement"

t_2 = "Pentax Optio S60 Digital Camera"

We need to determine whether t_1 and t_2 represent the same physical entity or not. In some cases, t_1 and t_2 might be "almost" the same – this is the "softly matched" case discussed as Category 3 in Table 1. The output is a score, representing the confidence in marking the two titles "identical".

We begin by turning the titles into two sets of words.

$$T_1 = \{\text{optio, s60, camera, battery, charger, replacement}\}$$

$$T_2 = \{\text{pentax, optio, s60, digital, camera}\}.$$

Then the two components introduced in Section 1 act on these sets as follows:

- The first component called *Cross-Title Enrichment* (CTE) aims to use a small number of Web searches to enhance the quality of T_1 and T_2. In particular, using Web searches we "enrich" T_1 and T_2 by adding/deleting tokens to them in such a way that the enriched T_i still represents the same physical entity as the original T_i (for $i \in \{1, 2\}$), and yet the enriched versions make it "easier" to tell whether t_1 and t_2 refer to the same physical object. By rewriting the titles as stated above, we "hope" that the basic similarity measures that were ineffective earlier can be reused now.
- The second component called *Community Detection for Approximate Matching* (CDAM) is responsible for computing the distance between (enriched) T_1 and T_2 in cases where CTE could not resolve the ambiguity. This distance is computed based on the strength of the constructed "social" network between the private tokens, i.e., $(T_1 \setminus T_2) \cup (T_2 \setminus T_1)$. If these tokens form a network with separate "communities", then they are deemed far apart; otherwise if the strong ties give way to a single close-knit community, then the corresponding titles are deemed a match.

The motivation for CTE, lies in the following facts: (a) not all tokens have the same discriminatory power, (b) even powerful tokens like model number get their distinguishing power only in conjunction with other tokens like brand name, and (c) for matching titles, the set of shared tokens between the two titles are the ones that have most semantic correlation with a title's private tokens. So then, if we have a method that is able to "expand" the token-set shared by the titles and they are a mismatch, then the expansion will only increase the distinguishing power of the private tokens. On the other hand if the titles are a match, by this expansion one can "hope" to subsume all or part of the private tokens. So if C refers to shared context $(T_1 \cap T_2)$, then from previous example we get:

$$C = \{\text{optio, s60, camera}\}$$

$$T_1 \setminus C = T_1 \setminus T_2 = \{\text{battery, charger, replacement}\}$$

$$T_2 \setminus C = T_2 \setminus T_1 = \{\text{pentax, digital}\}$$

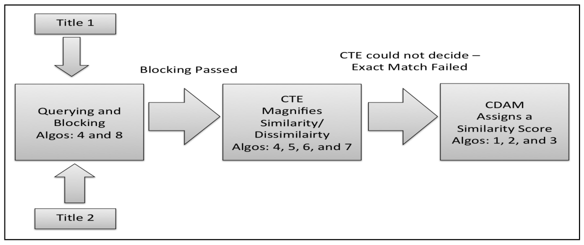

Title 1
Title 2
Querying and Blocking Algos: 4 and 8
Blocking Passed
CTE Magnifies Similarity/ Dissimilairty Algos: 4, 5, 6, and 7
CTE could not decide – Exact Match Failed
CDAM Assigns a Similarity Score Algos: 1, 2, and 3

Figure 1: Component Level Flow Diagram With Algorithms

Thus, the objective of CTE is to "expand" C, so that if t_1 and t_2 are really non-matching titles then discriminatory power of $T_1 \setminus C$ and $T_2 \setminus C$ increases, i.e., in this case, "battery", "charger" and "replacement". On the other hand, if the private tokens are not significant then their powers will fade away, since C would tend to subsume them as it should in the case of "pentax" and "digital".

Note that we make use of word "hope" when describing the expected behaviour of CTE. This is due to errors that are introduced by a Web search engine; thus, the need for CDAM. Apart from that, CDAM is also expected to handle the situations of semantic correlation as in Categories 2 and 3 of Table 1. Thus, if even after "expansion", $T_1 \setminus C$ and $T_2 \setminus C$ are not equal to $\emptyset$, then CDAM measures the semantic equivalence of the private tokens across the two titles.

One can conclude from the above discussion that the role of CTE is to try and make the pair of titles compact and precise, and check if they are an exact match. CDAM on the other hand quantifies the dissimilarity in cases where CTE is not able to make a decision. Thus, CDAM can handle all the cases which CTE can but vice-versa is not true, making CDAM the core module. CTE can be considered as performance booster as, apart from just adding contextually relevant tokens to aide CDAM, it also helps in reducing the Web query firing by removing tokens that are contextually redundant, thus pruning out private tokens for CDAM.

To get a holistic view of an end-to-end system with all the presented algorithms, we can visualise the matching task as a sequence of 3 stages - refer Figure 1:

(1) Querying and Blocking Stage - Responsible for issuing one Web query for each title and storing the obtained result set. This is then used for blocking and also CTE. Blocking helps in the first level of filtering where out-right mismatches are removed from comparison.

(2) CTE - Responsible for re-ranking the result set tokens (obtained in previous step) based upon the compared title. It identifies new relevant tokens and removes unimportant tokens. For example, in comparison between *EOS 600D* vs. *Rebel T3i*, *canon* & *camera* get added to both the titles due to their high frequencies in the search results. However, *eos* which has a lower frequency is still added to T_2 (by rescaling) because of its presence in the enriched set of T_1.

(3) CDAM - Responsible for disambiguating two enriched titles that are not exactly similar after CTE, for example quantifying the dissimilarity between {600D} and {Rebel, T3i,} under the shared context of {Canon, EOS, Camera}.

3. ALGORITHMS

Here we introduce the algorithms in a bottom-up approach to create a logical thought progression. For this, we first introduce CDAM because although ideally the input to CDAM is a pair of enriched titles this is not mandatory - CDAM measures the semantic relatedness of the private tokens in a given pair. We follow the discussion on CDAM by algorithms that assist in expanding C and improve the performance of CDAM - CTE. Each of these can be considered as an individual module that is simply plugged into the overall system - see Figure 1.

3.1 Community Detection for Approximate Matching - CDAM

Going by the notation introduced, the task of this section is to measure the semantic relatedness between the private tokens of T_1 and T_2 and quantify if the difference is substantial enough to be labelled a mismatch. As discussed earlier, we model this as a graph of private tokens $((T_1 \setminus C) \cup (T_2 \setminus C))$ where the edges represent the relationship of "can replace".

Furthermore, because each token in $T_1 \setminus C$ and $T_2 \setminus C$ derives its power only in conjunction with tokens in C, we pair each of them with C and issue a Web query. The result of the Web query is used to create an adjacency matrix $\mathbf{A}$ for a directed graph G, where the weight of an edge from node a to b represents the frequency of b in the result set, when a is a part of the query - this depicts the implied association between the tokens. Typically a Web search API returns an XML with three fields for each result: title, abstract and the URL of the Web-page. Based upon the level of granularity, we can choose any of them for selecting the tokens - the choice dictates the level of complexity. In our experiments, we restricted ourselves to only the title and used a simple frequency based method to identify the tokens.

Let us examine the example introduced earlier in this context. We generate the query set, Q, by pairing each element in $T_1 \setminus C$ and $T_2 \setminus C$ with C. Thus, :
$Q =${optio s60 camera battery, optio s60 camera charger, optio s60 camera replacement, optio s60 camera pentax,optio s60 camera digital}

If we restrict the number of Web results to 10, the co-occurrence adjacency matrix $\mathbf{A}$ obtained by firing each element in Q is:

$$\begin{bmatrix} \text{Tokens} & \text{battery} & \text{replacement} & \text{charger} & \text{pentax} & \text{digital} \\ \text{battery} & 0 & 3 & 5 & 0 & 0 \\ \text{replacement} & 3 & 0 & 3 & 0 & 0 \\ \text{charger} & 8 & 3 & 0 & 0 & 0 \\ \text{pentax} & 0 & 0 & 0 & 0 & 4 \\ \text{digital} & 0 & 0 & 0 & 8 & 0 \end{bmatrix}$$

We do additional processing on the adjacency matrix $\mathbf{A}$ to obtain a collapsed graph G'. For this, we define two terms here, namely *phrase* and *dominant token* as follows.

DEFINITION 1. *Any two tokens, u and v, are said to form a phrase under a given context C, if $A[u, v] \approx A[v, u]$, where $u, v \in T_1 \setminus C$ or $u, v \in T_2 \setminus C$.*

DEFINITION 2. *Token u is said to* dominate *v under a given context C if $A[u, v] \gg A[v, u]$, where $u, v \in T_1 \setminus C$ or $u, v \in T_2 \setminus C$.*

In our setting, two values are approximately equal if the difference $\leq$ (0.3 $\times$ the total number of results) and a token subsumes the other if the ratio of two values is >1.5.

Definition 1 consolidates private tokens of a given title that have a semantic interdependency given the shared set of words between the titles. Notice the importance of private tokens being part of the same title; this definition cannot be extended to private tokens across titles, as, such a relationship amounts to "equivalence" or "can replace", which we are trying to quantify in the first place.

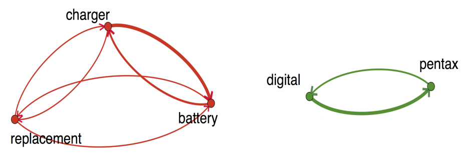
Algorithm 1 GRAPH COLLAPSING

```
 1: **Input:** $T_1$, $T_2$, $C$ adjacency matrix $\mathbf{A} = (a_{uv})$
 2: **Output:** Collapsed Graph - $G'$
 3: **repeat**
 4:     **if** $\exists u, v \in T_1 \setminus C$ **or** $\exists u, v \in T_2 \setminus C$, $|\ a_{uv} \approx a_{vu}$ **then**
 5:         Collapse $u$ and $v$ into one node
 6:         Update adjacency matrix $\mathbf{A}$
 7:     **end if**
 8: **until** No more reduction is possible
 9: **repeat**
10:     **if** $\exists u, v \in T_1 \setminus C$ **or** $\exists u, v \in T_2 \setminus C$, $|\ a_{uv} \gg a_{vu}$ **then**
11:         Update adjacency matrix $A$ to replace $v$ by $u$
12:     **end if**
13: **until** No more reduction is possible
14: Return $G'$ formed using $\mathbf{A}$
```

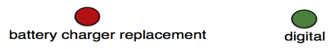
Definition 2 prunes the private token set space by identifying contextually dominating tokens within a set. Thus, if a covers b substantially in its result set but not vice-versa, it means b can be ignored under the context C in the presence of a.

Algorithm 1 enumerates the process of graph collapsing. It begins by enumerating all the possible edges based on the adjacency matrix. Continuing with the earlier example, the candidates for t_1 are (*battery*, *replacement*), (*charger*, *replacement*), (*battery*, *charger*). The candidates (*battery*, *replacement*) and (*charger*, *replacement*) are ranked higher than (*battery*, *charger*) as the weights of the former are much closer than the latter. Hence based on Definition 1, battery and replacement are first combined to give (*battery replacement*) as one token, which is subsequently combined with *charger* to yield (*battery replacement charger*) as one token. Thus, all the tokens in $T_1 \setminus C$ get collapsed to one token {*battery charger replacement*}.

The second half of Algorithm 1 deals with Definition 2 that prunes out redundant tokens within a title. From $\mathbf{A}$, it is clear that for t_2, *digital* dominates *pentax* under the context described by C. Thus, at the end of this step, we obtain a collapsed graph as shown in Figure 2b.



(a) Original Graph



(b) Collapsed Graph

Figure 2: Example of Graph Collapsing

Obtaining a collapsed graph allows us to narrow our focus onto important private tokens that are unique to a given title. Having identified these, the next step is to quantify the strength of correlation between the private tokens of this collapsed graph. To do so, we formulate the problem as one of graph clustering/community detection that measures strength of cohesion and finds the number of communities. If the strength of cohesion is high enough to result in a single community, then it implies that the reduced private tokens share high semantic dependency; if so we can term the two entities as a match, otherwise a mismatch.

In general the average out-degree and in-degree of the nodes in a collapsed graph are about 2 and 1 respectively. The average number of nodes in the graph is around 5, refer Section 5.4. Modularity detection algorithms on such small graphs can give misleading results [14]. Furthermore, edges between tokens across titles need to be interpreted differently. Assume that the two titles create two dense sub-graphs and an edge connects them. According to edge-betweenness and other community detection algorithms, there are at least two communities [15]. However, this interpretation fails if an edge connects two important tokens across the two titles, as depicted in Figure 3, implying there is a high likelihood that the two titles are equivalent and the graph has just one community.

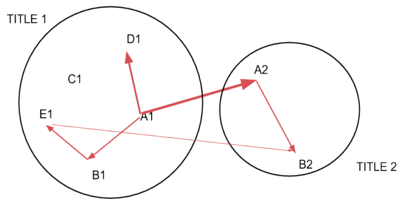


Figure 3: Example of a Graphical Relationship Between Tokens (represented as nodes) of Two Titles

Thus, the number of communities is a function of inter-title edges and their respective nodes. This is formulated as:

$$\max\left\{ (f(g(a), g(b), w(a,b)) - \lambda) \;\middle|\; \begin{smallmatrix} a \in T_1 \setminus T_2 \\ b \in T_2 \setminus T_1 \\ \exists E(a,b) \end{smallmatrix} \right\} \qquad (1)$$

where:

$g(.)$ = avg. weight of out-edges(.)−avg. weight of in-edges(.)

if in-degree(.) $= 0$, avg. weight of in-edges(.) $= -1$

$f(.)$ = any aggregating function like addition

λ = penalty value for isolated nodes (nodes with no edges)

$w(a,b)$ = edge weight between a and b

The penalty can be a constant value or as explained in Section 3.2 can be chosen adaptively. The logic behind Equation 1 is to weigh an edge by factoring in the importance of the participating vertices. The importance of a vertex is directly proportional to its weighted out-degree and inversely proportional to its weighted in-degree. This is consistent with our hypothesis that, nodes with high discrimination power tend to identify more nodes and are rarely identified by others, i.e. high out-degree and a low in-degree. As it has an upper bound of 1, a node with no incoming edges is assigned a value of 1. As the maximum value of g(.) can be 1, the maximum value of f(.) can be 3. If the result of Equation 1 > threshold τ, T_1 and T_2 are said to be a match - refer Algorithm 2.

Continuing our example: since after graph collapsing we get only two nodes - *battery charger replacement* and *digital* which are isolated, as per Equation 1 we obtain 2 communities inferring the products are different.

Algorithm 2 Community Detection Algorithm

```
1: Input: Collapsed Graph - $G'(V', E')$ , threshold $\tau$
2: Output: Number of Communities
3: $CE$ ←set of edges that connect inter title tokens.
4: for all $c \in CE$ do
5:     Let a be the start node and b the end node.
6:     if Indegree($a$) = 0 then
7:         InContribution($a$) = - 1
8:     else
9:         InContribution($a$) = Incoming edge weights of $a$
10:    end if
11:    OutContribution($a$) = Outgoing edge weights of $a$
12:    $g(a)$ = OutContribution($a$) - InContribution($a$)
13:    Repeat 6-12 for b and calculate $g(b)$
14:    $w_i \leftarrow f(g(a), g(b), edge\_weight(c))$
15: end for
16: $max\_val \leftarrow \forall_i max(w_i)$
17: for all $v \in V'$ do
18:    if $v$ is an isolated node then penalise $max\_val$
19:    end if
20: end for
21: if $max\_val > \tau$ then return 1 (to indicate match)
22: else return -1 (to indicate mismatch)
23: end if
```

Algorithm 3 Approximate Matching Algorithm

```
1: Input: Pair of product title in set form - $T_1$ and $T_2$,
   total number of results = $\kappa$
2: Output: Match or Mismatch
3: Compute $C$ as $T_1 \cap T_2$
4: Define the symmetric difference $UC := (T_1 \setminus C) \cup (T_2 \setminus C)$
5: Fire $C$ and remove tokens from $UC$ that are present in
   the top-$\kappa$ result set and add them to $C$.
6: Query set $Q := UC \times \{C\}$
7: $\forall x \in Q$ : fire $x$ to obtain top-$\kappa$ results. Create the
   adjacency matrix $A$ of tokens in $UC$, covered versus $x$
8: Collapse A using Algorithm 1
9: Run the community detection algorithm - Algorithm 2.
10: if (number of communities = 1) then return Match
11: else return Mismatch
12: end if
```

To illustrate a walk-through for Algorithm 2, consider the example of Category 2 of Table 1. Let t_1 be "rebel t3i" and t_2 be "eos 600d". In this example $C = \phi$ and $T_1 \setminus C = \{rebel, t3i\}$ and $T_2 \setminus C = \{eos, 600d\}$ and by proceeding exactly in the same way like the earlier example, we get **A** as:

$$\begin{bmatrix} \text{Tokens} & \text{rebel} & \text{t3i} & \text{eos} & \text{600D} \\ \text{rebel} & 0 & 0 & 0 & 0 \\ \text{t3i} & 8 & 0 & 5 & 0 \\ \text{eos} & 1 & 0 & 0 & 0 \\ \text{600D} & 3 & 5 & 7 & 0 \end{bmatrix}$$

After "graph collapsing", the adjacency matrix **A** reduces to two nodes - *t3i* and 600*d*. The inter-title edge along with the weight (for number of Web results = 10) is: (600*d*, *t3i*)=0.5. The result based on Equation 1 is tabulated in Table 2. Please note that since *t3i* and 600*d* do not have any inlinks from their own title tokens, they get the in-degree score of -1 to signify high importance. Algorithm 3 enumerates all the steps in this subsection.

Table 2: Walk-through Example of Community Detection Approach

Edge	In-degree Score of Node 1	Out-degree Score of Node 1	Total for Node 1	In-degree Score of Node 2	Out-degree Score of Node 2	Total for Node 2	Edge Weight	Total Score
(600d, t3i)	-1.0	0.35	1.0	-1.0	0.85	1.0	0.5	2.5

Algorithm 4 Generate Result Set

```
Input: Title $t_1$, number of results - $k$
Output: Result Set $R(t_1)$
$R(t_1) \leftarrow$ top-$k$ Web search result titles for $t_1$
```

Algorithm 5 Cross Title Enrichment

```
1: Input: Result set for title $t_1$ and $t_2$ - $R(t_1)$ and $R(t_2)$, support threshold $\eta$, support threshold for self determination $\mu$
2: Output: Enriched title $E(t_1)$ and $E(t_2)$
3: For each token $w_i$ in the $i^{th}$ ranked result in $R(t_1)$, let $freq(w_{t_{1_i}})$ be the weighted frequency of the token, calculated using DCG
4: Similarly compute $freq(w_{t_{2_i}})$ for all $w_i$ in $R(t_2)$
5: Let $w_{t_{1_1}}, w_{t_{1_2}}, \ldots, w_{t_{1_q}}$ be tokens of set $\{freq(w_{t_1})\}$ arranged in decreasing order of weighted frequency;
6: $\Omega_{t_1} \coloneqq w_{t_{1_i}}$ if $freq(w_{t_{1_i}}) > \eta \times freq(w_{t_1})$
7: Perform similar operation on for $\Omega_{t_2}$.
8: Let $avg_s$ & $avg_t$ be average of frequencies in $\Omega_{t_1}$ & $\Omega_{t_2}$
9: For each tok $\in \Omega_{t_1} \wedge T_1$ if $freq(tok) < \mu \times avg_s$: delete
10: Perform similar operation on for $\Omega_{t_2}$.
11: Using Algorithm 6 calculate gain and update $\Omega_{t_1}$ & $\Omega_{t_2}$
12: return $E(t_1) \coloneqq \Omega_{t_1}$ and $E(t_2) \coloneqq \Omega_{t_2}$
```

3.2 Cross Title Enrichment - CTE

While the previous section was concerned with only measuring the semantic relatedness, this section will instead focus on improving the shared context between the titles so as to magnify the semantic relatedness. This would not only improve the performance of CDAM but at the same time improve its efficiency by identifying and removing unimportant tokens. This task involves observing the titles under comparison and then adding or removing tokens that would amplify the similarity (or dissimilarity). This is precisely how our method differs from [16]. Algorithm 4 lists the initial steps for the above task.

Algorithm 5 describes the CTE process. The input to the algorithm is the result set of the two comparable titles t_1 and t_2 obtained through Algorithm 4.

A Web search engine returns a set of results sorted by relevance. This notion of relevancy needs to be propagated into the token selection mechanism. Since the results are ranked from highest to lowest, we apply a monotonically decreasing function - Discounted Cumulative Gain (DCG). The discount in DCG is given by the formula $1/\log_2(i)$ which we modify to $\lceil n \times \log_2(b+1) \rceil$ - where n is the total number of results returned, b is the bucket in which the $result_i$ falls in and i=1...n. The total number of buckets is calculated by $\lceil \sqrt(n) \rceil$ and the results are equally distributed into buckets. The need for bucketing is to provide a stratified decreasing function and avoid heavy penalisation of adjacent titles. The weighted frequency of tokens is then calculated using this modified discount.

Algorithm 6 calculate Gain

```
1: Input: Weighted frequency of tokens from $t_1$ and $t_2$ i.e. $w_{t_{1_i}}$ & $w_{t_{2_i}}$, and $\Omega_{t_1}$ & $\Omega_{t_2}$
2: Output: Gained weighted frequency for $w_{t_{1_i}}$ & $w_{t_{2_i}}$
3: repeat
4:     for each $tok \in \{(\Omega_{t_2} \setminus \Omega_{t_1}) \wedge w_{t_{1_i}}\}$ do
5:         calculate contribution of $tok$ towards the dissimilarity between $\Omega_{t_1}$ & $\Omega_{t_2}$
6:         increase weight of $tok$ in $w_{t_{1_i}}$ proportionately
7:     end for
8: until All tokens are processed
9: Repeat for all tokens $\in \{(\Omega_{t_1} \setminus \Omega_{t_2}) \wedge w_{t_{2_i}}\}$
10: return $w_{t_{1_i}}$ and $w_{t_{2_i}}$
```

We use two support thresholds in pruning out the candidate set of tokens. These are dependent on the weighted frequencies rather than on the number of results. The first support, typically around $0.5 \times \eta$ - where η is the highest weighted frequency, is more liberal in nature and is primarily responsible for adding new tokens. The second threshold, typically around $0.7 \times \mu$ - where μ is the average of weighted frequencies obtained from the previous step and is responsible for pruning tokens. Since search engines tend to return results that cover as many query tokens as possible, the secondary threshold acts as a regulariser and prevents uncontrolled preference to tokens from higher ranked results.

As stated before, the most distinguishing feature of our process is the cross title context awareness. For example, say for title 1 the token set obtained from previous step was {*canon*, *eos*, 600*d*, *camera*} and for title 2 {*canon*, *rebel*, *t3i*, *camera*}. Furthermore, the token *eos* was found during enriching title 2, but was pruned out due to low support. We thus, want to boost frequencies for all such tokens that: (1) did not make it to the final set, (2) were close to their support cut-off and (3) are present in the final set of the compared title. The boost percentage added depends on the contribution of that token to the dissimilarity between these intermediate enriched titles. Thus, we boost the frequency for *eos* in proportion to its contribution in the dissimilarity. If the modified frequency now meets the support threshold, this token is added to the set - refer Algorithm 6.

The co-occurrence matrix prepared based on the result set can be used to measure the token's value in a given title. This value can be used in Equation 1 as the penalty value for an isolated node. Based on the enrichment result thus obtained, one of the following situation can occur:
(1) The two sets are identical - the titles are an exact match.
(2) One set is a subset of other (subsumption). This generally occurs when a product is compared with its accessory.
(3) The case which is neither (1) nor (2) - handled by the algorithms in Section 3.1.

3.2.1 *Handling Subsumption Case*

Prima facie, comparisons under (2) should be labelled as

Algorithm 7 HANDLING SUBSUMPTION CASE

```
1: Input: $E(t_1)$ (shorter title), $E(t_2)$ & support $\eta$
2: Output: Match or Mismatch
3: Let private token set UC := $E(t_2) \setminus E(t_1)$
4: Fire $E(t_1)$ and get the top-$k$ Web-pages.
5: For each $tok \in UC$: $w(tok) \leftarrow$(# of document $tok$ co-occurred with $E(t_1)$)
6: if $\forall i \in UC : w(UC_i) \geq \eta \times k$ then return Match
7: else return Mismatch
8: end if
```

Algorithm 8 BLOCKING ALGORITHM

```
1: Input: Titles $t_1$, $t_2$; Result Sets $R(t_1)$, $R(t_2)$; threshold $\gamma$
2: Output: Pair $t_1$ and $t_2$, or null
3: Represent $R(t_1)$, $R(t_2)$ as vectors $V_{t_1}$, $V_{t_2}$ respectively
4: if cosine similarity between $V_{t_1}$ and $V_{t_2} \geq \gamma$ then
5:     return the pair $t_1$ and $t_2$ as candidate match
6: else return null
7: end if
```

a mismatch. However, if we delve further, we can see that this need not always be true due to various characteristics of a search engine. Consider the queries "Dell vostro 1400" and "Dell vostro 1400 laptop" in Google. The results we get are quite different from each other leading to a scenario where one enriched title is a subset of other. Handling this scenario requires us to go into page level processing of the results obtained from issuing the smaller title as the query and checking the coverage of the private tokens - Algorithm 7.

3.3 Blocking

The objective of this component is to reduce the total number of comparisons than that generated by a cartesian product of titles from the two sources. In other words if there are m titles from Source-1 and n titles from Source-2 and k matching titles across the two sources, then the objective of the blocking is to generate p titles such that $p \ll m \times n$, and $k \cap p$ should be high. Another important criterion is that the technique should be computationally light, preferably linear. With these considerations in mind, we propose a simple blocking technique that piggybacks on the existing framework to significantly reduce the number of comparisons, while maintaining a high recall rate. The idea here is to re-use the result set obtained from the enrichment phase. To recap, the enrichment phase involves firing one query per title. Our blocking system is embedded within this module - between the query firing and cross title enrichment phases. For each title, based on a very liberal threshold α, we generate a token set with frequencies that serves as a vector representation for that product. We then use the vector representations of both the titles under comparison to calculate their cosine similarity which, if greater than a threshold γ are considered for further processing; else rejected as a mismatch. Refer Algorithm 8 for details.

4. EXTENSIBILITY TO OTHER DOMAINS

By formulating the problem as one quantifying the dissimilarity allows, we can extend our method to broader entity resolution problems. We demonstrate this using a variety of examples drawn from diverse domains. Some of these are also used by Bing to promote their "Synonyms API". We finally conclude this section by showing how the proposed algorithms can be used to perform limited segmentation on titles, which can be used for machine learning purposes.

4.1 Generic Entity Resolution

To illustrate our framework's capability to even disambiguate entities other than product titles, consider the example of a location represented in different ways - "Seattle Tacoma International Airport" versus "Sea Tac". Let t_1 be "*Sea Tac*" and t_2 be "*Seattle Tacoma International Airport*". After application of Algorithm 5 to improve the context C, we get $E(t_1)$ as $\{airport, seatac, seattle\}$ and $E(t_2)$ as $\{seattle, international, airport, seattletacoma\}$. After applying Algorithm 1, *seattletacoma* and *international* form a phrase and the adjacency matrix is:

$$\begin{bmatrix} \text{Tokens} & \text{seattletacoma international} & \text{seatac} \\ \text{seattletacoma international} & 0 & 1 \\ \text{seatac} & 6 & 0 \end{bmatrix}$$

By the walk-through of Algorithm 2, we get the highest score of 2.6 for the inter-title edge $\{seatac$ $seattle\text{-}tacoma$ $international\}$, thereby implying them to be synonymous.

As an example from bibliography domain and also to show the importance of Algorithm 6 in CTE consider the following example. Let t_1 be a text of authors and their paper (in bold) - *Jerry Baulier, Philip Bohannon, S. Gogate, S. Joshi, C. Gupta, A. Khivesera, Henry F. Korth, Peter McIlroy, J. Miller, P. P. S. Narayan, M. Nemeth, Rajeev Rastogi, Abraham Silberschatz, S. Sudarshan* ***DataBlitz: A High Performance Main-Memory Storage Manager*** and t_2 be the list of authors - *P. P. S. Narayan, S. Joshi, M. Nemeth, Abraham Silberschatz, Henry F. Korth, A. Khivesera, Jerry Baulier, S. Sudarshan, Philip Bohannon, Peter McIlroy, J. Miller, S. Gogate, Rajeev Rastogi, C. Gupta.* The output of enrichment without Algorithm 6 gives an absurd enrichment for t_2 - $\{dblp\}$, whereas for t_1 we get a more meaningful token-set, $\{Jerry, Baulier, dblp\}$, as enrichment (considering the query was fired on a generic search engine vertical and not like Google Scholar). With cross title approach the situation improves considerably and we get the right output at the enrichment phase itself; we get $E(t_2)$ as $\{Jerry, Baulier, dblp\}$, which is equal to $E(t_1)$. The above result is also an example of a "soft match" - the original text is reduced to something more generic which could have more than one possible match.

Finally consider "Jennifer Lopez"(t_1) versus "Jlo"(t_2). This is an example that is handled by Algorithm 7 as after the cross title enrichment, we get $E(t_1)$ as $\{jennifer, lopez\}$ and $E(t_2)$ as $\{jlo, jennifer, lopez\}$. After issuing the shared token as query ("jennifer lopez"), the Algorithm 7 is able to identify the private token "jlo" significantly and hence, the pair is labelled a match.

4.2 Segmentation

The graphical formulation of the problem allows us to use Definition 1 to perform "limited" segmentation. Consider the example of *Jennifer Lopez*; the fact that it is a single phrase is domain knowledge. However, with the help of adjacency matrix, we can deduce that the co-occurrence of "Jennifer" and "Lopez" is very high and hence, it's very likely a phrase. For the product domain, consider the example "linksys dual-band wireless-n gaming adapter - wga600n".

Based upon the adjacency matrix, the system outputs the following impressive set of phrases $\{wga600n$ *linksys*, *dual-band wireless-n*, *gaming adapter*$\}$.

5. EXPERIMENTAL RESULTS

Having described the relevant algorithms and modules, we now present parameter sensitivity analysis, a detailed evaluation and reasoning of results of our algorithms.

Datasets: In our experiments, we have made use of 5 real world datasets. The first four datasets are publicly available benchmark datasets [19]. The fifth dataset is created by us, comprising pairs from heterogeneous domains. The matches described in the first four datasets are "exact matches" and fall predominantly under Category 1 and 5 of Table 1 with some them being in Category 4, whereas those in fifth belong to Category 2, 3 and 4.

(1) **Abt - Buy Dataset** contains 1097 matching product titles. This dataset comprises of electronic products and usually has unique identifiers in each title.
(2) **Amazon - Google Dataset** has 1300 matching product titles and unlike the Abt - Buy dataset, the titles do not always have discriminatory tokens.
(3) **ACM - DBLP Dataset** is made up of 2224 matching article titles. This dataset is subjected to extensive manual curing at source level.
(4) **DBLP - Scholar Dataset** has around 5347 matching article titles. Compared to ACM - DBLP dataset, this is fairly more difficult to process.
(5) **Heterogeneous Dataset** has 500 pairs, equally divided into matches and mismatches. It is a result of aggregation from domains like bibliographies, celebrities and their short-forms, locations and airport codes, products, etc.

Evaluation Metrics: The first 4 datasets being benchmark datasets have been extensively studied in [19, 16], and hence we use the same evaluation measure, F1-score, to compare the efficacies in cases of "exact matches". Through "Heterogeneous" dataset we demonstrate the superiority of our approach in handling soft-matches and synonyms. Although structured data might be available for the domains we experimented with, we did not consider them during experiments as we wanted to show the general applicability of our solution (when structured data is not available).

Using these datasets, we show the versatility of our system to adjust the degree of required matching - from soft and synonymous matches to exact-matches, simply by varying the threshold of the community detection module.

Though we have been giving examples from the product domain so far, through datasets (3), (4) and (5) we would like to show that our method is scalable to other domains. Note, that the queries are fired on a generic search engine vertical that is not tuned towards a particular domain.

Schemes Compared: We evaluate the system performance under the following three schemes:

1. **Cross-title Enrichment:** - Employs all the components of the system.
2. **Simple Enrichment:** - Exactly like above except for Algorithm 6.
3. **No Enrichment:** - This corresponds to just the graph-based approach with no enrichment applied.

Once we have identified the right parameter settings and scheme, we then evaluate the efficacy of our community detection algorithm. The algorithms compared are:

1. **Edge-Betweeness:** - This algorithm was proposed by Newman and Girvan in [15]. It is a popular community detection algorithm which identifies the communities based on inter-community edges. With this base-line we would demonstrate that the classical approach of community detection is unsuitable for our exercise.
2. **Modularity:** - With this we demonstrate that for small communities, as in our case, modularity detection algorithm does not work well. Here, we use Gephi's implementation of [4].

For Web queries we use the Yahoo! Boss Search API. In order to show the performance of our blocking mechanism and complexity, we have included a section that compares its recall to false positives for various datasets.

Our experimental results show that **CTE+CDAM**:

1. Outperforms several state of the art techniques including many of the supervised approaches.
2. Has the dexterity to adapt to the required level of matching - exact vs. soft and also perform well in disambiguating "synonymous" titles that have very little token overlap.
3. Can be generalised to other domains.

We begin the analysis with experiments on the product domain and discuss the evaluations and parameter settings on Abt-Buy dataset. Once the right combination is identified, we evaluate the performance on other datasets. To be fair, we reused the same settings for bibliography domain. We conclude the section with a brief discussion on blocking and running time.

5.1 Performance Measure on Product Titles

This section is organised based on the datasets. We use Abt-Buy dataset to establish the right scheme and parameters, and then show the response curve under that setting on both Amazon-Google and "Heterogeneous" dataset.

5.1.1 Abt-Buy Dataset

Figure 4 shows the response of F1-scores to different parameter settings for Abt-Buy dataset, where η and μ refers to support thresholds referred in Algorithm 5. As can be seen, better scores are obtained when self-support threshold μ is higher than base support η. This is due to dropping of spurious or redundant tokens from query, thereby increasing accuracy. The remaining variable is the community detection threshold τ, with a value ≥ 3 indicating the module is "switched-off". We see the output stabilising at $\tau = 2.8$, as lower values lead to aggressive approximate matching and thus, False Positives (FPs). We later show that this threshold is dependent on the nature of the dataset and should not be viewed as a failure of the algorithm but rather as resilience to meet the required level of matching.

Efficiency vs. Effectiveness Analysis: - Here we analyse the performance of our system by varying the different support parameters. As observed from Figure 4, we set $\eta = 0.3$ and $\mu = 0.5$. The first parameter to be considered is the number of results - k, fetched from the search engine for each query during enrichment phase. We observe, in Figure 5, that the F1-score improves with increasing k until $k = 20$. Thereafter, the scores continue to dip due to the nature and quality response of the underlying search engine. Algorithm 3 also exhibits identical behaviour (Figure 6) but is more aggravated for lower thresholds because

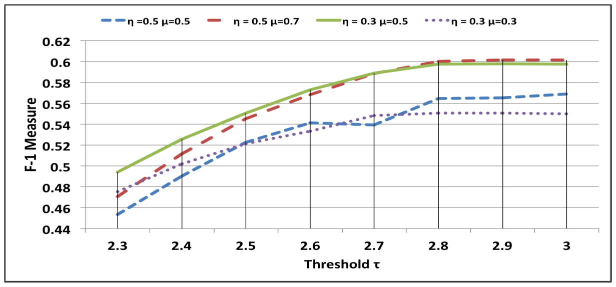


Figure 4: F1-Score for Abt-Buy Dataset

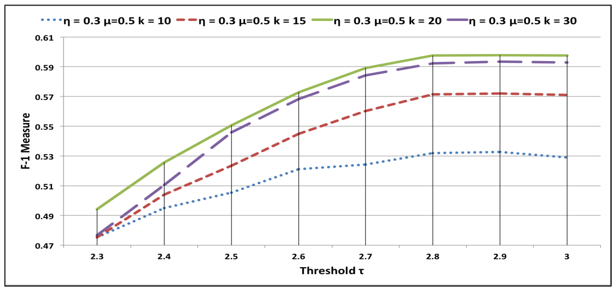


Figure 5: Response of F1-score to different k values

of shorter queries; thus for a better performance including lower processing cost we set $\kappa = 10$.

Importance of the Enrichment Module: - Here we experimentally prove the importance of the cross title enrichment module, namely the effect on system performance using simple enrichment and no enrichment. Figure 7 shows the importance of Algorithm 6, the heart of cross title enrichment. One can see, plots with gain have a higher F1-measure compared to their counterparts. Figure 8 shows the F1-score without enrichment for the same parameter setting of $\kappa = 10$. The figure shows the F1-score under both assumptions: "soft matches" as FPs and "removing soft matches from consideration".

5.1.2 Amazon-Google Dataset

We start with the parameters as fixed in the previous run i.e., $\eta = 0.3$, $\mu = 0.5$, $k = 20$, $\kappa = 10$, and present the result on Amazon-Google dataset. Figure 9 shows the response curve of our system under different scheme. As before, the scheme with cross title enrichment performs the best.

Table 3 compares of our method with many benchmark techniques published in [19] and [16]. A detailed discussion on the experimental results is provided in Section 5.3.

5.1.3 Heterogeneous Dataset

Figure 10 shows the F1-score curve for the given parameter thresholds on our heterogeneous dataset. Compared to Figure 4 and Figure 9, we can see the best F1-score does not lie towards the end of the x-axis, implying there is a significant contribution of the community detection module towards the performance. This is expected, as the enrichment module by itself is incapable of handling the characteristics of the data - synonyms and abbreviations. In line with earlier observations, schemes with no enrichment and no cross title re-ranking give inferior results, thus reinforcing our methodology. Table 4 compares the proposed method with nearest competitors from Table 3.

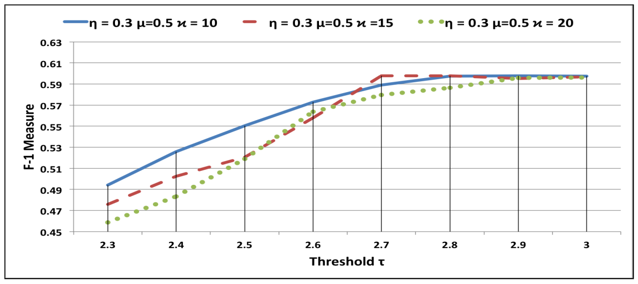


Figure 6: Response of F1-score to different κ values

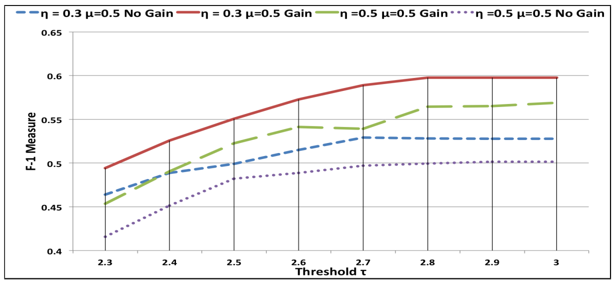


Figure 7: F1-Score Simple Enrichment vs. CTE

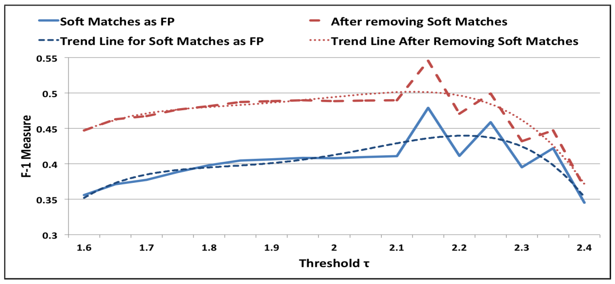


Figure 8: F1-score Without Enrichment Module

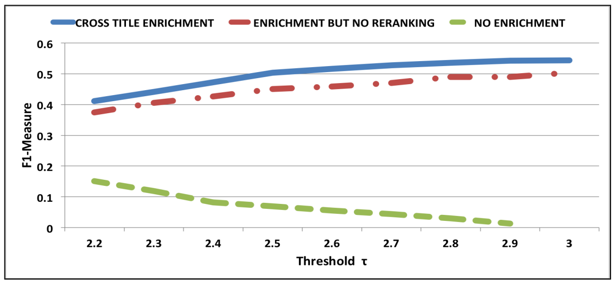


Figure 9: F1-Score for Amazon-Google Dataset

5.1.4 Comparison of Community Detection Algorithms

Here we experimentally show the superiority of our approach in detecting communities in a graph under this problem setting. One can treat the non-learning approach as elementary clustering baselines under the premise of pair-wise matching. We compare the F1-score obtained by employing [15] and [4] in our framework. Recall that a threshold of 3 in above ROCs indicate the "switching off" of the community detection module. Hence we will ignore that threshold and its neighbourhood in our comparison for Abt-Buy

Table 3: Comparison of F1-Score Across Benchmark Techniques - Product Domain

Dataset	Learning Techniques			Non-learning Techniques				
	FEBRL SVM comb.	MARLIN ADTree comb.	MARLIN SVM comb.	FEBRL FellegiSunter + Trigram	PPJoin+ + Jaccard	IDF based Cosine Similarity	EN+IMP	**CTE+CDAM**
Abt-Buy	44.5%	18.4%	54.8%	44.5%	47.4%	57%	62%	**59.8%**
Amazon-Google	46.5%	45.0%	50.5%	48.4%	41.9%	52.9%	36.4%	**54.4%**

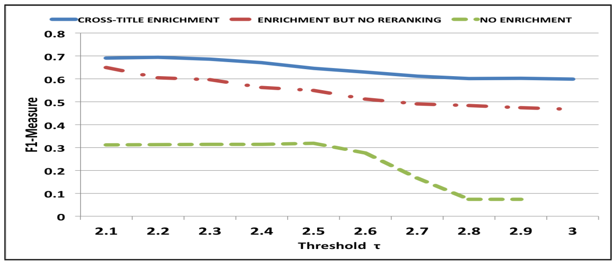


Figure 10: ROC curve for Heterogeneous Dataset

Table 4: Comparison for Heterogeneous Dataset

Methods	IDF	EN+IMP	**CTE+CDAM**
F1-score	50.4%	53.5%	**69.43%**

and Amazon-Google Datasets. In fact for these, we will report the worst results from our framework to show that it still outperforms the compared techniques. For the heterogeneous dataset, we report the best numbers as the corresponding threshold is not close to the "switch-off" value - refer Table 5. The failure of Edge-Betweeness is due to semantics associated to inter-title edges (refer Figure 3) and is explained in Section 3.1. Though the definition of modularity is very well suited to our problem statement, it has performance issues on smaller graph (shown here experimentally and proven theoretically in [14]).

Table 5: Comparison of Community Detection Algorithms

Dataset	Edge Betweenness	Blondel's Algorithm	**CDAM**
Abt-Buy	29.8%	37.8%	**49.41%**
Amazon-Google	26.8%	31.6%	**41.06%**
Heterogenous	58.48%	63.2%	**69.43%**

5.2 Performance on Bibliography Dataset

Here we briefly discuss on how well our method scales to other domains. As can be seen from Table 6, our method gives promising results. However, unlike the product domain, the best F1-score lies between the threshold ranges of 2.6 to 3. Because these datasets do not have unique identifying tokens like model number, we have removed IDF based cosine similarity from comparison. EN+IMP has a limited capability in assigning right scores and hence there is a high fluctuation in scores across the two datasets. It is to be noted that the search engine vertical is generic and despite this, we have achieved a good F1-Score. Thus it is our hypothesis, that with the right search vertical, the performance of the system can but only improve.

5.3 Discussion on Results

So far we have evaluated our system on 5 datasets, each of which presents unique characteristics. The results are summarised below:

(1) Cases like Abt-Buy dataset that have unique identifiers and no accessory titles are simpler. This can be seen from relatively high F1-Scores compared to Amazon-Google dataset. IDF and EN+IMP are conceptually at an advantage but our method is still able to match their performance.

(2) In cases where the semantics play an important role, like Amazon-Google, "Heterogeneous" and bibliography domain datasets, our method *consistently* ranks in top 2 and this despite making use of a generic search engine. In fact we rank the best in the more difficult of the two bibliography dataset (DBLP-Scholar) - Table 6.

(3) Compared to learning techniques we perform better as we are not restrained by its static features and also owing to semantic awareness that is lacking in its training part.

(4) As shown in Table 5, the proposed technique for community detection is better suited to our problem domain than some of the state of the art. The community detection module is *strictly useful* for detecting "soft-matches" and "synonyms". As can be seen in ROC curves for Abt-Buy and Amazon-Google datasets, the best score lie towards the "switch-off" value. This is because of the nature of the datasets and should not be misconstrued as ineffectiveness or redundancy of the community detection module. These datasets have exact matches and they are handled promisingly by cross-title enrichment module and do not require approximate matching facilitated by CDAM. Thus, based on the nature of dataset and the required level of matching (hard vs. soft) one can fine-tune the threshold value. The importance of community detection module is evident in the "heterogeneous dataset" and in bibliography domain dataset. It is also worthwhile to point out that, though with the community detection module switched-on does not yield the best result in Abt-Buy and Amazon-Google datasets, it still performs better than some of the techniques in Table 3.

(5) We are able to better handle the accessories scenario than our nearest competitors. Table 7 gives a comparison of False Positives (FPs) for about 100 product-accessory comparison. EN+IMP because of its within title importance ranking scheme tends to do a lot better than IDF based cosine which assigns a global score to the tokens and is not locally sensitive. Our method goes beyond EN+IMP as it performs cross title context aware analysis. CDAM by itself being stricter than CTE and CTE+CDAM has high precision and hence has fewer FPs but suffers from recall as seen in earlier graphs - no enrichment. Because CTE+CDAM is to be used only for soft-matches and synonyms where approximate matching is required, one can settle to simple CTE for accessory comparison.

(6) We will compare our run-time complexity with EN+IMP, our closest ideological competitor. EN+IMP requires at least $N \times (^{n}C_2 + 1)$ Web-firings, where n is the average number of tokens in a title and N the number of titles, while the CTE+CDAM requires $N \times (1 + k)$ Web-queries

Table 7: F1-Score comparison for accessories

	IDF+cosine	EN+IMP	CTE	CDAM	CTE+CDAM
% FPs	79.76%	49.4%	34.5%	0%	46.7%

Table 6: Comparison of F1-Score Across Benchmark Techniques - Bibliography Domain

Dataset	Learning Techniques			Non-learning Techniques			
	FEBRL SVM comb.	MARLIN ADTree comb.	MARLIN SVM comb.	FEBRL FellegiSunter + Trigram	PPJoin+ + Jaccard	EN+IMP	**CTE+CDAM**
DBLP-ACM	97.3%	96.4%	96.4%	97.6%	91.9%	68.62%	**96.99%**
DBLP-Scholar	81.9%	82.6%	82.6%	57.2%	77.8%	70.26%	**82.77%**

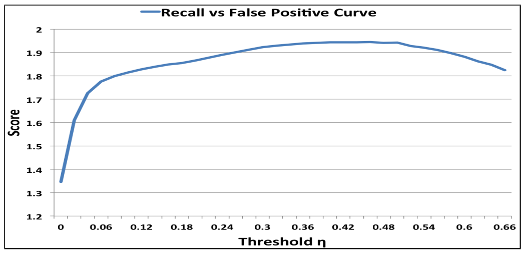


Figure 11: ROC curve for Blocking Mechanism

for yielding similar results in Abt-Buy and Amazon-Google dataset, since the best results are obtained with just cross title enrichment module itself. k above represents the k in Algorithm 7. As this operation can be regarded as accessing the page and not a query, we can ignore this constant.
(7) A draw-back of using Web-search engine is that say a manufacturer discontinues a product and he is regarded as an authority source. The search engine in its updates removes the corresponding pages from the top results severely affecting the FPs. For example, "Samsung 52' TV" and "Samsung 40' TV" from the Abt-Buy dataset (dated 2009) turn up as matches owing to product discontinuity.

5.4 Blocking and Run Time Performance

Here we briefly discuss the efficiency of our technique. We present the True Positive Rate vs. False Positive Rate for all the dataset and for brevity will just show the total number of firings for Abt-Buy dataset. Please note, unlike other methods, our run-time complexity is solely on the number of search query firings as the cost involved in processing the queries are external constants like bandwidth, server speed, etc. Thus, the primary objective of blocking is to reduce this - the enrichment section also partly assists in doing so. As the best results for "exact matches" is obtained by "switching-off" the community detection module, the theoretical bound as stated earlier is $N \times (1 + k)$, where N is the number of titles and k a constant representing page access.

Figure 11 shows the ROC curve for different thresholds for Abt-Buy Dataset. The score is similar in definition to F1-score and is computed based on the recall and FPs. One can see, the best point is at $\eta = 0.48$ which gives a recall of 1072/1097 matches in case of Abt-Buy dataset, and FPs of 11418. This is far less FPs compared to the complete cross of 1081×1092. In terms of queries, out of 12,490 pairs of comparisons 3,176 were decided by the enrichment phase itself, leaving 9,314 pairs of title for Algorithm 2. This filtering drastically reduced the number of search engine queries to 4.42 per pair, which is about the average number of private tokens in a pair. Table 8 presents the blocking result on various datasets and compares it with the values in [19].

6. RELATED WORKS

Entity resolution is a well-established problem that has been studied extensively in [12][5]. However, as enumerated in [19], product domain presents a unique challenge and much of the existing work under-performs in this regard. Furthermore, most widely used techniques are either supervised or require some user interaction [23]. A full-fledged supervised system cannot handle the dynamics of product representation as described in Table 1 and hence, the poor F1-scores as shown in Table 3.

To rectify this problem, work has been done that uses Web search engines to mimic domain knowledge without user intervention and provide reasonably good results [16], [8], [9]. While [8], [9] use a search engine or Web corpus to identify discriminative tokens, [16] uses them instead to add context and measure each token's importance. All the three techniques suffer from the problems described in this paper as they use similarity measures that do not capture the implicit relationship between tokens across titles. Pair-wise de-duplication is studied in [2] and there has been usage of this approach in literature. However, our method differs from these papers in terms of defining the relationship across pairs and uses Web-search instead to quantify the implicit relationships. Also, the enrichment done is local to the pair giving us a better and contextually more relevant enrichment set.

Another approach towards entity resolution in product domain is to identify the annotation labels like "brand", "model-number", etc. Once this tagging is done all that one has to do is to match titles on comparable annotations. Such annotations though widely available through resources like *Freebase* may not be applicable for all the different categories of products such as clothing, video-games, etc. Also, these annotations will not be completely useful in cases involving synonymous titles with no token overlap - Category 2 in Table 1 or in cases of soft-matches (variation in model number) as shown in Category 3 of Table 1. These cases will require training to understand and assign appropriate weights to the annotations labels for matching. For instance the colour attribute "green" in Category 3 is irrelevant while the accessory "battery charger replacement" is more important than model number "fe-140" in case of Category 4. Furthermore, availability of manually curated fields for other text domains like bibliography, celebrities, etc. complicates the portability of this approach to other domains.

[18] considers the problem of matching unstructured offers to structured product descriptions. A feature vector is constructed using similarity between attribute values that match the offer and the title. The similarity function is then learnt by training a classifier. Ours is unsupervised and does not require training data or specifications. [3] also formulates the problem as a graph and perform clustering. However, they tend to use the relations present in the reference document to describe the edge, for example whether the authors ever co-authored a paper and using "author-of" type relationships; resulting in scalability issues to other domains. We on other hand establish the implicit relationships and do not use any token related features.

Table 8: Comparison of Blocking Techniques

Sources	Source Size		Perfect Result	Mapping size (#correspondences)			Our Performance	
	Source 1	Source 2		Cartesian product	Trigram Blocking Result	Our Blocking Mechanism	Recall	False Positives
Abt-Buy	1,081	1,092	1,097	1.2 million	164,072	12,490	1,072	11,418
Amazon-Google	1,363	3,226	1,300	4.4 million	342,761	13,202	1,240	11,962
DBLP-ACM	2,616	2,294	2,224	6 million	494,000	8,185	2,143	6,042
DBLP-Scholar	2,616	64,263	5,347	168.1 million	607,000	13,834	4,604	9,230

7. CONCLUSIONS AND FUTURE WORK

In this work, we have proposed and validated an unsupervised and online framework that is semantically conscious and hence capable of disambiguating not only product but also generic text entities. We achieve a better consistent F1-scores compared to most state of the art techniques, including supervised algorithms. The state of being semantically conscious allows our framework to disambiguate entities where other techniques fail outright, synonyms and abbreviations. Furthermore, we demonstrate our method's ability to perform limited phrase detection and segmentation as an implicit and intrinsic part of the larger process.

In this aspect we find an interesting research problem: Can the adjacency matrix along with limited training samples be used to train a tagger that is able to tag brands, model numbers, etc. helping us in blocking phase? Will this allow us to use the labels in conjunction with our framework's output to train a supervised algorithm for matching, thus reducing the queries and time? Another aspect to investigate is making use of a custom query engine and removing Web-search aspect. In this regard, can a customised search engine on top of reference data that incorporates domain knowledge improve the disambiguation quality?

8. ACKNOWLEDGEMENT

A.Z's work is partly supported by NSF grant 1218393. H.Q.N's work is partly supported by NSF grant CCF-1319402.

9. REFERENCES

[1] R. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. pages 131–140, 2007.

[2] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *The VLDB Journal—The International Journal on Very Large Data Bases*, 18(1):255–276, 2009.

[3] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[5] D. G. Brizan and A. U. Tansel. A survey of entity resolution and record linkage methodologies. *Communications of the IIMA*, 6(3):41–50, 2006.

[6] A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online expansion of rare queries for sponsored search. In *Proceedings of the 18th international conference on World wide web*, pages 511–520, 2009.

[7] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, pages 5–5, 2006.

[8] S. Chaudhuri, V. Ganti, and D. Xin. Exploiting web search to generate synonyms for entities. In *Proceedings of the 18th international conference on World wide web*, pages 151–160, 2009.

[9] S. Chaudhuri, V. Ganti, and D. Xin. Mining document collections to facilitate accurate approximate entity matching. *Proceedings of the VLDB Endowment*, 2(1):395–406, 2009.

[10] Daniel Buchuk. Uk online porn ban: Web traffic analysis of britain's porn affair. *SimilarWeb Blog*, 2013.

[11] H. L. Dunn. Record linkage*. *American Journal of Public Health and the Nations Health*, 36(12):1412–1416, 1946.

[12] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16, 2007.

[13] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[14] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, pages 36–41, 2007.

[15] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, pages 7821–7826, 2002.

[16] V. Gopalakrishnan, S. P. Iyengar, A. Madaan, R. Rastogi, and S. Sengamedu. Matching product titles using web-based enrichment. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 605–614, 2012.

[17] M. Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *Knowledge Engineering: Practice and Patterns*, pages 329–346. 2008.

[18] A. Kannan, I. E. Givoni, R. Agrawal, and A. Fuxman. Matching unstructured product offers to structured product specifications. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 404–412, 2011.

[19] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, 3(1-2):484–493, 2010.

[20] H. Köpcke, A. Thor, S. Thomas, and E. Rahm. Tailoring entity resolution for matching product offers. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 545–550, 2012.

[21] Lise Getoor, Ashwin Machanavajjhala. Entity resolution for big data. *KDD Tutorial*, 2013.

[22] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[23] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *SIGKDD*, pages 269–278, 2002.

[24] H. Wang, X. Zhang, J. Li, and H. Gao. Productseeker: entity-based product retrieval for e-commerce. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 1085–1086. ACM, 2013.

[25] W. Willinger, D. Alderson, and J. C. Doyle. *Mathematics and the internet: A source of enormous confusion and great potential.* Defense Technical Information Center, 2009.

[26] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *(TODS)*, 36(3):15, 2011.

[27] W. V. Zhang and R. Jones. Comparing click logs and editorial labels for training query rewriting. In *WWW 2007 Workshop on Query Log Analysis: Social And Technological Challenges*, 2007.