Journal of Web Engineering, Vol. 13, No.5&6 (2014) 378-404

GATHERING WEB PAGES OF ENTITIES WITH HIGH PRECISION

BYUNG-WON ON

Department of Statistics and Computer Science, Kunsan National University

558, Daehak-ro, Gunsan-si, Jeollabuk-do 573-701, Republic of Korea

bwon@kunsan.ac.kr

MUHAMMD OMAR GYU SANG CHOI*

Department of Information and Communication Engineering, Yeungnam University

280, Daehak-ro, Gyeongsan-si, Gyeongsangbuk-do 712-749, Republic of Korea

m.omar.nazeer@gmail.com castchoi@ynu.ac.kr

JUNBEOM KWON

Department of Software Science, Dankook University

152, Jukjeon-ro, Suji-gu, Yongin-si, Gyeonggi-do 448-701, Republic of Korea

32091933@dankook.ac.kr

Received June 4, 2014
Revised September 2, 2014

A search engine like Yahoo looks for entities such as specific people, places, or things on web pages with search queries. Depending on the granularity of query keywords and performance of a search engine, the retrieved web pages may be in very large number having lots of irrelevant web pages and may be also not in proper order. It's infeasible to manually decide the relevance of each web page due to the large number of retrieved web pages. Another challenge is to develop a language independent relevance classification of search results provided by a search engine. To improve the quality of a search engine it is desirable to automatically evaluate the results of a search engine and decide the relevance of retrieved web pages with the user query and the intended entity, the query is all about. A step towards this improvement is to prune irrelevant web pages out by understanding the needs of a user in order to discover knowledge of entities in a particular domain. We propose a novel method to improve the precision of a search engine which is language independent and also free from search engine query logs and user clicks through data (widely used in recent times). We devise language independent novel features to build support vector machine relevance classification model using which we can automatically classify whether a web page retrieved by a search engine is relevant or not to the desired entity.

Key words: Precision, Support Vector Machines, Supervised Learning, Query Expansion
Communicated by: D. Schwabe & M. Bielikova
* *Corresponding Author*

1 Introduction

The section begins with the motivation behind our work with an example and then defines the problem by introducing some notations, definitions and relevant figures supporting our intuition and in the end summaries our contributions.

1.1 Motivation

The world-wide web commonly known as the web is a network of documents, some of which are hyperlinked with each other via Internet. The web documents or pages may contain text, images, videos, and other multimedia. Since 1989, the web has gradually become the centre of information source around the world. We can imagine that without relying on the web each individual does nothing in his or her daily life. In particular, Wardrip-Fruin and Montfort mentioned in their book [15] that the web was developed to be a pool of human knowledge and human culture, which would allow collaborators in remote sites to share their ideas and all aspects of a common project.

Recently, regardless of human users or software agents, it has been in the limelight to discover knowledge of an entity from web pages. However, it is a non-trivial task because there are a huge large number of web pages on the web. It is recently known that Google has about 98 petabytes of web sites indexed by it. In general, to gain knowledge of an entity of interest, we usually use query keywords describing the entity well in order to search web pages related to the entity using a search engine like Yahoo, Google or Bing. However, some of retrieved web pages may be irrelevant with the target entity. In this sense, the web contains worthless information about the entity as well. Therefore, it is considerably significant to filter irrelevant web pages out so as to discover knowledge of the entity. In addition, note that a large number of web pages are retrieved from a search engine. This is too large to manually decide the relevance of web pages in real time.

We can evaluate a search engine in terms of effectiveness (ability to find right information) and efficiency (how quickly information is found). To measure the effectiveness of a search engine in a specific application is very valuable [3]. For a given query, and a specific definition of relevance, we can more precisely define effectiveness as a measure of how well the ranking produced by the search engine corresponds to a ranking based on user relevance judgments. Efficiency is defined in terms of the time and space requirements for the algorithm that produces the ranking. There is no reliable technique that significantly improves effectiveness that cannot be incorporated into a search engine due to efficiency considerations and this may change in the future as suggested in [3].

The problem that we have addressed in this paper is a step towards discovering the knowledge of an entity efficiently but yet effectively. We tackle the problem of automatically determining whether or not each web page retrieved by a search engine is relevant with a target entity. This problem is also known as the automatic evaluation of search engine results. To demonstrate the need for a solution to this problem, let us present a real example drawn from the Google search engine.

Example 1. Figure 1 is a screen-shot image of top-10 web pages retrieved by Google using a query "troy." The search result is a mixture of web pages related to movies, historical and modern cities, a state university, and private companies. If the target entity is a troy movie starring by Brad Pitt, the web pages indicating the university, cities, and companies are irrelevant ones. In general, despite using appropriate queries, it is not true that every retrieved web page is relevant with the target entity.

In addition, the number of the search result is very large. As shown in Figure 1, Google shows about 216 million web pages. Thus, it is infeasible to manually decide if each web page retrieved is relevant or not. Moreover majority of search engine users see only first page of search result i.e. he or she may see only top twenty pages ordered and retrieved by the search engine. The order of retrieved lists of URLs provided by a search engine is not perfect in general due to different reasons, e.g. may be incorrect use of query keywords etc.

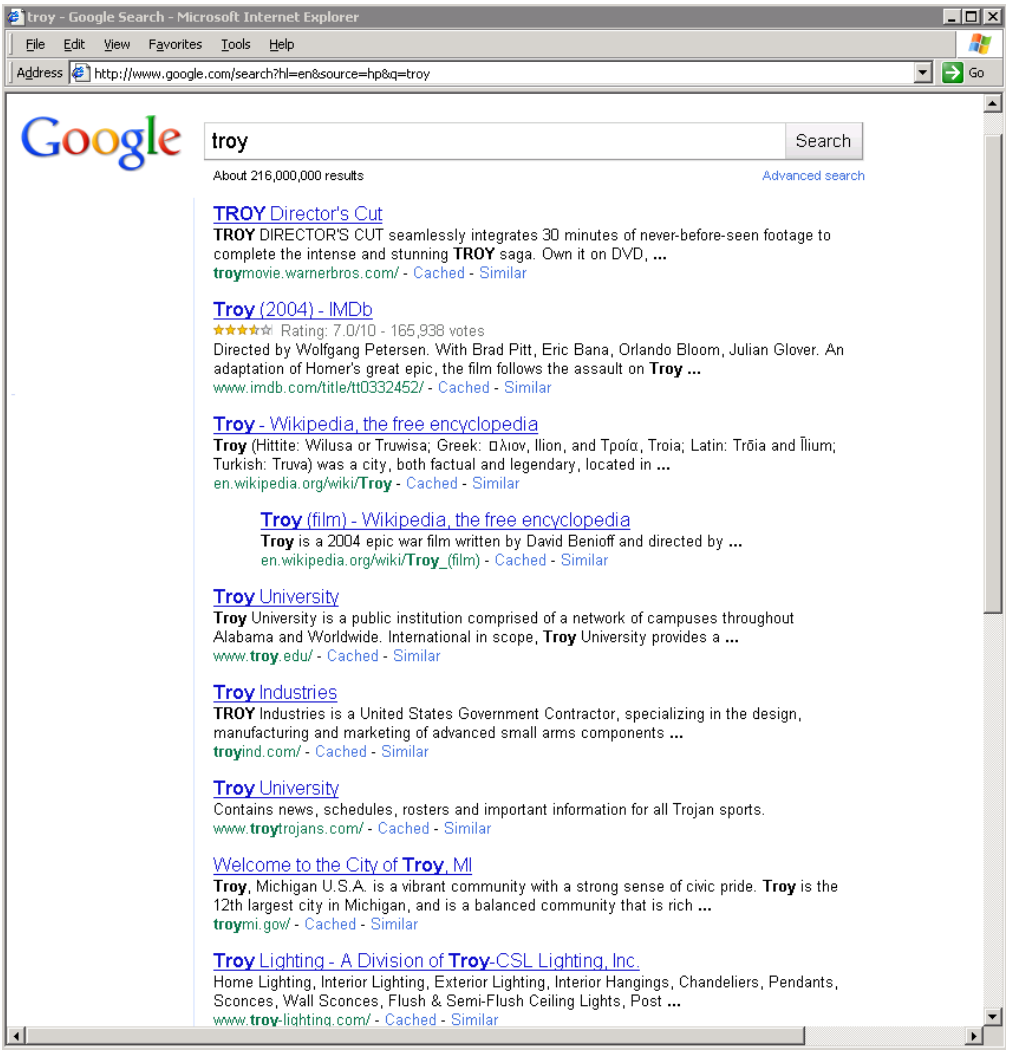


Figure 1 An example of web pages retrieved by Google using a query "troy"

The two most common effectiveness measures are recall and precision, introduced in the Cranfield's studies as mentioned in [11], to summarize and compare search results. Intuitively, recall measures how well the search engine is doing at finding all the relevant documents for a query, and

precision measures how well it is doing at rejecting non-relevant documents. Our focus is in improving the precision of the search provided by a search engine. The definition of these measures assumes that, for a given query, there is a set of documents that is retrieved and a set that is not retrieved (the rest of the documents). If, in addition, relevance is assumed to be binary, then we can define precision and recall as follows:

- $\text{Precision} = \frac{|X \cap Y|}{|Y|} = \frac{\text{Total number of retrieved and relevant documents}}{\text{Total number of retrieved documents}}$
- $\text{Recall} = \frac{|X \cap Y|}{|X|} = \frac{\text{Total number of retrieved and relevant documents}}{\text{Total number of relevant documents}}$

where X is the relevant set of documents determined by either human annotators or machines [3] and Y is the set of retrieved documents for the query. The benefit of manual evaluation is the accuracy with respect to user's expectation. However, labelling web pages with (ir)relevance is subjective and time-consuming. On the other hand, automatic evaluation is simply applied even if the web pages are altered. In addition, instead of human users, machines can deal with the large amount of web pages on the web [8]. In particular, note that most of automatic evaluations are based on the process of mining click-through data in query logs of search engines. In Section 2, we will discuss the details of existing methods for evaluating search engine results. However, most of automatic approaches to evaluate search engine results no longer work unless search engines provide well-defined query logs or users clicks-through data. In this work we propose an automatic evaluation without relying on search engines' query logs and users' click-through data.

In other words, recall is the proportion of relevant documents that are retrieved, and precision is the proportion of retrieved documents that are relevant. We can also view the search results as the output of a binary classifier [3], as we deal in this paper. When a document is retrieved, it is the same as making a prediction that the document is relevant. From this perspective, there are two types of errors that can be made in prediction (or retrieval). These errors are called false positives (an irrelevant document is retrieved) and false negatives (a relevant document is not retrieved). Recall is related to one type of error (the false negatives), but precision is not related directly to the other type of error. Instead, another measure known as fallout, which is the proportion of non-relevant documents that are retrieved, is related to the false positive errors. A natural question is why we use precision if fallout and recall[1] are enough to evaluate search results. The answer given in [3] is very simple, "precision is more meaningful to the user of a search engine" as precision has larger value as compared to fallout. Fallout will always have very small value due to large number of irrelevant documents provided by the search engine. For example precision of 0.7 means 70% of the retrieved documents are relevant.

1.2 Problem Definition

Our problem is: *Given a web page, the goal is to automatically determine if it is relevant with a target entity or not*. On the web we have large number of web pages in different languages. We need a scalable solution that is also independent of web page contents or independent of a particular

[1] Since the solution set to a particular domain is unknown, we cannot measure Recall values of our proposed models. On the other hand, Precision values can be measurable with the help of human experts. For example, the human experts can determine whether or not a retrieved web page is relevant with a target entity.

language. We present a supervised classification model called relevance classification model, built on a labelled training set (labelled by us). The training set used for building the relevance classifier consists of novel language independent features.

Table 1 Basic Terms/symbols and notations used throughout the paper

Term	Definitions and Examples
Page (p)	It's a web page retrieved by a search engine using query q. It's represented by lower case p. We use the word page and web page interchangeably in the paper according to the context and clarity.
Query (q)	Keywords used by a user (in an unstructured format). A query is formed from attribute values of an entity. For example **q_1:**<Yoyogi restaurant Asian> and **q_2:**<Yoyogi restaurant> are two queries. It's represented by small case q.
Template (t)	It's a query template t, formed from combination of attributes of an entity. For example t_1 and t_2 are two query templates related with queries q_1 and q_2 where **t1:**<Name Cuisine> and **t2:**<Name>. In our intuition we can extract or form queries and templates from a page relevant to an entity using the words present in the page. We use the words template and query template interchangeably in the paper. The term template extraction is simply template formation using the words in page.
Entity (e)	In our context an entity e, means a specific object e.g., a particular restaurant or a particular smartphone. We assume the same type of entities follows a common relational schema $R(a_1,a_2,...,a_n)$. Attributes $\{a_1,a_2,...,a_n\}$ are properties which describe the entity. The ID or key attributes of an entity are properties which can uniquely identify an entity. The ID of a restaurant is the restaurant's name if all the restaurants in a relation/table have different names otherwise we need other attribute(s) may be in combination with name. In real world we may encounter entities which don't have unique ID's. So searching such entities will be ambiguous, a real challenge for search engines. We have used nine attributes for the restaurant entity in our example, namely the attributes are *restaurant name*, *street*, *city*, *zip code*, *phone number*, *special dish*, *cuisine*, *category*, and *neighborhood*. In addition, in the smartphone domain, the attributes are *maker*, *Petname*, *os*, *lcd*, *ap*, *memory*, *lte*, *battery capacity*, and *weight*.
Domain	The word domain is actually a set of all related entities. More specifically the generic word Restaurants is a domain and a specific restaurant say Yoyogi restaurant is an entity. We collect entities from a common domain in a single relation or table. Each row will be an entity and columns of the table are descriptive attributes/properties of an entity. For our experiments, we made use of two different data sets – (1) restaurants in Singapore and (2) smartphones
Precision	Precision (also called positive predictive value) is the fraction of retrieved instances (say web pages) that are relevant. We can represent precision of a template, entity, query and a page by P(t), P(e), P(q) and P(p), respectively.
Precision Probability	We define precision in terms of probability and so it is called precision probability and we derive probabilistic inference equations/models that derive probability estimates for precision of each different type of construct say query, template, page and entity. Precision Probability of a template, entity, query and page is represented by PP(t), PP(e), PP(q), and PP(p), respectively.

Table 1 is about some notations and definitions used throughout the paper. In our consideration an entity e, is a specific object about which a web surfer wants to search through a search engine using some query q. As a result search engine (say Yahoo) provides ranked results in terms of ordered pages. Very often, retrieved web pages are very large in number and may be ranked incorrectly due to many reasons. In our context the term domain is actually a set of all related entities. More specifically the generic term restaurant is a domain and a specific restaurant say Yoyogi restaurant is an entity. Similar entities (e.g. all the restaurants) belong to a domain (Restaurants). Let's introduce four

important constructs namely queries, templates, web pages and entities. A query q is a user query and it is formed from the keywords used by a user (in an unstructured format). More concretely, in our experiment the query is formed from descriptive features or attribute values of an entity. For example q1: <Yoyogi restaurant Asian> and q2: <Yoyogi restaurant> are two queries. These queries q1 or q2 may be used by a user of a search engine to search for Yoyogi restaurant which is an Asian cuisine in Singapore. Whereas, a template t is a query template, formed from combination of attributes of an entity. For example t1: <Name Cuisine> and t2: <Name> are two query templates related with queries q1 and q2. In this example the attribute "Name" has associated value "Yoyogi restaurant" and the attribute "Cuisine" has associated value "Asian" for the entity Yoyogi restaurant.

In our intuition these four constructs (templates, queries, pages, and entities) are related to each other. We assume that the three constructs template, query, and entity are linked to each other through a relevant page. More specifically, a page p has a central place in the relationship, as shown in Figure 2. A page is related to an entity e.g. home page of "Yoyogi restaurant" and that page has descriptive words about it. In our intuition we can extract or form queries and templates from a page relevant to an entity using the words present in the page. We use the words template or query template interchangeably in the paper. The term template extraction is simply template formation using the words in the page. In short using a search engine say Yahoo we can retrieve that page using a query consisting of some keywords present in the page. Using the intuition of Figure 2, we construct a heterogeneous graph as shown in Figure 4, where a page plays a central role with its related constructs namely query, template and entity.

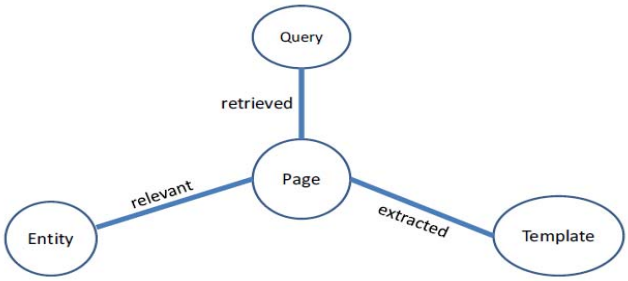


Figure 2 Four constructs - pages, queries, templates and entities are related with each other where a page plays central role in this relationship

1.3 Contribution

Our proposed methodology in simple words is described here in the following lines and Figure 3 about system architecture in Section 4 clarifies the description. More formal and step by step technical discussion about our methodology is discussed in Section 4 and experimental results are discussed in Section 5. Given a domain say Restaurants as shown in Table 2, we first select one entity at random, and then generate all possible queries by combining attribute values of the entity. For instance, suppose that an entity named "Yoyogi restaurant" from the restaurant domain is chosen at random from Table 2. By utilizing Table 3, we can generate different queries to the "Yoyogi restaurant" entity as <Yoyogi restaurant>, <33 mohamad sultan road>, <Asian>, <Yoyogi restaurant

33 mohamad sultan road>, <Yoyogi restaurant Asian>, <33 mohamad sultan road Asian> and <Yoyogi restaurant 33 mohamad sultan road Japanese>. And the corresponding query templates to these queries are <Name>, <Street>, <Cuisine>, <Name Street>, <Name Cuisine>, <Street Cuisine> and <Name Street Speciality> respectively. Then, using a search engine such as Yahoo, web pages are retrieved by using each query q related with entity e. From the search results, top-k pages are chosen related with each query of each entity. Now each query (e.g., <Yoyogi restaurant Asian>) has the corresponding template (e.g.,<Name Cuisine>) and top-k web pages. As we all know that these top-k web pages may contain irrelevant web pages about the "Yoyogi restaurant" entity. We evaluated the search results in person. Therefore we utilizes the idea that, let human experts who posted the query decide whether each of the top-k web pages is relevant or not. The entity (e.g., Yoyogi restaurant) and its relevant web pages are put to a label set. The label 1 means relevant and -1 means irrelevant page to a particular entity (i.e., a restaurant) and relevant query to the entity. Above process is repeated for a small subset of entities randomly chosen from a specific domain say Restaurants. Finally, we will have a labelled set of top-k pages related with each query in the chosen subset. From this label, we can form a heterogeneous graph consisting of four different types of nodes (see Figures 2 and 4) - Query (e.g., <Yoyogi restaurant Asian>), Template (e.g., <Name Cuisine>), Page (e.g., Top-k web pages retrieved from a search engine using the query <Yoyogi restaurant Asian>) and Entity (e.g., the "Yoyogi restaurant"). As an example, in Figure 4, q_1 and q_2 stands for queries and assume that q_1 and q_2 are <Yoyogi restaurant Asian> and <Yoyogi restaurant> respectively. Also t_1 and t_2 means the templates of q_1 and q_2 respectively. Specifically t_1 and t_2 are <Name Cuisine> and <Name>. The four web pages are p_1, p_2, p_3 and p_4. In the graph Figure 4, t_1 and q_1 are linked to three pages p_1, p_2 and p_3. This indicates that the three web pages p_1, $p_2$and p_3 are retrieved using the query q_1. And it's obvious that query q_1 is extracted from template t_1 by assigning attribute values to the template members, Name and Cuisine. Similarly, p_3 and p_4 are retrieved by q_2 and hence q_2 is connected to p_3 and p_4 in the graph. Now it's obvious that query q2 is associated with the template, t_2 (<Name>). Thus, t_2 is also linked to p_3 and p_4. Finally, using labels of human experts from the label set we can manually connect relevant pages to the entities in the graph. An entity e_1 is linked to p_2 and p_3 because those web pages are turned out to be relevant with e_1, while p_1 is irrelevant with e_1. Utilizing this heterogeneous graph, the precision probability value of each template node can be estimated based on a random walk model (by using Algorithm 1, Section 4.2.1), and then we can find top-k relevant pages to each top query template (using Algorithm 2, Section 4.2.2). Then we can find discriminative feature for our learning model. In Section 4.3, we discuss the details of the features proposed in this paper. Using Support Vector Machines (SVM), we learn a relevance model to automatically classify whether or not a web page p is relevant with an entity e in the testing set.

In this paper, our contributions are as follows:

- *To gain knowledge of entities on a domain, our approach automatically decides whether or not each web page is relevant based on a learning model.*
- *We define precision in terms of probability and propose precision probability models for different constructs involved in the search process namely entity, query, query template and web page.*
- *We devise language independent discriminative features. Based on these language independent features, our relevance classification model is able to classify non-English web pages as well.*

The remainder of the paper is organized as follows. In Section 2, we discuss existing search engine evaluation methods. Next, in Section 3 we discuss and derive our precision probability equations. Then we discuss relevance classification process in Section 4. Experimental results are discussed in Section 5. Finally, we conclude our work in Section 6.

2 Related Work

Evaluating search engines is one of key challenges in information retrieval. At present, most evaluation methods are based on the study of Cranfield's as mentioned in [11]. Croft et al., dealt with how to evaluate search engines and in their book [3], the evaluation of search engines is the "optimization" between effectiveness and efficiency. According to [3], effectiveness is a measure of how right information is found and efficiency is how quickly this is done. To optimize performance in terms of both effectiveness and efficiency, the best values for these parameters are determined using training data and a cost function. In particular, the cost function is a quantitative measure for the ranking algorithm that maximizes the effectiveness using the training data. In the book, the authors mainly discuss different "evaluation measures" for effectiveness - e.g., precision, mean average precision, interpolation, discounted cumulative gain, binary preference, and so on. Li et al. [8], showed three different types of relevance scoring methods such as vector space model, Okapi similarity measurement, and cover density ranking in order to automatically evaluate six major search engines' query results based on a large number of sample queries. Then, they proposed a three-level scoring method based on two steps. In the first step, given a query q with n terms and a web page x, A(q, x) is calculated by $\frac{t_n \cdot k^{n-1} + t_{n-1} \cdot k^{n-2} + \cdots + t_1}{k^{n-1}}$, where (1) k is the weight for longer sub phrases and (2) t_i is # of occurrence of the sub phrases of length i. In the second step, A(q, x) is converted to a three-level similarity score in terms of a certain threshold value θ. This is, (1) sim(q, x) = 2 if A(q, x) ≥ θ (2) sim(q, x) = 1 if θ> A(q, x) ≥ αθ (3) sim(q, x) = 0 if A(q, x) < αθ. In the equations, α is in the range [0, 1], representing partial relevance. Recent studies [4, 5, 7, 9] on automatic evaluation of search results are based on query logs of search engines and user click through data of past users. Dupret et al. [4], evaluated the performance of search engines based on the click through data of past users. Further, their basic idea is based on the probability that a web page is relevant to a set of queries. For this, they proposed a generative model to predict user clicks on document snippets based on user sessions recorded in query logs. Hosseini and Abolhassani [5] presented a query-URL co-clustering for a web site. For example, all queries and clicked URLs corresponding to a particular web site are collected from a query log. Then, the queries and URLs are formulated as a bipartite graph. Using a clustering method, the queries and URLs are clustered. Finally, using information entropy, the clusters of queries and URLs are used for evaluating link structure and information architecture. Liu et. al. [9], selected the topics and automatically annotated answers based on the analysis of users' query log and click-through data. In addition, Joachims [7] proposed a ranking algorithm for re-ranking search results using user clicks. Similarly, Zhuang and Cucerzan [18] developed Q-Rank to build query context from query logs and hence rank search results based on the query context. Taneva et. al. [14], proposed a image search framework in which the extended queries to an entity retrieve different candidate images from image search engines, and then better rankings of images are determined using weighted voting methods. Their voting approaches are similar to our features. However, they focused on the ranking problem and precision probabilities are not considered unlike our solution. In our methodology we redefine precision in terms of probabilities and call it precision

probabilities. The approach proposed in this paper to improve the precision of a search engine results is novel and new. We observed that our proposed precision probability equation (19) (discussed in Section 3) is similar to PageRank algorithm in Google [2]. The Google search engine has two important features [2] that help it produce high precision results. In [2] intuition of PageRank algorithm is also discussed. First, PageRank makes use of the link structure of the Web to calculate a quality ranking for each web page. This ranking is called PageRank. Second, Google utilizes link to improve search results. PageRank can be thought of as a model of user behaviour. The probability that the random surfer visits a page is its PageRank. And, the damping factor d (mostly used d = 0.85 in their experiments) is the probability at each page the "random surfer" will get bored and request another random page. The graph structure of the web used by PageRank consists of pages linked by hyperlinks. In our approach our graph is heterogeneous and bases on four related constructs pages, entities, queries and templates. But our precision probability equation (19) drawn from the graph is similar to PageRank and it is discussed in detail in the next Section 3.

3 Precision Probabilities Modelling

The section begins with introducing statistical preliminaries, useful for deriving our probabilistic inference modelling equations. Here we redefine the precision in probabilistic sense and define precision probabilities of different constructs namely pages, entities, queries and templates.

3.1 Graph Formulation and Statistical Preliminaries

As discussed before, we have four distinct types of constructs - pages, queries, templates and entities. In this section we have developed probabilistic sense of precision (we call this precision as precision probability) for all these four constructs based on their interrelationship. According to our intuition these constructs are related or linked to each other where a page plays central role in this relationship as shown in Figure 2.We can think of a graph like Figure 2 and suppose that these four constructs represented by P, Q, T and E are the nodes of the graph. In Figure 4 this graph is explained by a toy example. In this graph the page set P is set of all the web pages (relevant or not) retrieved by queries in Query set Q. E is set of entities (of a particular domain) relevant to pages in page set P, the queries (in Q) are all about. For example a query q in Q retrieves a page p in P, hence in this sense q is related or liked to p by the relation "retrieved." We say that a page p is retrieved from query q. Similarly since template t can be extracted from p (as discussed before), there is also relation between p and t namely "extracted." That is a template t which is extracted from a page p. More specifically for clarity purpose, suppose q is <Bombay grill Indian> and t is <Name Cuisine>. By seeing t and q we can say that here entity e is a restaurant and name of the restaurant is "bombay grill," this restaurant is of Indian cuisine category. It is obvious that p is retrieved by q, and both q and t occur in p. So we can say that due to presence of query words in page p the page is retrieved from query q using a search engine. We can form or extract template t from the page p, as a query q is combination of attribute values in the page and template t is combination of attribute names, the attribute values are associated with. In this sense a template is implicitly present in the page and we can extract it from the page contents. For example, if both query q_1 and template t_2 co-occurs in a page p_1, we can represent this as $p_1 = (q_1, t_2)$.

We can model the co-occurrence of a template t and a query q in a page p of page set P as

$$P = \{p \mid p = (q, t), q \in Q, t \in T, q \text{ and } t \text{ co-occurs in a domain } D\} \quad (1)$$

If we know all the relevant pages in P related with all the entities in E we can represent this relation in a set L where L = {(e, p)| p is related with e}. Here p is connected to e in terms of relevant relationship.

Let I_t denote the set of pages that any query q and a template t instantiates.

$$I_t = \{p \mid p \in P, p=(*, t)\} \quad (2)$$

I_q is a set of pages that a query q and any template instantiates.

$$I_q = \{p \mid p \in P, p=(q,*)\} \quad (3)$$

Further let P_R stands for a set of all pages relevant with any entity in the set of entities and we define it as;

$$P_R = \{p \mid p \text{ is a relevant page with entity } e_i \in E\} \quad (4)$$

Now let's generalize precision for templates in the probabilistic sense and we represent this by PP(t). In the information retrieval community, precision is defined as $P(t) = \frac{|I_t \cap P_R|}{|I_t|}$. To generalize the notion of such a precision to the precision probability of a query template t, let's rewrite it in a statistical way as

$$P(t) = \frac{|I_t \cap P_R|}{|I_t|} \quad (5)$$

In probabilistic sense we can rewrite precision of template t, P(t) as precision probability of t, PP(t);

$$PP(t) = \frac{Pr(p \in I_t \text{ and } p \in P_R)}{Pr(p \in I_t)} \quad (6)$$

$$= Pr(p \in P_R | p \in I_t) \quad (7)$$

In (5), since the numerator is the count of $I_t \cap P_R$, it represents the probability, when we draw a random page p, that $p \in I_t$ and $p \in P_R$ both hold, i.e., $Pr(p \in I_t \text{ and } p \in P_R)$. Similarly, the denominator represents probability $Pr(p \in I_t)$. By the definition of conditional probability $Pr(Y|X) = \frac{Pr(X \cap Y)}{Pr(X)}$, (6) is converted into (7). As a result, we obtain the probabilistic precision of t as the conditional probability of $p \in P_R$ given $p \in I_t$. This is the likelihood that p is relevant, given that it is instantiated by t. Since the precision probabilities of queries and entities are quite similar, we omit the derivations here.

For pages $p \in P$, we can also rewrite precision of page p, $P(p) = \frac{|P_R \cap I_p|}{|I_p|}$ as precision probability of p as, $PP(p) = Pr(x \in P_R | x \in I_p)$. Further, since I_p is simply {p}, set of pages itself, the condition $p \in I_p$ means x = p, which thus simplifies the expression to just $Pr(p \in P_R)$. In words, the precision of p

measures how likely p is relevant. We can thus generalise precision probability of a page, PP(p) as follows:

$$PP(p) = Pr(p \in P_R) \tag{8}$$

3.2 Probabilistic Inference Modelling

The probabilistic inference model derives probability estimation of a node (or a construct) in terms of its related nodes (or constructs), through their semantic relevance. Let's redefine precision probabilities of all the nodes (or constructs) in terms of our intuition see Figure 2. In the following lines PP(t) is precision probability of a template, PP(q) is precision probability of a query, PP(e) is precision probability of an entity and PP(p) is precision probability of a page. As these constructs are modelled by a graph we are using the words constructs or nodes interchangeable. So we can say that PP(t) is a precision probability of a template node in place of precision probability of a template, but these are similar in concept. Let's derive precision probability of a template node PP(t) as follows.

$$PP(t) \equiv Pr(p \in P_R | p \in I_t) \tag{9}$$

$$= \sum_{p_i \in P} Pr(p \in P_R, p = p_i | p \in I_t) \tag{10}$$

$$= \sum_{p_i \in I_t} Pr(p \in P_R, p = p_i | p \in I_t) \tag{11}$$

$$= \sum_{p_i \in I_t} Pr(p \in P_R | p = p_i, p \in I_t) \cdot Pr(p = p_i | p \in I_t) \tag{12}$$

$$= \sum_{p_i \in I_t} Pr(p \in P_R | p = p_i) \cdot Pr(p = p_i | p \in I_t) \tag{13}$$

$$= \sum_{p_i \in I_t} PP(t) \cdot \frac{|I_{p_i t}|}{|I_{p_i}|} \tag{14}$$

In (10), we expand it to the joint distributions with every $p_i \in P$. (11) restricts p_i to only those instantiated by t, or $p_i \in I_t$, which are the neighbors of t. By the Bayes' theorem, (12) rewrites using $Pr(xy|z) = Pr(x|yz)Pr(y|z)$. In (13), since $p_i \in I_t$ is conditionally independent of $p \in P_R$, given $p = p_i$, we can remove $p \in I_t$. As the first term equals $\frac{|I_{p_i t}|}{|I_p|}$ and the second term PP(t), (14) completes the rewriting. We can derive PP(t) and PP(e) quite similarly. In summary the precision probabilities are as below:

$$PP(t) = \sum_{p \in I_t} PP(p) \times \frac{|I_{pt}|}{|I_t|} \tag{15}$$

$$PP(q) = \sum_{p \in I_q} PP(p) \times \frac{|I_{pq}|}{|I_q|} \tag{16}$$

$$PP(e) = \sum_{p \in I_e} PP(p) \times \frac{|I_{pe}|}{|I_e|} \tag{17}$$

Since the neighbors of page p are queries, templates, and entities as in Figure 2, the precision probability equation can be formed by combining Eq. (15), (16) and (17).

$$PP(p) = \sum_{p \in I_t} PP(t) \times \frac{|I_{pt}|}{|I_p|} + \sum_{p \in I_q} PP(q) \times \frac{|I_{pq}|}{|I_p|} + \sum_{p \in I_e} PP(e) \times \frac{|I_{pe}|}{|I_p|} \tag{18}$$

These equations can also be represented as a matrix equation as follows:

$$PP^{(i+1)} = \alpha A PP^{(i)} + (1- \alpha)y \tag{19}$$

where i, A, and y denote the number of iterations, a transition matrix, and a personalized vector, where entity components have 1/|E| and 0 otherwise, respectively. In addition, $PP^{(0)}$ is an initial precision vector, where all entries have 1/|V| (where V stands for the number of nodes in the graph). Interestingly note that our precision equation is similar to Google PageRank[2] [2]. Due to this similarity for our experiments, we set 0.85 to α in our experiment (as in [2] they used 0.85 for damping factor d). This precision probabilistic equation (19) is utilized in Algorithms 1 and 2, proposed in this paper. We will further discuss this equation (19) in our experimental result section as well.

[2] In the end, our proposed precision probability equation can be induced to Eq. 19. And it seems to be similar to PageRank. The equation in Eq. 19 is based on a random walk model on a heterogeneous graph with four different types of nodes. PageRank is the same except that it works on a hyperlink graph with homogeneous nodes like web pages. Interestingly, in our method, the precision probability values of each node are similar in only odd iterations and those of each node are similar in only even iterations (See Table 5). This is because both query and (query) template nodes always exist on a heterogeneous graph. This result is different from that of PageRank.

4 Relevance Classification

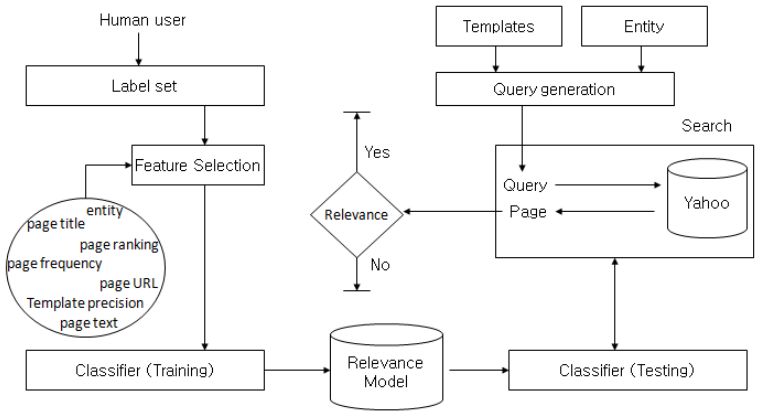


Figure 3 System Architecture

Figure 3 about system architecture describes our approach. Let's dive into the detail and summarize the steps. For a specific domain (say collection of restaurants or smartphones) we prepared and utilized data sets for different purposes and the preparation of these sets is discussed in detail in the next subsection called Data sets. We use one label set in forming the graph G_0, utilized in Algorithm 1. Other label set is utilized in page relevance classification and is discussed in Section 4.4. We have proposed sampling algorithm (Algorithm 1 in Section 4.2.1) which outputs top-k (in our experiment k=50) query templates (related to entities of a specific domain say restaurants). These templates are ranked by precision probabilities proposed in this paper and discussed above in Section 3. In Algorithm 1 we proposed a sampling approach to calculate the precision probabilities in an efficient way. Then we generate queries from each top query template provided by Algorithm 1 and collect top-50 pages ranked by Yahoo search engine. Next, we select top-40 pages in each entity using our precision voting algorithm (See Algorithm 2 in Section 4.2.2). Then we extract novel features (See Section 4.3) and build classifier in Section 4.4 based on the data set prepared from those features. Following subsections explains our approach in a step by step manner.

4.1 Preparation of Data sets

In this section we discuss about the data sets used in our experiments.

4.1.1 Collection of High Frequency Pages using Exhaustive Query Generation

In this subsection we discuss the process of collecting high frequency pages using Yahoo search engine by generating queries from exhaustive combination of attribute values of an entity. Let's select twenty restaurants from http://www.hungrygowhere.com. Table 2 shows the list of these chosen restaurants at random. These twenty restaurants are actually twenty entities belonging to the domain Restaurants. In Table 2 we can see different types of Cuisines. For example, "daidomon" is a Japanese Cuisine and "sushi jiro" is an Asian Cuisine. For Muslims, a halal Cuisine "Olive" is also in

the list. Furthermore, we manually collected information regarding twenty popular smartphones on the web.[3]

We describe each entity by nine attributes. As shown in Table 3, the attributes are restaurant's name, street, city, zip code, phone number, special dish, cuisine, category, and neighbourhood in the restaurant domain. On the other hand, those in the smartphone domain are maker, petname, os, lcd, ap, memory, lte, battery capacity, and weight. From the attribute values, we can generate all possible queries. Since we have nine attribute values, we can create 2^9-1=511 possible queries for a single entity. For example, based on Table 3, we can generate queries <Yoyogi restaurant>, <33 mohamad sultan road>, <Yoyogi restaurant 6887-4669>, <Yoyogi restaurant Asian river valley>, etc. We can also have the query templates corresponding to the queries, e.g., <Name>, <Street>, <Name phone>, <Name Cuisine Neighbourhood>, etc. For each restaurant entity in Table 3, top-50 web pages per query are downloaded using Yahoo Search Engine. Next, unique web pages are collected in a table, and then sorted by the frequency of web pages. Let's explain the process by an example. Suppose two queries q1 and q2 and three web pages p1, p2 and p3. Further assume that using Yahoo Search Engine pages p1 and p2 are retrieved by query q1, while p2 and p3 are retrieved by q2. In this case, the frequencies of p1, p2 and p3 are 1, 2, and 1, respectively. After the frequencies of web pages are counted, top-40 web pages with the highest frequencies per restaurant (amongst the 20 chosen restaurants) are selected. Finally, we have obtained 800 high frequency web pages of the twenty entities from the restaurant domain.

Table 2 Twenty entities from two domains

Restaurant Domain			Smartphone Domain		
ID	Name	Cuisine	ID	Name	Maker
1	Highlander coffee bar	WESTERN	1	Iphone3g	APPLE
2	Daidomon	JAPANESE	2	Iphone4 16g	APPLE
3	Shri anandhem restaurant	ASIAN	3	Venue	DELL
4	Sushi jiro	ASIAN	4	Take2	KTTECH
5	Swensen's cafe restaurant junction 8	WESTERN	5	Love shake	KTTECH
6	The arena	WESTERN	6	Desire hd	HTC
7	Jia thai thomson plaza	THAI	7	VEGA X	PANTECH
8	Yoyogi restaurant	ASIAN	8	Vega no.5	PANTECH
9	Tandoori restaurant	ASIAN	9	Take janus	KTTECH
10	Aburiyatei	ASIAN	10	Optimus g	LG
11	Food haven the restaurant	CHINESE	11	Lollipop	LG
12	Big bites	INDIAN	12	Wine	LG
13	Sawasdee thai food	THAI	13	Prada 3.0	LG
14	Tasty thai hut	ASIAN	14	Atrix	MOTOROLA

[3] For this, some experts in LG Electronics helped to collect the smartphone data set used in our experiment.

15	Ras the essence of india	INDIAN	15	Nexus one	HTC
16	Tatsuya	JAPANESE	16	Galaxy note	SAMSUNG
17	Phin's steakhouse liang seah	WESTERN	17	Galaxy s3	SAMSUNG
18	Olive	HALAL	18	Galaxy note II	SAMSUNG
19	Andhra curry	ASIAN	19	Nori	SAMSUNG
20	Sakura international buffet restaurant	JAPANESE	20	Armani	SAMSUNG

We have prepared such kind of high frequency table twice one with twenty entities as discussed above and other with 100 entities using the same process and same URL http://www.hungrygowhere.com. In summary we will have 800 and 4000 high frequency web pages for twenty and hundred entities respectively. Note that entities are chosen at random from the http://www.hungrygowhere.com. These two sets of entities are from the same domain "Restaurants" and disjoint with each other. The table of 800 high frequency web pages related with 20 entities is used in Section 5. And the table of 4000 high frequency web pages related with 100 entities is used in Algorithm 1 and also in Algorithm 2 (how? It is discussed in the subsections of Algorithm 1 and 2 in Sections 4.2.1 and 4.2.2.). In the same way, we have collected 800 high frequency web pages of the twenty entities from the smartphone domain.

Table 3 Examples of entities described with nine attributes

Restaurant Domain		Smartphone Domain	
Attribute	Attribute Value	Attribute	Attribute Value
Name	Yoyogi restaurant	Maker	Apple
Street	33 mohamad sultan road	Petname	IPHONE3G
City	Singapore	OS	iOS
Zip code		LCD	3.5 320 480
Phone	6887-4669	AP	S5PC100 600MHz
Specialty	Japanese	Memory	16G
Cuisine	Asian	LTE	No
Category	Restaurant	BatteryCapacity	1220
Neighbourhood	River valley	Weight	135

4.1.2 Preparation of Label Set, Training Set and Test Sets

We need a labelled data set to train and test our page relevance classification model. We have used Support Vector Machines [13] as a binary relevance classification model. Figure 3 illustrates the

system architecture of our model. Given a domain of interest say Restaurants, we randomly select a set of restaurants (say 20 entities) and their relevant web pages (say 800 pages) using the process discussed above in Section 4.1.1. We form a table called Label set with the meta information of entity name, applied query and relevant template, page URL, page title, page text contents (after removing high frequency stop words) and a label column having 1 or -1, indicating that page is relevant to the entity or not. Page relevance is judged by human annotators[4]. From the label set we have extracted six features (discussed in Section 4.3) and prepared a feature set consisting of all these six features. With the combination of label set and feature set, we have generated a labelled training set of features having information about each entity in terms of six features (we proposed) with the relevance label (1 or -1) whether or not a page is relevant to an entity. Then using this training set we train and test the relevance classifier using Support Vector Machine model, and this process is discussed in Section 5. Given a testing set consisting of a set of entities we will repeat the same process to build the table of high frequency web pages for each entity as discussed in Section 4.1.1. This is, we will generate top-k queries which correspond to top-k query templates with the highest precision probabilities discussed in Section 3. Then, top-k web pages are retrieved using a common search engine (e.g., Yahoo) with a query. In the end our trained relevance classification model will decide whether each retrieved web page is relevant with a desired entity or not (See Figure 3).

4.2 Proposed Algorithm

In this paper we have proposed two algorithms that are discussed in this section.

4.2.1 Algorithm 1: Sampling Algorithm

In Section 3 we define probability inference models to measure precision probabilities of nodes from a graph similar to Figure 4 based on the intuition of Figure 2. However, in our models, computing precision probabilities is expensive if the number of entities is very large. Thus efficient inference has to be performed. To achieve this we have proposed a sampling algorithm based on probability inference models discussed in Section 3.

Algorithm 1: Sampling

Input: G_0 and Q_s
Output: Top-k templates and precision probabilities
$G = G_0$;
$e = e_0$;
$Q = Q_s$;
for $(i = 0; i < n; i++)$ **do**
  for $q \in Q$ **do**
    Get the first search result page from a search engine using q;
    Extract top-10 URLs from the first search result page;
    P: download the web pages of the top-10 URLs;
    for $p \in P$ **do**
      if *Jaccard similarity(p, e)* $> \theta$ **then**
        p is relevant with e;
    Add edges (i.e., edge(q,p), edge(t,p) or edge(e,p)) to G;
  Compute precision probabilities from G using random walks;
  Select top-k templates with the highest precision probabilities;
  Pick a next entity e_i at random from the sample;
  $e = e_i$;
  Q: generate queries from the top-k templates;

[4] This is a labour-intensive task which should be done for different types of entities.

As mentioned before we prepared a table of 4000 high frequency web pages related with hundred restaurant entities. These hundred restaurants are chosen at random from the http://www.hungrygowhere.com. Given a set of entities, we first create an initial graph G_0 from a subset of entities. The intuition behind creating the initial graph is to avoid from the generation 2^m-1 queries initially for each entity where m is number of attributes in our experiment (we use nine attributes to describe an entities say a restaurant as discussed in Section 4.1.1). To create the graph G_0, we select a small set of entities (say twenty entities and we call it a sample) at random from the entity set (or domain, this is the same set from which 100 entities are chosen). Then, manually find relevant pages to each entity (e) and label them as 1 or -1 (indicating relevant page to entity or not) and form "s" number of queries and collect in set Q_s. The queries are formed manually from the pages using keyword matching approach. Finally, we create G_0 (similar to Figure 4) with corresponding queries, templates, pages, and entities using label set based on the intuition discussed in Figure 2. In Algorithm 1, starting with G_0 we extend it automatically using Jaccard similarity between a page and an entity, then we compute precision probabilities of nodes using random walks. Through manual inspection, we find that θ = 0.1 is the best threshold value for Jaccard similarity between p and e. If page's similarity (using Jaccard similarity) with entity is greater than 0.1, then the page p is linked to the entity e in the graph. We measure precision probabilities of nodes in the graph by means of Eq. (19). Finally, we prune out nodes with lower precision probability nodes, and then conduct the same process for a next entity in the sample. Finally, Algorithm 1 will output top-k important query templates and their precision probabilities. For our experiment we use k=50.

In summary two inputs of algorithm 1 are a manually prepared graph G_0, similar to Figure 4 (discussed above) and manually prepared set of selected queries Q_s. The set Q_s is Algorithm 1, returns top-k (in our experiment k is 50) query templates with their associated precision probabilities. The ranking (top-k) is based on precision probabilities of template nodes. To expand the initial graph G_0 used in Algorithm 1, we use hundred entities and their corresponding high frequency table of 4000 pages, already discussed before.

Algorithm 2: Precision probability voting algorithm

Input: f_s: A search function and t: A query template
Output: top-40 pages
q: A query corresponding to t;
$P(t)$: Precision probability of t by Eq. 19;
$f_s(q) \rightarrow p$ //The search engine retrieves a web page p by q;
$w(p) = P(t)$, where $t \rightarrow q$ and $f_s(q) \rightarrow p$;
Compute score(p,e)=$\sum_{i=1..|P_e|} w(p)$;
Rank pages by score(p,e);
Select top-40 pages;

4.2.2 Algorithm 2: Precision Probability Voting Algorithm

The output of Algorithm 1 i.e. top-k templates (where k=50 in our experiment) along with their precision probabilities is used by Algorithm 2 as an input to calculate top 40 relevant pages associated

with an entity e. In Algorithm 2, for a top template, we can generate or form its associated query q by assigning values to attribute(s) in the template. And use this query in Yahoo search engine to find top-50 web pages ranked by Yahoo. Then we accumulate precision probability score, where page p is retrieved from Yahoo search engine using query q associated with a top template and e is the associated entity with the query.

4.3 Feature Selection

We use public search API of Yahoo [17] to download web pages. We extract only text data of each web page, and then stop words are removed in terms of the list of stop words listed in [6]. We have proposed six discriminative features F1, F2, F3, F4, F5 and F6 using the data of 800 web pages relevant to twenty restaurant entities (and 800 web pages relevant with twenty smartphone entities), as discussed in Section 4.1.1. The five novel features F1...F5 are web page contents independent features and hence are called non-content features. F6 is web page content dependent feature. Similarity in features F1, F2 and F6 is measured by Jaccard similarity measure using open source tool SecondString [12], where all space-delimited words from text strings are treated as tokens. Let's define our proposed features as follows:

- *F1: Similarity score between page title and entity name*
- *F2: Similarity score between URL terms and entity name*
- *F3: Rank voting score measured by* $\sum_{t=1..m} P(t)\frac{k+1-r_t(p)}{k}$*, where t: a template, PP(t): precision probability of template t, k: top-k pages, and* $r_t(p)$*: rank of page p retrieved from the Yahoo search engine*
- *F4: Precision voting score in terms of Algorithm 2*
- *F5: Normalized page frequency. This is same as the precision voting Algorithm 2 except that weight w(p)=1*
- *F6: Similarity score between page content and entity attribute values*

In particular, note that we can categorize these features to three groups. The first group contains F1 and F2 which are collected by additional information i.e. web page title and URL. Calculations of F1 and F2 are similar to the approach used in [16] and in calculating F3 our intuition is based on the approach of [14]. In the second group, features F3, F4, and F5 are generated based on precision probability and precision probability voting algorithm. The last group includes feature F6 and is a web page contents dependent feature.

4.4 Relevance Classification Model

Our goal in the paper is that we want to classify a web page (retrieved using Yahoo search engine) whether or not it is relevant to an entity. Note that in our intuition (and its quite natural) a query is formed by using attribute values of an entity, about which a user wants to search through a search engine. We use Yahoo search engine in our experiments. In our empirical study in Section 5.2.3, we build two SVM based relevance classification models by using all the six features F1, ..., F6 and by using only the non-content features F1, ..., F5. Classification results in Table 7 show that classifier built on all the six features (including web page content feature F6) is only better than 1.24% on

average accuracy as compared to the classifier built using only non-content features F1, ..., F5. This observation makes our non-content features novel and also makes our proposed relevance classification model language-independent.

5 Experimental Validations

The section is about performing the experiments using the proposed precision probability models, prepared data sets, proposed Algorithms 1 and 2, and the relevance classification based on novel features.

5.1 Set-up

Let's summarise our experimental setup by mentioning the data set, implementation details and the research questions in which we are interested.

5.1.1 Data set

In our experiment we use the table of 800 high frequency pages related to twenty restaurants and those related to twenty smartphones gathered on the web, discussed and prepared in Section 4.1.1. In Table 3 we have examples of both restaurant and smartphone entity described with nine attributes and their particular values. To experiment our classification problem, we randomly selected twenty restaurants in http://www.hungrygowhere.com and twenty smartphones among 127 ones that we manually gathered on the web. Table [3] shows the list of the chosen entities.

5.1.2 Implementation

All the experimentations were performed on 4x2.6 GHz Opteron processors with 32GB of RAM. We implemented Algorithm 1 and Algorithm 2 using Java programming language. For each query, web pages were downloaded using public search API of Yahoo [17]. Through HTML parsers, only text data of web pages were extracted, and then stop words were removed in terms of the list of stop words in [6]. To compute the similarity score between the text content of a web page and attribute values of a restaurant (or a smartphone), we used the open source tool, SecondString [12], where all space-delimited words from text strings are treated as tokens. In our manual inspection, we also decided the best threshold value for Jaccard similarity ($\theta = 0.1$ for our experiments). In addition, to compute the precision probabilities of query templates, we implemented the data structure and source codes for our precision probability equations based on Java template codes of [1]. The precision probability computation is stopped when $\eta = 0.0000001$ (η means the difference between previous precision probability vector and current one). For our classification experiment, we used Support Vector Machines (SVM) [13].

5.1.3 Summary of Set-up

Based on this set-up, we seek to answer the following questions:

- *Q1: In Section 3, we proposed precision probability equations. To show the correctness of the equations, we will analyze the outcome of precision probabilities using a toy example (Figure 4) in Section 5.2.1.*

- *Q2: What are top-10 templates that are discovered by Algorithm 2? We will empirically study the top-10 templates with the highest precision probabilities in Section 5.2.2.*
- *Q3: To figure out how many web pages are actually relevant, we will survey if each web page is relevant with a target entity or not. Then, we measure the average precision score at real top-k relevant web pages. Finally, we will see the reason why some web pages are irrelevant in Section 5.2.3.*
- *Q4: To evaluate our classification model, we will deal with the results of SVM based on different types of features, and then find the most discriminative feature in our model. For the details, see Section 5.2.4.*

5.2 Results

5.2.1 Correctness of Precision Probability

To verify our probabilistic inference models, we show the results of our model with a toy example. Figure 4 depicts a simple graph, where q_1 (<Yoyogi restaurant Asian>) and q_2 (<Yoyogi restaurant>) are queries; t_1 (<Name Cuisine>) and t_2 (<Name>) are templates; p_1, ..., p_4 are web pages; and e_1 is an entity "Yoyogi restaurant." In this graph, nodes are linked one another because of three different types of relationships, shown in Figure 2.

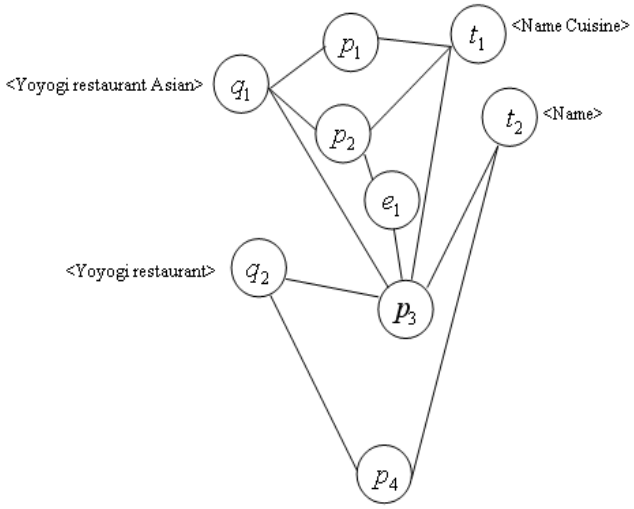


Figure 4 A toy example graph

Table 4 The transition matrix (A) of the graph in Figure 4

	t_1	t_2	q_1	q_2	p_1	p_2	p_3	p_4	e_1
t_1	0	0	0	0	1/3	1/3	1/3	0	0
t_2	0	0	0	0	0	0	1/2	1/2	0

q_1	0	0	0	0	1/3	1/3	1/3	0	0
q_2	0	0	0	0	0	0	1/2	1/2	0
p_1	1/2	0	1/2	0	0	0	0	0	0
p_2	1/3	0	1/3	0	0	0	0	0	1/3
p_3	1/5	1/5	1/5	1/5	0	0	0	0	1/5
p_4	0	1/2	0	1/2	0	0	0	0	0
e_1	0	0	0	0	0	1/2	1/2	0	0

Table 5 Precision probabilities of nodes of the graph in Figure 4

	i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7
t_1	0	0	0.1284	0.0193	0.1149	0.0336	0.1003	0.0436
t_2	0	0	0.0723	0.0108	0.0799	0.0212	0.0769	0.0296
q_1	0	0	0.1284	0.0193	0.1149	0.0336	0.1003	0.0436
q_2	0	0	0.0723	0.0108	0.0799	0.0212	0.0769	0.0296
p_1	0	0	0	0.1092	0.0164	0.0977	0.0286	0.0853
p_2	0	0.2833	0.0425	0.1699	0.0616	0.1433	0.0739	0.1302
p_3	0	0.17	0.0255	0.1265	0.0406	0.1131	0.515	0.1043
p_4	0	0	0	0.0614	0.0092	0.0679	0.018	0.0654
e_1	1	0.15	0.3427	0.1789	0.276	0.1935	0.259	0.2033

First, a search engine retrieves p_1, p_2 and p_3 using q_1. Second, t_1 can be extracted from p_1, p_2, and p_3. Finally, both p_2 and p_3 are determined to be relevant with e_1 if $sim(p_2, e_1) \geq \theta$ and $sim(p_3, e_1) \geq \theta$. Given the graph, we expect that t_1 should be more highly ranked than t_2. This is because t_1 is connected to more relevant web pages, comparing to t_2 in the graph. For each node (e.g., t_1), our proposed model will compute the precision probability of t_1 with which random surfers start at t_1, visiting neighbour nodes, and finally arrives at e_1 in the graph. As we already from Eq. (19) that our precision probability equation is $PP^{(i+1)} = \alpha APP^{(i)} + (1- \alpha)y$, where i is the number of iterations and α=0.85 in our experiment. As mentioned before, since our precision probability equation is similar to PageRank of Google [2], we use α=0.85 as used in [2] for damping factor. A is the transition matrix of the graph in Table [4]. And "y" is a personalized vector using which random surfers jump to other entity nodes if a node at which the surfers visit is a dangling node. In the y vector, every entity component has 1/|E|, where |E| is the number of entities, otherwise it is 0. In addition, $PP^{(0)}$ is an initial precision vector, where all entries have 1/(# of nodes). Table 5 shows the precision probability of each node in every iteration. The iteration is stopped when η =.0000001. In Table 5, we can clearly take a look at the fact that t_1 has higher precision probability than that of t_2, as expected.

5.2.2 Top-10 Templates

Table 6 Top-10 query templates in terms of our precision probability inference model.

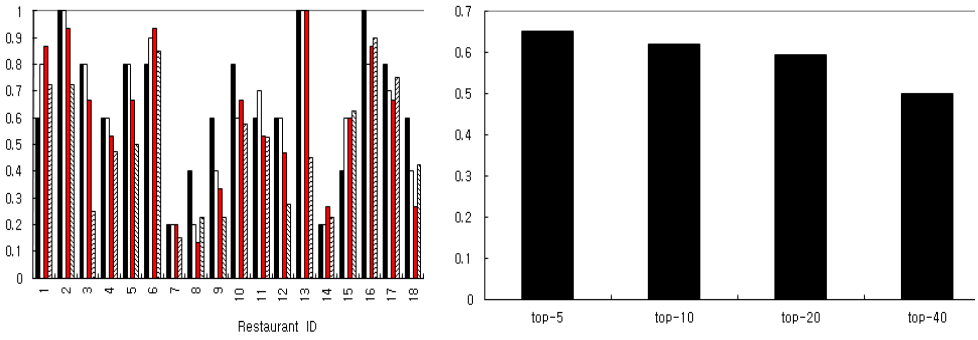
Restaurant Domain			Smartphone Domain		
Rank	Template	Precision Probability	Rank	Template	Precision Probability
1	<Name Phone Specialty Cuisine>	0.0054	1	<Maker>	0.1595
2	<Name Phone Cuisine>	0.0054	2	<Maker OS>	0.11
3	<Name City>	0.0048	3	<Maker Petname>	0.0443
4	<Name>	0.0048	4	<Maker Petname OS>	0.0101
5	<Name Specialty>	0.0048	5	<Maker Petname OS LTE>	0.0083
6	<Name City Specialty>	0.0048	6	<Maker Petname LTE>	0.0073
7	<Name Phone>	0.0044	7	<Maker LCD BatteryCapacity>	0.0042
8	<Name Phone Specialty>	0.0044	8	<Maker Petname OS Weight>	0.0032
9	<Name City Cuisine>	0.0037	9	<Maker Petname OS LTE Weight>	0.003
10	<Name City Specialty Cuisine>	0.0036	10	<Maker Petname LTE Weight>	0.0026

As shown in Table 6, we obtained 10 query templates with the highest precision probabilities based on Algorithm 1. At first, we noticed the top-1 query template <Name Phone Specialty Cuisine>. This top query template consists of multiple attributes rather than single attribute. This is why search engines tend to retrieve a few web pages containing multiple attributes at the same time by precision definition. If a web page includes multiple attribute values, it will be likely to be a relevant page. On the other hand, the <Name> template is also ranked highly in the top-10 template list. In general, it has been known that using only name query will cause false positive examples commonly. That is, if there are two restaurants such as highlander coffee bar at both Bishan and Stamford Road, web pages related to the two entities will be retrieved by the name query. However, according to our observation, "Name" is still a good indicator for deciding whether or not a given web page is relevant with a desired entity. For instance, in Table 2, if terms "highlander", "coffee", and "bar" frequently appear within a web page, we can strongly conjecture that the web page may be relevant with the highlander coffee bar entity. Thus we conclude that the Name attribute describes entities better than the other attributes. At least, as a single-attribute query, the Name attribute is better than Cuisine and Neighbourhood. Independently, we also performed a user study to see which queries retrieve more relevant web pages. Then, we observed that <Name>, <Name Phone>, <Name City> tend to show good search results. Such templates are highly ranked in Table 6 as well. In the smartphone domain, <Maker>, <Maker OS>, and <Maker Petname> are highly ranked among 512 query templates. Compared to many restaurants in Singapore, there are a relatively small number of smartphones around the world and major ones may be made by a few global IT companies like Apple, Samsung, Motorola, etc. If we use keywords such as <Samsung>, <Galaxy S5>, or <Samsung Galaxy S5>, we will be able to get relevant web pages with an entity Galaxy S5 at most. Each maker (either Samsung or Apple) is likely to have different Operating Systems and Petname (e.g., Apple-iPhone-iOS vs. Samsung-Galaxy-Android). On the other hand, both Apple and Samsung tend to have similar values

of LCD, weight, LTE, memory size, etc. Thus maker, os, and Petname are better than the others to obtain relevant web pages and Table 6 supports that maker, os, and Petname are more highly ranked than lcd, lte, weight, and so on. In particular, note there are not top-10 queries including Battery Capacity, AP, and Memory.

According to the results in Table 6, the "precision probability" values will be the best way to predict good query templates for different domains.

5.2.3 Empirical Study on Relevant Web Pages



(a) Precisions of each individual entity (b) Average precisions of all entities

Figure 5 Precisions of top-k web pages

Prior to applying our learning model, we first survey the relevance of eight hundred web pages of the twenty entities in Table 2. We manually decided that each web page is relevant or not. Figure 5(a) and (b) show average precisions of individual entity and of all entities, respectively. In Figure 5(a), the horizontal axis indicates entity ID and the vertical axis means average precision values measured by $\frac{|\text{relevant pages within top-k pages}|}{|\text{top-k pages}|}$. In the figure, regardless of top-k values, the average precision of each individual entity is totally different. The "daidomon" Japanese restaurant (ID = 2 in Table 2) has high precision values whereas precisions of the "jia thai" restaurant (ID = 7 in Table 2) at top-k are poor. This is because of the different reasons for each entity. By and large, we found six main reasons. First, some web pages were not found even if the pages had been indexed by Yahoo Search Engine in previous time. Second, there exist different restaurants with same name spellings on the web. For example, there are at least two swensen restaurants at different locations. Third, the main topic of some web page is not a target entity but other entity. For instance, a blog web page reviews a "yumeya" Japanese restaurant, comparing to the special dishes of the "yoyogi restaurant." In this case, the web page includes a little information of "yoyogi restaurant" but it mainly talked about the "yumeya" restaurant. Next, some entities are ambiguous e.g., "Sawasdee" is a person name as well. Fifth, attribute values of some entities are not accurate. For example, in Table 2, "olive" should be replaced by "fig and olive." Finally, in case of some small restaurants which are not popular (e.g., "jia Thai"), there are only a few web pages on the web. This is because of a little information on the web.

Overall, as shown in Figure 5 (b), the average precision of all entities is 0.65 at top-5 and 0.5 at top-40 pages. In summary, because of ambiguity and less information about entities on the web, the average precisions of manually finding relevant web pages are in between 0.5 and 0.65.

5.2.4 Classification

Through the process discussed in Section 4.1, we have eight hundred labelled pages. Using the label set, we deal with a binary classification problem. We assign 1 to one class if a web page p is relevant (52% in the label set), and -1 otherwise (48% in the label set). For evaluating our learning model, we divided the label set to four runs, each of which has six hundred web pages in a train set and two hundred web pages in a test set (This is Cross-Validation for avoiding overfitting as to our SVM model). Then, we applied the test sets to SVM as a classifier. Table 7 shows the results of SVM based on different feature sets that we propose.

Table 7 Page relevance classification results using SVM

Domain	Features	Run 1	Run 2	Run 3	Run 4	Average
Restaurant	Using All Six features F1-F6	0.6185	0.635	0.605	0.6	0.6146
	F1-F5 (using only five non-contents features)	0.5838	0.645	0.625	0.555	0.6022
Smartphone	Using All Six features F1-F6	0.805	0.785	0.69	0.69	0.7425
	F1-F5 (using only five non-contents features)	0.805	0.785	0.76	0.63	0.745

Table 8 Ranking of non-content features

Restaurant Domain				Smartphone Domain			
Rank	ID	Feature	Weight	Rank	ID	Feature	Weight
1	F5	Page frequency	4.1768	1	F5	Page frequency	5.2174
2	F1	Similarity between page title and entity	3.397	2	F1	Similarity between page title and entity	3.6085
3	F4	Precision voting	3.1248	3	F2	Similarity between URL and entity	2.00004
4	F3	Rank voting	2.1348	4	F4	Precision voting	0.957
5	F2	Similarity between URL and entity	1.7187	5	F3	Rank voting	0

In our experimental result, we note that our learning model improves about 10% in the restaurant domain and about 30% in the smartphone domain over random prediction. Since the maximum average precision is about 0.65 in our survey (see Section 5.2.3), the average precision of SVM (i.e., 0.62) is almost close to the upper bound in the restaurant domain. In addition, we observed there is no significant gap between content feature and non-content features in both domains. In other words, our proposed classification model is not associated with text content features of web pages. As a result, our model will be able to work in the classification problem with non-English web pages. In Table 8, we can also see that F5[5] is the most discriminative feature among non-content features.

In summary, our experimental results show that our approach can be applied to different domains with high accuracy. Although the "restaurant" domain is considerably different from the

[5] Pages are retrieved by top-10 queries in Table 6.

"smartphone" domain, our classification model shows coherent results which are high accuracy, compared to the result of binary classification at random (0.5). Please note that the average accuracies of the restaurant and smartphone domains are about 0.62 and 0.74, respectively, and these results are higher than 0.5. As a result, our classification model will be likely to work effectively regardless of particular domains.

6 Concluding Remarks and Future Work

In this paper, we focus on the problem of automatically deciding whether or not each web page retrieved by Yahoo search engine is relevant with an entity. To tackle this problem, we propose a novel approach for web page relevance classification, and then validate that our classification model will be likely to work effectively with different domains with high classification accuracy. We trained relevance classification model using novel features (proposed in this paper) related to an entity. Five of the features are web page content independent features and one is page content dependent feature. We build two SVM based relevance classification models by using all the six features and by using only the five non-content features. Classification results in Table 7 shows that classifier built on all the six features (including web page content feature) is only better than 1.24% on average accuracy as compared to the classifier built using only non-content features in the restaurant domain. On the other hand, the classifier with non-content features is slightly better than the classifier with all features including the content feature. This observation makes our non-content features novel and makes our proposed relevance classification model language independent. This observation suggests that our scheme can be applied on large collection of web pages as it doesn't use web contents at all. Our sampling algorithm for selecting top-k query templates makes our method efficient and scalable as we are utilizing a smaller annotated subset to make initial the graph and then expand the graph and connect other nodes automatically as discussed in sampling algorithm. The graph structure of the web in PageRank used by Google, consists of pages linked by hyperlinks. PageRank can be thought of as a model of user behaviour. In our approach, a graph is also a user model based on search engines. So our proposed graph is heterogeneous and based on four related constructs pages, entities, queries and templates. But our precision probability equation (19) drawn from the graph is similar to the PageRank.

In our future work we want to increase the size of data set for a particular domain and also want to increase number of domains to test our claims on large scale data. We also like to check the impact on precision accuracy of SVM by incorporating name entity disambiguation in preparing data sets, queries and query templates. We also want to compare the accuracy of our efficient sampling approach in terms of selection of top-k pages classification accuracy with the non-sampling approach. Since the non-sampling approach is time consuming and to test it we can utilize parallel computing using Hadoop or GPGPU computing on large scale data set. Another future direction is to investigate the interestingness of a web page. Given a "relevant" web page, the objective is to compute the ranking score of the page based on "interestingness" which can be estimated based on the parameters of informative, interesting remarks, nice images, rating scores, non-duplicate information, and so on.

Acknowledgement

This paper was supported by research funds of Kunsan National University for the first author and by the 2013 Yeungnam University Research Grant for the third author.

References

1. R. Sedgewick and K. Wayne (2007), Introduction to programming in Java: An interdisciplinary approach, Addison-Wesley (New York).

2. S. Brin and L. Page (1998), The anatomy of a large-scale hypertextual web search engine, Computer Networks and ISDN Systems, Vol.30, pp. 107-117.

3. B. Croft, D. Metzler, and T. Strohman (2009), Search engines: Information retrieval in practice, Pearson Education.

4. G. Dupret, V. Murdock, and B. Piwowarski (2007), Web search engine evaluation using click through data and a user model, Proceedings of the World Wide Web Conference.

5. M. Hosseini and H. Abolhassani (2007), Mining search engine query log for evaluating content and structure of a web site, Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence.

6. S. Howard, H. Tang, M. Berry, and D. Martin (2009), GTP: General text parser, http://www.cs.utk.edu/ lsi/.

7. T. Joachims (2002), Optimizing search engines using clickthrough data, Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

8. L. Li, Y. Shang, and W. Zhang (2010), Relevance evaluation of search engines' query results, Proceedings of the World Wide Web Conference.

9. Y. Liu, Y. Fu, M. Zhang, S. Ma, and L. Ru (2007), Automatic search engine performance evaluation with click-through data analysis, Proceedings of the World Wide Web Conference.

10. L. Lovasz (1993), Random walks on graphs: A survey, Combinatorics, Vol.2, pp. 1-46.

11. T. Saracevic (1995), Evaluation of evaluation in information retrieval, Proceedings of ACM Special Interest Group on Information Retrieval.

12. W. Cohen, P. Ravikumar, S. Fienberg, and K. Rivard (2003), SecondString: An open-source Java based package of approximate string-matching techniques, http://secondstring.sourceforge.net/.

13. T. Joachims (2008), SVM-Light Support Vector Machines, http://svmlight.joachims.org/.

14. B. Taneva, M. Kacimi, and G. Weikum (2010), Gathering and ranking photos of named entities with high precision, high recall, and diversity, Proceedings of ACM International Conference on Web Search and Data Mining.

15. N. Wardrip-Fruin and N. Montfort (2003), The new media reader, MIT Press.

16. T. Weninger, F. Fumarola, J. Han, D. Malerba (2010), Mapping web pages to database records via link paths, Proceedings of ACM Conference on Information and Knowledge Management.

17. Yahoo Developer (2011), Yahoo! Search BOSS API, http://developer.yahoo.com/search/boss/.

18. Z. Zhuang and S. Cucerzan (2006), Re-ranking search results using query logs, Proceedings of ACM Conference on Information and Knowledge Management.