Matching Product Titles using Web-based Enrichment

Vishrawas Gopalakrishnan*
Suny Buffalo
Buffalo, NY
vishrawa@buffalo.edu

Suresh Iyengar
Yahoo Labs
Bangalore, India
supartha@yahoo-inc.com

Amit Madaan*
TheFind, Inc
Mountain View, CA
amadaan@thefind.com

Rajeev Rastogi*
Amazon
Bangalore, India
rastogi@amazon.com

Srinivasan Sengamedu*
Komli Labs
Bangalore, India
shs@komli.com

ABSTRACT

Matching product titles from different data feeds that refer to the same underlying product entity is a key problem in online shopping. This matching problem is challenging because titles across the feeds have diverse representations with some missing important keywords like brand and others containing extraneous keywords related to product specifications. In this paper, we propose a novel *unsupervised* matching algorithm that leverages web search engines to (1) *enrich* product titles by adding important missing tokens that occur frequently in search results, and (2) compute *importance scores* for tokens based on their ability to retrieve other (enriched title) tokens in search results. Our matching scheme calculates the Cosine similarity between enriched title pairs with tokens weighted by their importance scores. We propose an optimization that exploits the templatized structure of product titles to reduce the number of search queries. In experiments with real-life shopping datasets, we found that our matching algorithm has superior F1 scores compared to IDF-based cosine similarity.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms

Keywords

Web-based enrichment, Entity resolution

*Work done while at Yahoo Labs, Bangalore

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

1. INTRODUCTION

Consumers are increasingly turning to online shopping sites (e.g., shopping.yahoo.com, shopping.bing.com) to research products prior to making buying decisions. These sites leverage data from multiple sources that include aggregator sites (e.g., PriceGrabber, CNET), merchant sites (e.g., buy.com, amazon.com) containing sales offers, reviews sites (e.g., epinions.com, newegg.com) containing reviews and ratings information, and auction sites (e.g., eBay, Amazon marketplace). The sites consolidate the specifications, prices, ratings, and reviews information for each product present in the various source data feeds, and present a unified view of each product to the user. Thus, the sites allow users to compare prices offered by different vendors for a product or browse through reviews for a product from different web sites.

A key challenge to providing a unified product view is matching records for the same product from different data feeds. This record matching problem has been extensively studied in the literature under different names like record linkage [13], duplicate detection [12, 20], entity resolution [2, 1], and merge/purge [14]. Much of this previous work has focused on matching records with multiple overlapping attributes. The general approach is to first compute similarity scores for each attribute using traditional similarity metrics like Jaccard similarity, Cosine similarity, edit distance, etc. [17, 12], and then combine these attribute-level similarity scores to derive record-level matching scores using unsupervised and supervised techniques. However, as discussed below, prior approaches may not work well in our product record matching scenario.

In our product setting, records in the different data feeds may have heterogeneous schemas and thus contain diverse sets of attribute values. For example, sales offer feeds from merchants may contain information about prices but not reviews, while a feed from a blogging or a product forum site may contain ratings and reviews information but not prices. Furthermore, even if these attributes are present, their values for the same product may vary widely across feeds making them unreliable to use for matching. This is because different vendors may price the same product very differently; similarly, product reviews across feeds may be written by diverse users with disparate viewpoints and writing styles. In the absence of universally agreed upon unique product identifiers between the various information providers, the only

attribute that uniquely identifies a product and that is consistently present in all the feeds is *product title*. So in this paper, we rely on product titles for matching records in the different data feeds.

A product title is a short unstructured textual description that uniquely identifies a product. Previous work has used similarity metrics like Jaccard similarity, Cosine similarity, edit distance, etc. for determining the similarity between string-valued attributes like person names or addresses. However, different feeds may use diverse product title representations for the same product. Consequently, traditional string similarity metrics may not work well for matching product titles.

Consider the 3 product title pairs from product aggregators PriceGrabber and CNET in rows (a)-(c) of Table 1. The pair of titles in row (a) are different representations of the same camera product with model number "d200". The brand name "nikon" is missing in Title 1 while Title 2 does not contain relevant descriptive keywords like "digital slr camera" and other product specifications. Even though the two titles correspond to the same product, the Jaccard similarity between their token sets (using white space as delimiter) is only $\frac{1}{22} = 0.045$. In contrast, the pair of titles in row (b) correspond to different camera models but have a much higher Jaccard similarity of 0.125. Now, based on the title pairs in rows (a) and (b), it may be tempting to declare two titles as matching if they have identical model numbers. However, besides the obvious difficulty of identifying a wide range of model number formats within titles, this strategy will not work in many cases. For instance, the two titles in row (c) have the same model number but they represent different products. Specifically, Title 2 corresponds to a camera while Title 1 refers to its accessory – a camera battery charger.

Clearly, traditional similarity metrics like Jaccard similarity fare very poorly at the task of detecting matching product titles – the first matching title pair in Table 1 has a much lower similarity score compared to the other two non-matching pairs. The problem with simple Jaccard similarity is that it treats all tokens equally. However, as we saw earlier, tokens like model number are more important and thus should be assigned a higher weight. A popular weight assignment method in the IR literature [17] is *Inverse Document Frequency* (IDF) – it assigns each token w a weight of $I_w = \log \frac{N}{N_w}$ where N is the the total number of records in the feeds and N_w is the number of records that contain token w. The Cosine similarity metric (with IDF token weights) between a pair t, t' of titles is then given by $\frac{\sum_{w \in (t \cap t')} I_w^2}{\sqrt{\sum_{w \in t} I_w^2} \sqrt{\sum_{w \in t'} I_w^2}}$. We show the Cosine similarity scores for the title pairs in the final column of Table 1 (the IDF scores are computed over a corpus of 30M product titles obtained from PriceGrabber). As can be seen, taking IDF weights into account, the Cosine similarity score of the matching title pair in row (a) increases but is still much below the similarity scores for non-matching titles in rows (b) and (c).

One of the main reasons for the poor performance of the Cosine similarity metric is that product titles for the same entity across feeds have fairly diverse representations. Some titles could be missing important tokens like brand (e.g., "nikon" in Title 1 of row (a)), while others may contain extraneous tokens corresponding to product specifications that are not critical for identifying the product (e.g., Title 1 in row (a)). This hurts the similarity score of the matching title pair in row (a). Furthermore, in some instances, IDF weights do not accurately capture the importance of tokens. In Table 2, we show the IDF weights (in parenthesis) assigned to tokens belonging to the titles in Table 1.

As can be seen, in Title 1 of row (a), several extraneous tokens like "3872" and "2592" are assigned higher weights than the model number "d200". Similarly, in Title 1 of row (c), important tokens like "charger", "2800mah" and "rechargeable" that describe the battery charger accessory are assigned low weights compared to the model number "fe140". A shortcoming of IDF is that it assigns a single weight to each token proportional to the inverse of its global frequency in the feeds independent of the context. Essentially, IDF does not take into account the relevance of a token to its product title context when computing its weight.

To overcome the above-mentioned challenges associated with using traditional similarity metrics to match product titles, we leverage the web. Our approach to matching product titles involves three key steps: (1) *Enrich* each product title by filling in missing tokens that frequently occur in the broader context of the product title, (2) Compute an *importance score* for each token in the enriched product title based on its power to identify the underlying product, and (3) Match enriched titles with tokens weighted by their importance scores. We use web search engines in the first two steps to determine an expanded context for each product title in order to enrich it, and to compute importance scores. Web search engines allow us to efficiently sift through billions of pages to identify the most relevant web pages for a product.

Our main contributions can be summarized as follows:

1) We present an end-to-end system architecture for matching product titles with varying formats. The system is completely unsupervised, and performs a sequence of tasks starting with enrichment of product titles and computation of token weights, followed by matching the potentially co-referent enriched title pairs.

2) We propose a method for enriching product titles with (missing) keywords that appear frequently in the context defined by search engine results.

3) We assign an importance score to each token in the enriched title based on its ability to retrieve other tokens (of enriched title) in search results. Our matching algorithm calculates the Cosine similarity between enriched title pairs with tokens weighted by their importance scores.

4) We propose an optimization for reducing the number of calls to the search engine. Our optimization predicts approximate enrichments and importance scores for a product title based on past enrichments for similar-structured product titles.

5) In experiments with real-life shopping datasets, our matching algorithm achieves higher F1 scores compared to IDF-based Cosine similarity. Furthermore, our enrichment and importance score prediction optimizations reduce the number of search queries significantly without adversely impacting matching accuracy.

Table 1: Matching and non-matching product titles.

#	Title 1 (PriceGrabber)	Title 2 (CNET)	Jaccard Similarity	Cosine Similarity
(a)	d200 10.2 megapixel digital slr camera body with lens kit - 18 mm - 135 mm (2.5" lcd - 7.5x optical zoom - 3872 x 2592 image)	nikon d200	0.045	0.19
(b)	dcr-ip55 handycam (2.5"" hybrid lcd)	sony handycam dcr-sx41 (red)	0.125	0.27
(c)	fe-140 digital camera battery charger replacement for 4 aa nimh 2800mah rechargeable batterie	olympus fe-140	0.07	0.47

Table 2: IDF scores for product title tokens.

#	Title 1 (PriceGrabber)	Title 2 (CNET)
(a)	*d200(10.2)* 10.2(8) megapixel(8.5) digital(6.3) slr(9.4) camera(6.6) body(5.7) with(2.6) lens(6.1) kit(3.9) 18(4) mm(6.9) mm135(15.6) 2.5"(5.7) lcd(6.8) 7.5x(12.6) optical(6.8) zoom(7.8) *3872(12.6) 2592(12.2)* image(6.9)	nikon(9.5) d200(10.2)
(b)	dcrip55(13.7) handycam(11.3) 2.5""(5.7) hybrid(7.8) lcd(6.8)	sony(8) handycam(11.3) dcrsx41(16.5) red(4)
(c)	*fe140(15.6)* digital(6.3) camera(6.6) battery(5.6) *charger(6.7)* replacement(4.9) for(2.6) 4(3.7) aa(8.5) nimh(9.5) *2800mah(11) rechargeable(8.5)* batterie(5.6)	olympus(10) fe140(15.6)

2. MATCHING SYSTEM OVERVIEW

The input to our matching system consists of a pair of data feeds D_1, D_2, each containing product title records. Each record contains a variable length string that represents the title of a unique product. For a product title record t, we abuse notation a bit and use t to also denote the set of tokens (using space as delimiter) in t. Our system outputs matching product title pairs from the two data feeds that refer to the same underlying product entity.

As we saw earlier, product titles contained in data feeds are unstructured and have diverse formatting conventions – some may be missing important keywords while others may contain redundant keywords that are not required for identifying the product. This makes the task of matching product title pairs extremely challenging. Our matching system consists of four main components: (1) Enrichment, (2) Importance score computation, (3) Blocking, and (4) Pairwise matching. Below, we describe the high-level functionality of each component, deferring the presentation of algorithmic details to the next section.

(1) *Enrichment.* The enrichment module enriches product titles by adding relevant tokens like brand or product line that may be previously absent. It leverages web search engines to determine the broader context for a product title t, and then augments the title with high-frequency tokens in the context to get the enriched title $E(t)$.

(2) *Importance score computation.* Each token w belonging to an enriched title $E(t)$ is assigned an importance score $I_t(w)$ based on its power to identify the underlying product entity. This is computed based on the search results obtained by pairing each enriched title token w with the remaining tokens. $I_t(w)$ is proportional to the number of enriched title tokens retrieved in these results.

(3) *Blocking.* The enriched feeds $E(D_1), E(D_2)$ containing the enriched product titles with token importance scores are input to the blocking module. Comparing every pair of enriched titles to see if they match is extremely inefficient especially for large feeds containing hundreds of thousands of records. The blocking module uses prefix filtering [6] to prune away enriched title pairs that cannot possibly match.

(4) *Pairwise matching.* A subset of the enriched title pairs $E(t)$, $E(t')$ (returned by blocking) are compared, and output if they are found to match. More specifically, if the Cosine similarity between the titles with tokens weighted by their importance scores exceeds a pre-specified threshold, then titles t and t' are declared to be a matching pair.

The enrichment and importance score computation modules are the most expensive from a computational perspective – as they repeatedly invoke the search engine to enrich the product titles and calculate token weights. In Section 4, we propose an optimization to reduce the overall number of web search queries for enrichment and importance score calculation. Our optimization predicts enrichments and importance scores for product titles based on previously computed enrichments and importance scores for product titles with similar structure/templates.

3. ALGORITHMS

In the following subsections, we describe the algorithms at the core of the four components of our matching system.

3.1 Enrichment

The enrichment module is responsible for filling in missing relevant tokens in the product title. Our enrichment solution described in Algorithm 1 achieves this by first determining the expanded context for the product title, and then adding to the title frequently occurring tokens in the context.

Algorithm 1 uses the web to determine the broader context for an entity given its product title. The web is a vast repository of documents containing information on virtually every possible entity or topic, from the very esoteric to the

Algorithm 1 ENRICH

Input: Product title t, support threshold τ;
Output: Enriched title $E(t)$;

Get top-K web search result titles $t_1, \ldots, t_K$ by issuing t as query;
For each token w, let $freq(w)$ be the number of distinct titles t_i that contain w;
/* Compute high-frequency tokens */
Let $w_1, w_2, \ldots, w_q$ be tokens with $freq(w_j) > \tau$ arranged in decreasing order of $freq(w_j)$;
Let j be the index for which $freq(w_j) - freq(w_{j+1})$ is maximum;
High-frequency tokens $h = \{w_1, \ldots, w_j\}$;
$E(t) = t \cup h$;
return $E(t)$;

Algorithm 2 COMPUTE_IMPORTANCE_SCORES

Input: Enriched title $E(t)$, support threshold τ;
Output: Token importance scores $I_t(w)$;

for each token $w \in E(t)$ **do**
  $I_t(w) = 0$;
  for each token $w' \in (E(t) - \{w\})$ **do**
    Get top-K web search result titles $t_1, \ldots, t_K$ by issuing the keyword pair w w' as query;
    for each token $w'' \in (E(t) - \{w, w'\})$ **do**
      Let $freq(w'')$ be the number of distinct titles t_i that contain w'';
      if $freq(w'') > \tau$ **then** $I_t(w) = I_t(w) + 1$;
    end for
  end for
end for
return $\{I_t(w) : w \in E(t)\}$;

extremely common. For product entities, the web contains dedicated pages containing product information like title, price, description, reviews, ratings etc. in a wide range of web sites like Wikipedia, and aggregator sites like PriceGrabber, CNET, Amazon, Yahoo!, MSN, etc. These sites contain mentions of product entities in a variety of different formats. Thus, we can use the different representations of a given entity on the web, and also the keywords that occur in the vicinity of entity references within web pages to establish the extended context for the entity.

The key question, of course, is how to sift through the billions of pages on the web to locate the most relevant pages with mentions of a product entity. This is exactly the task that web search engines are good at. Given a query with a subset of keywords describing a unique product, search engines return the top web pages related to the product. To surface the most relevant pages for a given query, search engines exploit a range of signals like font, size, and position of query keywords within a page, presence of query keywords in anchor text for the page, PageRank, query and page categories (e.g., electronics, music, apparel), etc.

Search engines assign a ranking to the returned web pages with the ranking algorithm designed to place the most relevant pages at the top. Consequently, in our work, we only consider the top-K results returned by the search engine since these are very likely to contain most of the highly relevant product pages. For each result of a query, the search engine interface returns the page URL, title, and a short summary comprising a few sentences from the page containing query terms.

Our enrichment procedure starts by invoking the search engine interface with the product title t as the search query. It then considers the titles t_i of the top-K results returned by the search engine as the expanded context for the underlying product entity. Finally, it uses the high-frequency tokens in the context to enrich the input title t. To identify the high-frequency tokens, Algorithm 1 considers the tokens w with frequency $freq(w)$ greater than the support threshold τ and clusters them into two sets such that the minimum gap in frequencies between the two sets is maximized. The tokens in the cluster with the higher frequencies are then considered to be the high-frequency tokens h. Note that clustering helps to separate the more relevant tokens with higher frequencies from the less relevant ones which are pruned. The final enriched title $E(t)$ is then obtained by adding the high-frequency tokens in h to the title t.

In our experiments, we found that selecting $K = 20$ and $\tau = 0.2$ gave enrichments with the best matching performance. Also, in our implementation, we shrunk the gap $freq(w_j) - freq(w_{j+1})$ in the frequencies of tokens w_j and w_{j+1} by a factor $(0.9)^j$ – this further biased token selection towards higher frequencies and increased relevance.

Table 3 shows the enrichments for the product titles in Table 1 – the new keywords that are added to the enriched titles are shown in bold. As can be seen, the enriched representations now contain important tokens like brand (e.g., "nikon", "sony") that were previously missing.

3.2 Importance Score Computation

Not all tokens in an enriched title are equally important – clearly, certain tokens like model number are more important than others. Intuitively, a token in an enriched title is important if it is critical for identifying the underlying product entity. For example, in Title 1 in row (a) of Table 1, the model number "d200" is more important than "zoom" or "2592" since it plays a bigger role in identifying the product. Consequently, we assign importance scores to tokens based on their identification power.

The main question then is: how do we measure the identification power of a token? For this, we rely on web search. A key observation here is that the enriched product title contains several relevant keywords for describing (and uniquely identifying) the corresponding product. As a result, if the search results for a token contain many repetitions of keywords in the enriched title, then the results are very likely for the underlying product and we can conclude that the token has high identification power. Thus, the number of repeatedly appearing enriched title keywords in the web search results for a token is a good proxy for its identification power.

In practice, however, individual tokens can be ambiguous, causing search results to span the diverse meanings for the token. So to disambiguate queries, we pair each token w with other tokens w' in the enriched title, and consider the number of frequently occurring title tokens in the web search results of all the pairs as a measure of importance of the token w. Algorithm 2 describes our procedure for computing importance scores for the tokens of an enriched title $E(t)$. Notice that when computing the importance score $I_t(w)$ for

Table 3: Enriched product titles and importance scores.

#	Enriched Title 1 (PriceGrabber)	Enriched Title 2 (CNET)	Cosine Similarity
(a)	**nikon**(35) *d200(69)* 10.2(55) megapixel(54) digital(23) slr(60) camera(34) body(35) with(26) lens(33) kit(17) 18(31) mm(20) mm135(2) 2.5"(19) lcd(35) 7.5x(93) optical(44) zoom(42) *3872(58) 2592(38)* image(14)	nikon(1) d200(1)	0.36
(b)	**sony**(2) dcrip55(9) handycam(6) 2.5""(5) hybrid(5) lcd(4)	sony(2) handycam(3) dcrsx41(7) red(4)	0.18
(c)	*fe140(22)* digital(21) camera(21) battery(7) *charger(14)* replacement(11) for(8) 4 aa(15) nimh(24) *2800mah(27)* *rechargeable(20)* batterie(16)	olympus(1) fe140(1)	0.25

token $w \in E(t)$, we only consider tokens w'' in $E(t)-\{w, w'\}$ that frequently occur in the search result titles for keyword pair $w\ w'$. This is because the search result titles for query $w\ w'$ will most likely contain the keywords w and w' irrespective of how important tokens w or w' are; however, if the result titles also repeatedly contain other keywords from $E(t)-\{w, w'\}$, then it is a strong indication that either token w or w' or both are important.

Traditional IR approaches use IDF scores to capture the importance of tokens. The basic premise is that rare words in a corpus have high identification power, and so are more important. Thus, IDF assigns each token a single score that is inversely proportional to its global frequency in the corpus. In our experiments, we found that IDF usually assigns high scores to important tokens like model number. However, there are also instances when it assigns high scores to redundant keywords like "2592" (in row (a) of Table 2) because they are rare and low scores to important keywords like "charger" (in row (c) of Table 2). A general drawback of IDF is that it assigns each token a single score based only on global context independent of the co-occurring keywords in the token's local product title context. Furthermore, IDF score computation requires a base corpus, and token scores can vary depending on the choice of corpus.

Our importance score computation approach is very different from the IDF approach. For each token, we directly estimate its identification power by measuring the overlap between its local product title context and web search results, and use the degree of overlap as a measure of the token's importance. Thus, our approach computes importance scores taking into account only local contextual information for each product title and web search results – this has multiple consequences: (1) Our approach does not require a base corpus and is not sensitive to the choice of corpus, and (2) A token can have multiple importance scores depending on the local product title context that it appears in.

The importance scores computed by our approach for tokens in enriched product titles are shown in parenthesis in Table 3. As can be seen, in Title 1 of row (a), the model number "d200" with high identification power is also assigned a high importance score while the less relevant tokens like "zoom" and "2592" are assigned much lower importance scores. Similarly, important tokens like "charger" and "2800mah" in Title 1 of row (c) are also assigned high scores relative to the model number "fe140". In contrast, IDF assigns high scores to tokens "3872" and "2592" relative to the model number "d200" (see row (a) of Table 2). Also, "charger" and "2800mah" are assigned low IDF scores of 6.7 and 11 compared to the model number "fe140" which is assigned an IDF score of 15.6 (see row (c) of Table 2). Thus, our importance scores capture the significance of a token in a product title better than IDF scores.

Algorithm 3 MATCHING_PAIRS

Input: Titles t, t', support threshold τ, similarity threshold γ;
Output: Match/No match;

$E(t)$ =ENRICH(t, τ);
$\{I_t(w)\}$ =COMPUTE_IMPORTANCE_SCORES$(E(t), \tau)$;
$E(t')$ =ENRICH(t', τ);
$\{I_{t'}(w)\}$ =COMPUTE_IMPORTANCE_SCORES$(E(t'), \tau)$;
if $\frac{\sum_{w \in (E(t) \cap E(t'))} I_t(w) \cdot I_{t'}(w)}{\sqrt{\sum_{w \in E(t)} I_t(w)^2} \sqrt{\sum_{w \in E(t')} I_{t'}(w)^2}} > \gamma$ **then**
  return match;
else
  return no-match;
end if

Observe that Algorithm 2 issues a search query for every keyword pair $w\ w'$, where $w' \in E(t)-\{w\}$, when computing the importance score $I_t(w)$ for token w. This can result in a large number of calls to the search engine which can get computationally expensive. One way to reduce the number of queries is to randomly select a subset of tokens E' from $E(t) - \{w\}$, and then issue queries for only keyword pairs $w\ w'$ where $w' \in E'$. The importance score $I_t(w)$ for w is then computed by summing up the number of times tokens from $E(t) - \{w, w'\}$ frequently occur in the search results for the random queries.

3.3 Blocking

Comparing every pair of enriched titles to see if they match can be prohibitively expensive for large data feeds containing hundreds of thousands of records. Clearly, titles whose enrichments have no tokens in common cannot possibly match, and we can skip comparing such title pairs. The blocking module uses prefix filtering techniques [6] that look for overlap among small filtered prefixes of the titles to generate potentially matching title pairs.

3.4 Pairwise Matching

Algorithm 3 describes our matching algorithm to determine whether two product titles t and t' represent the same entity or not. Our matching procedure calculates the Cosine similarity between the two enriched titles with importance scores as token weights. The titles are a declared to be a match if their similarity exceeds the similarity threshold γ.

Table 3 shows the similarity scores between enriched title pairs using our matching approach. Compared to traditional IDF-based similarity between the original product titles (see Table 1), our matching algorithm returns higher similarity scores for matching titles (in row (a)) and lower scores for non-matching titles (in rows (b)-(c)). Moreover, observe that the similarity scores for matching titles are greater than those for non-matching titles, thus allowing the matching title pairs to be distinguished from the non-matching ones.

4. OPTIMIZATIONS

Issuing queries to the search engine and subsequently processing the search results to compute product title enrichments and importance scores is the most expensive step in our matching solution. In this section, we propose an optimization to reduce the number of search engine invocations.

Frequently, within a data feed, there are groups of product titles with very similar structure. For example, product titles for the same product line in Table 4 share a common template with the only difference being the model number. Now, within a group of similar-structured titles, if a sufficient number of titles have already been enriched, then we can use these enrichments to predict approximate enrichments for the remaining titles in the group. Similarly, we can leverage previously calculated importance scores for enriched titles to predict approximate importance scores for new enrichments. This is the key idea underlying our optimization that predicts enrichments and importance scores for product titles.

Algorithm 4 describes our procedure for predicting the enrichment and importance scores for a given input title t. The procedure starts by using Jaccard similarity (with threshold λ) to identify a subset S of previously enriched titles that are similar to t. Note that the subset S can be efficiently retrieved by indexing the entire set T of enriched titles and using prefix filtering techniques described in [6]. Now, if S contains very few titles (less than δ), then we simply compute the enrichment and importance scores using Algorithms 1 and 2 instead of trying to predict them. Else, we find the tokens F that consistently and frequently occur in the enrichments of the similar titles in S – clearly, these are very likely to also belong to the enrichment for t and so we add these to t to obtain its enrichment $E(t)$. These frequent tokens in the enrichments cover brand names and generic keywords like camera, etc. Observe that some of the tokens in F may not occur in the titles themselves. For each token w in F, we set its importance score $I_t(w)$ to the average of its importance scores $I_{t'}(w)$ for titles $t' \in S$. Furthermore, to overcome the variations in token importance scores across titles, we normalize scores by dividing each token score by the maximum score in the title.

Now, certain discriminating tokens like model numbers will not be frequent in the enrichments, and thus will not be in F. In order to predict importance scores for these (infrequent) tokens in $E(t) - F$, we introduce the notion of landmark tokens – these are basically tokens in t that occur frequently in the titles in S. We store landmark tokens in the set L. For each token w occurring in a title $t' \in S$, we construct a signature $sig_{t'}(w)$ based on w's distance from the landmark token set. More specifically, the signature $sig_{t'}(w)$ consists of pairs $(w', dist_{t'}(w, w'))$ for each landmark token w'. Here, $dist_{t'}(w, w')$ is the distance (in terms of tokens) of w from w' in the sequence-of-tokens representation for t'.

Algorithm 4 PREDICT

Input: Title t, set T of enriched titles, their enrichments $E(t')$, and normalized token importance scores $I_{t'}(w)$;
Input: Threshold parameters λ, δ, σ, μ;
Output: Predicted enrichment $E(t)$, importance scores $I_t(w)$;

$S = \{t' : t' \in T \wedge \text{JACCARDSIM}(t, t') > \lambda\}$;
if $(|S| < \delta)$ **then return** $\emptyset$, $\emptyset$;

/* Compute enrichment */
Let $E(S) = \{E(t') : t' \in S\}$;
Let $freq(w, E(S))$ = frequency of token w in $E(S)$;
$F = \{w : freq(w, E(S)) \geq \mu\}$;
$E(t) = F \cup t$;

/* Predict importance scores for tokens in F */
for all tokens $w \in F$ **do**
  $I_t(w) = \text{avg } \{I_{t'}(w) : w \in E(t') \wedge t' \in S\}$;
end for

/* Predict importance scores for tokens in $E(t) - F$ */
Let $freq(w, S)$ = frequency of token w in S;
$L = \{w : w \in t \wedge freq(w, S) \geq \sigma\}$;
for all titles $t' \in S$ **do**
  for all tokens $w \in t'$ **do**
    $sig_{t'}(w) = \{(w', dist_{t'}(w, w')) : w' \in L\}$;
  end for
end for
for all tokens $w \in (E(t) - F)$ **do**
  $sig_t(w) = \{(w', dist_t(w, w')) : w' \in L\}$;
  $I_t(w) = \text{avg } \{I_{t'}(w') : w' \in t' \wedge t' \in S \wedge sig_{t'}(w') \approx sig_t(w)\}$;
end for
return $E(t), \{I_t(w) : w \in E(t)\}$;

Table 4: Product titles along with importance scores.

#	Product Title	Importance Scores
(a)	sony cybershot dsct20 black	0.4 0.6 1 0.8
(b)	sony cybershot dsch10 black	0.4 0.6 1 0.8
(c)	sony cybershot dsct2 blue	0.4 0.6 1 0.8

Note that $dist_{t'}(w, w')$ is negative if w occurs to the left of landmark token w' in the title t', and positive if it occurs to the right. In case w' does not belong to t', then $dist_{t'}(w, w')$ is ∞.

Now, since titles in S and t follow the same template, tokens of the same type (e.g., model number, color) occur in similar positions across the titles and will thus have near-identical signatures. As a result, even though an individual token w in $E(t) - F$ is not frequent in the enrichments, w's signature will be frequent due to tokens of the same type as w in other titles. Furthermore, these tokens belonging to other titles in S, of the same type and with near-identical signatures as w, will also have similar importance scores as w. Thus, we can predict the importance score of token $w \in (E(t) - F)$ by averaging the importance scores of tokens belonging to $t' \in S$ with very similar signatures (that agree on the distances to most of the landmark tokens).

EXAMPLE 1. *Consider a new product title t = "sony cybershot dscw80 black" and let the product titles in Table 4 be the set S of titles similar to t. Let the enriched product titles be the same as the raw product titles. Furthermore, let the importance scores for the four tokens in each title be 0.4, 0.6, 1 and 0.8 as shown in Table 4. For support threshold $\mu = 2$, the frequent tokens F in the enrichments are {"sony", "cybershot", "black"}. Thus, the enriched title $E(t) = F \cup t$ = {"sony", "cybershot", "dscw80", "black"}. The importance scores for tokens in F are computed by averaging their importance scores across the enrichments. Thus, the importance scores (shown in parenthesis) for tokens in F are: "sony" (0.4), "cybershot" (0.6), and "black" (0.8). Next, we show how the importance score for token "dscw80" in $E(t) - F$ is calculated. For $\sigma = 2$, the landmark tokens L are {"sony", "cybershot", "black"} – these are the frequent tokens in the titles in S. Now, the signature for token "dscw80" in t is {("sony", 2), ("cybershot", 1), ("black", -1)}. The tokens "dsct20" and "dsch10" in their respective titles (in rows (a) and (b)) have the same signature. Thus, the importance score for token "dscw80" is 1 which is the average of the scores for tokens "dsct20" and "dsch10".* □

5. EXPERIMENTS

In this section, we present a detailed evaluation of our proposed enrichment, importance score computation, and optimization algorithms.

Datasets: We use two datasets in our experiments.

- **Abt-Buy dataset**[1]**:** This contains 1081 products from Abt and 1092 product titles from Buy. The matching quality for this dataset is also evaluated in [16].

- **PG-CNET dataset:** This is a smaller dataset containing 340 matching camera and appliance product titles from PriceGrabber and CNET. The ground truth is generated using MPN-UPC values, if available. For those products that did not have the MPN-UPC value, matching pairs are identified manually.

Evaluation Metrics: We use the popular F1 score measure to evaluate the matching quality. To characterize the effectiveness of enrichment prediction, we define two metrics: *Enrichment Prediction Coverage (EPC)* and *Enrichment Prediction Quality (EPQ)*. *EPC* is the fraction of titles for which enrichments as well as importance scores are predicted, and *EPQ* is the average Cosine similarity between the actual and the predicted importance score vectors over predicted titles. Observe that the cost savings in search queries (defined as the fractional decrease in search queries issued to get the enrichments and the importance scores) is equal to *EPC*. Also, *EPQ* measures the similarity between the actual and the predicted importance scores.

Schemes Compared: We evaluate the matching quality for the following two schemes:

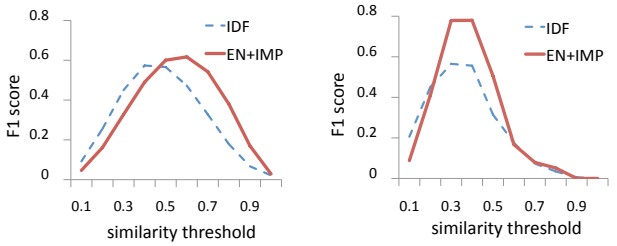
- **IDF:** We use the Cosine similarity between raw titles with the IDF weighting scheme as the baseline for comparison.

- **EN+IMP:** This is our matching algorithm with enriched titles and importance scores. We set the default values for the number of search results K and support threshold τ (in Algorithms 1 and 2) to 20 and 0.2 respectively.

- **EN+IMP(Opt):** This is our EN+IMP algorithm with enrichment prediction (Algorithm 4).

We use the Bing Search API for the search results. Furthermore, we perform blocking as discussed in Section 3.3 for all the above three schemes. Our experimental results demonstrate that (1) our web-based enrichments and importance scores improve the matching quality compared to IDF-based Cosine similarity, and (2) the search query volume reduces substantially with our proposed enrichment prediction scheme.

5.1 Matching quality of IDF vs EN+IMP



(a) Abt-Buy (b) PG-CNET

Figure 1: Matching quality of IDF vs EN+IMP.

The graphs in Figure 1 show the F1 scores for the IDF and EN+IMP schemes as the similarity threshold (parameter γ in Algorithm 3) is varied. As can be seen from the figure, our EN+IMP matching scheme with web-based enrichments and importance scores outperforms IDF-based matching. For instance, for the Abt-Buy dataset, IDF performs the best at $\gamma = 0.4$ with an F1 score of 0.57, as compared to our EN+IMP scheme which achieves a higher F1 score of 0.62 at $\gamma = 0.6$. The difference in F1 scores is much higher in the PG-CNET dataset, with IDF having its highest F1 score of 0.57 at $\gamma = 0.3$, as compared to 0.78 with our EN+IMP scheme.

Also, an important point to note here is that our method is completely unsupervised. The F1 scores obtained with our scheme, specifically for the Abt-Buy dataset, are comparable to the scores presented in [16], obtained using supervised techniques. The PG-CNET dataset performs better than the Abt-Buy dataset, mainly because the product titles in PG-CNET are more ambiguous than the Abt-Buy titles – missing key tokens and presence of extraneous words.

Table 5 compares the normalized scores assigned by IDF and EN+IMP to important keywords for a few example product titles. For a given product title, the brand name and model number are the representative tokens and are expected to get high scores. As can be seen, the IDF scores for brand names are significantly lower than the importance scores computed by EN+IMP. This is because brand names like Cannon and Samsung despite being important appear in many product titles, and so get assigned low IDF scores. Also, model number terms get higher importance scores compared to IDF scores most of the times.

[1]Available at `http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution`

Table 5: Normalized scores for brand name and model number in IDF and EN+IMP.

Product title	Brand (IDF)	Brand (EN+IMP)	Model (IDF)	Model (EN+IMP)
Canon Yellow Ink Cartridge - Yellow - **CLI8Y**	0.17	0.42	0.52	0.72
Canon Black Ink Cartridge - Black - **PG40BK**	0.20	0.47	0.64	0.68
Toshiba SD-P91S Portable DVD Player	0.37	0.42	0.75	0.68
Samsung LN40A650 40' LCD TV	0.26	0.43	0.61	0.62

Table 6: F1 scores with varying number of web results (K) for Abt-Buy.

similarity threshold	$K = 5$	$K = 10$	$K = 20$	$K = 25$
0.3	0.329	0.325	0.328	0.324
0.4	0.503	0.492	0.490	0.479
0.5	0.609	0.595	0.600	0.587
0.6	0.614	0.616	**0.617**	0.601
0.7	0.530	0.540	0.542	0.542

5.2 Parameter sensitivity analysis for EN-IMP

We perform a series of experiments to identify the optimal parameter settings for our algorithms. In this section, we present the parameter sensitivity results for the Abt-Buy dataset. The number of web results for each query (K) and support threshold (τ) are two important parameters in our matching scheme (Algorithms 1 and 2). Enrichments and importance scores are computed based on the search results fetched. The support threshold should not be high so as to miss some important keywords; and at the same time it should not be very low to allow erroneous keywords.

Table 6 shows the F1 matching scores with varying number of web results (K) for similarity match thresholds (γ) between 0.3 and 0.7. As can be seen, we obtain the highest F1 score at $K = 20$. Figure 2 shows the F1 scores for different values of support threshold (τ). Setting τ to 0.2 gives the best F1 score. So we set $K = 20$ and $\tau = 0.2$ in all our experiments. In general, we found that the matching quality of EN+IMP to be more sensitive to the support threshold (τ), as opposed to the number of search results (K). However, K plays an important role in limiting the number of results processed.

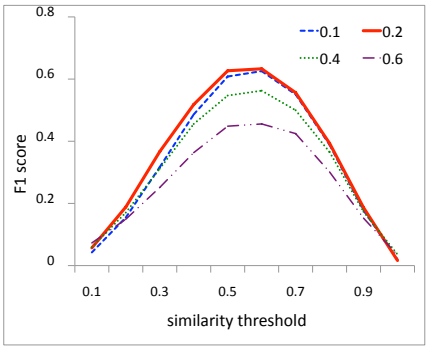


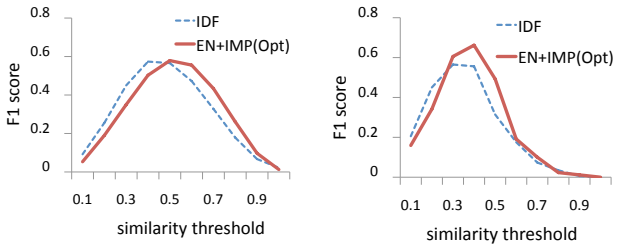
Figure 2: Varying support threshold τ for Abt-Buy.

5.3 Enrichment prediction results

Reducing the number of web searches is an important component of our matching scheme. In this section, we present a detailed evaluation of our enrichment prediction algorithm EN+IMP(Opt). We perform enrichment prediction in a batch setting. Here we use an initial seed set of (randomly selected) titles and their enrichments (B), and use this seed set for predicting enrichments and importance scores for the rest of the corpus. Recall that EPC captures the cost savings in the number of search queries and EPQ measures the quality of prediction - we compute these metrics for the non-seed titles. We also evaluate the effect of seed set size $|B|$ on EPC and EPQ. In our experiments, we set the default value of |B| for the Abt-Buy and PG-CNET datasets to 1000 and 133 respectively.

5.3.1 *Matching quality with enrichment prediction*

The graphs in Figure 3 compare the matching quality of EN+IMP with enrichment prediction to IDF, for both the datasets. From the graph 3(a), we observe that the EN+IMP(Opt) does marginally better than the IDF scheme, for the Abt-Buy dataset. The difference in performance is higher for the PG-CNET dataset, as seen from the graph 3(b). The highest F1 score with IDF is 0.57 as compared to 0.61 with EN-IMP(Opt). Furthermore, the EPC value is 0.8 implying that enrichments were predicted for 80% of remaining titles. This shows that with our prediction schemes based on the templatized structure of product titles, we can reduce the number of web searches significantly while preserving matching quality. One important observation to make here is the prediction algorithm does well if the initial seed set of titles is diverse and covers various title templates. We have not focussed on this batch selection problem. However, random selection of the seed set does ensure good coverage for frequently occurring title templates.



(a) Abt-Buy (b) PG-CNET

Figure 3: Match quality with prediction.

5.3.2 *Parameter sensitivity analysis for enrichment prediction*

Table 7 shows the EPC, EPQ values and the corresponding highest F1 score in the setting where we vary $|B|$ for the Abt-Buy dataset. Observe that, with a batch size of 1000, EPC and EPQ values are 0.87 and 0.89 respectively,

Table 7: EPC-EPQ values with highest F1 score for different $|B|$ for Abt-Buy.

$\|B\|$	EPC	EPQ	Highest F1 score
250	0.86	0.87	0.46
500	0.92	0.86	0.53
750	0.93	0.83	0.56
1000	0.87	0.89	**0.59**

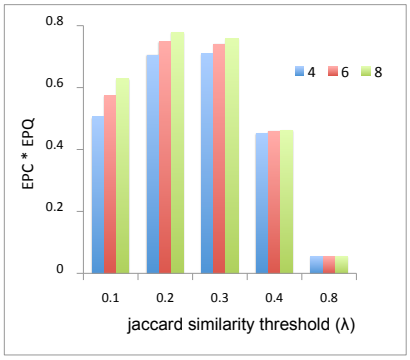


Figure 4: EPC*EPQ for varying λ and $\mu = 4, 6, 8$ for Abt-Buy.

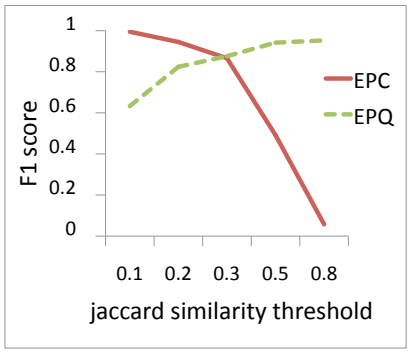


Figure 5: EPC vs EPQ with $\mu = 8$ for Abt-Buy.

Table 8: Example of enrichment prediction.

Token	Actual Score	Predicted Score
sony	0.4	0.41
cybershot	0.6	0.68
dscw80	1.0	0.99
black	0.8	0.70
EPQ	1.000	0.996

which implies that the coverage and quality of enrichment prediction are high. A batch size of 1000 and an EPC of 0.87 implies that with an initial seed set size of 1000, we could predict 87% of the remaining enrichments/importance scores. The F1 score achieved in this setting is 0.59, which is more than that of IDF which has a highest score of 0.57.

For a batch size of 1000, we analyze the sensitivity of two important parameters, support size μ and Jaccard similarity threshold λ (Algorithm 4). Figure 4 shows the EPC * EPQ values for different values of μ and λ. We want both EPC and EPQ values to be high; a high EPC value implies that we have predicted a high number of enrichments and a high EPQ value implies that the quality of prediction is high. As we can see, a support size μ of 8 performs the best at all thresholds. This can be attributed to better quality enrichment predictions at higher μ values. We further analyze the EPC and EPQ values for $\mu = 8$ to find an optimum value for λ. For $\mu = 8$, Figure 5 shows EPC and EPQ as a function of the Jaccard similarity threshold (λ). EPC values decrease with an increase in λ value, whereas enrichment quality EPQ increases. Observe that, a threshold value λ of 0.3 strikes a good balance between enrichment prediction coverage and quality, for the Abt-Buy dataset.

Table 8 gives an example of the importance scores with enrichment prediction. The predicted scores are very close to the actual scores; thus the Cosine similarity between the actual and the predicted scores (EPQ) is close to 1.

6. RELATED WORK

The problem of matching records has been extensively studied in the literature – [12] and [11] present comprehensive surveys of research work in the area. Prior research on matching has primarily focused on three aspects: (1) similarity functions for matching individual attribute values, (2) unsupervised and supervised techniques for matching records containing multiple attributes, and (3) blocking methods for determining record pairs to match.

Developing robust similarity metrics for attribute comparison is essential since the input data to the matching process is often noisy (e.g., due to spelling errors) and/or in different formats (e.g., phone numbers with and without '-'). There is a large body of work on approximate string similarity measures like edit distance or Cosine similarity applied to IDF-weighted string tokens, q-grams, etc. [12, 17, 5].

Existing matching approaches use attribute-level similarity measures as the basic building block, and can be either unsupervised or supervised. Unsupervised methods combine attribute-level similarity scores to define record-level similarity which is then compared with a fixed threshold to identify matching records. In the supervised approach, labeled record pairs with attribute similarity values as features are used to train a classifier. The trained classifier is then used to predict whether a new record pair matches or not. In essence, one can view the classifier as learning record-level similarity measures from attribute-level ones. A detailed evaluation of the matching performance of existing unsupervised and supervised techniques on two real-world product datasets is presented in [16]. In general, the authors find that conventional approaches result in low matching quality for product entities with supervised techniques (with multiple attributes) performing the best.

To handle continuously arriving product offers from merchants, [3] uses a voted perceptron classifier for online learning of similarity functions. In other work, [20] uses active learning to reduce classifier training data requirements. Finally, since running classifiers on all record pairs is expensive, blocking techniques are used to reduce the number of record pairs that enter into the matching phase. [9] contains a recent survey on the use of indexing techniques for blocking.

The focus of our work is not on record-level matching or blocking, but rather on devising a good attribute-level similarity function for product titles. Our matching algo-

rithm improves upon traditional Cosine similarity with IDF weights by leveraging the web to enrich product titles and assign token weights.

The enrichment step of our matching solution is closely related to previous work on query expansion [18], although the primary goal of query expansion is to improve recall as opposed to matching quality. Given the widespread prevalence, efficiency, and relevance of web search engines, there has been a growing trend to exploit search engines for tasks such as word similarity computation [10], query classification [4], extracting model numbers from titles [21], etc. Matching is no exception and recently there has been interest in leveraging web search engines (or more generally, the web corpus) for the matching problem. For instance, [21] uses web search for author name disambiguation. It augments the author name with associated information (like co-author, institution, etc.) and uses the search results of the augmented query for disambiguation. [19] also employs web search engines to compute context vectors for short text snippets that are then used to determine similarity. Weights for tokens in the context vectors, however, are still computed using IDF.

[7, 8] explore the use of web data for identifying discriminating token subsets (called "IDTokenSets") of comprehensive title strings in a reference entity table. These IDTokenSets are then matched with candidate strings in documents in order to disambiguate entity mentions. The papers propose efficient techniques for deriving IDTokenSets: [7] uses web search while [8] employs a web corpus. Both [7, 8] assume the existence of a reference entity table with comprehensive title strings containing the most relevant tokens to identify an entity, and possibly extraneous tokens. However, this is a strong assumption and may not hold in practice – as shown in Table 1, important tokens like brand may be missing from product title strings. Observe that we also leverage web search results in our work, but to compute enrichments and token importance scores as opposed to IDTokenSets.

[15] considers the problem of matching unstructured offers to structured product descriptions. Tokens in the offer that match attribute values in the product are tagged with attribute names, and a feature vector is constructed based on the similarity between attribute values. The similarity function is then learned by training a logistic regression classifier using the features. Note that our matching algorithm is completely unsupervised and does not require training data. Moreover, our focus in this paper is on matching unstructured product title pairs, and our approach does not require product specification tables.

7. CONCLUSIONS

In this paper, we have proposed two novel ideas to tackle the matching problem for product titles: enrichment of titles with essential missing tokens and importance score computation that takes context into account. Both ideas leverage the web corpus effectively through web search engines. To reduce repeated invocation of search engines, we also presented a technique for predicting enrichments and importance scores. This exploits the templatized structure of product titles. The experimental results demonstrate that the proposed techniques are effective and outperform the widely used IDF technique with higher F1 scores. Enrichment prediction reduces search queries significantly while retaining matching effectiveness over IDF. Our experimental results validate our approach for product titles having distinctive tokens like brand name and model number (product titles of cameras and appliances). We plan to extend this approach to other domains like apparel, toys, software product titles (e.g., from the Amazon-Google dataset [16]) which may not contain representative tokens.

8. REFERENCES

[1] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *VLDB Journal*, 18(1), 2009.

[2] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM TKDD*, 1(1), 2007.

[3] M. Bilenko, S. Basu, and M. Sahami. Adaptive product normalization: Using online learning for record linkage in comparison shopping. In *ICDM*, 2005.

[4] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *SIGIR*, 2007.

[5] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, 2003.

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

[7] S. Chaudhuri, V. Ganti, and D. Xin. Exploiting web search to generate synonyms for entities. In *WWW*, 2009.

[8] S. Chaudhuri, V. Ganti, and D. Xin. Mining document collections to facilitate accurate approximate entity matching. In *VLDB*, 2009.

[9] P. Christen. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE TKDE*, 2011.

[10] R. L. Cilibrasi and P. M. B. Vitanyi. The google similarity distance. *IEEE TKDE*, 2007.

[11] C. F. Dorneles, R. GonÃğalves, and R. dos Santos Mello. Approximate data instance matching: a survey. *Knowledge and Information Systems*, 2011.

[12] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE TKDE*, 2007.

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

[14] M. A. Hernandez and S. J. Stolfo. Real-world data is dirty: Data cleansing and and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1), 1998.

[15] A. Kannan, I. E. Givoni, R. Agrawal, and A. Fuxman. Matching unstructured product offers to structured product specifications. In *KDD*, 2011.

[16] H. Kopcke, A. Thor, and E. Rahm. Evaluation of entity resolutoin approaches on real-world match problems. *PVLDB*, 3(1), 2010.

[17] C. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[18] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *SIGIR*, 1998.

[19] M. Sahami and T. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW*, 2006.

[20] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *SIGKDD*, 2002.

[21] Y. F. Tan, M.-Y. Kan, and D. Lee. Search engine driven author disambiguation. In *JCDL*, 2006.