Why Not, WINE?: Towards Answering Why-Not Questions in Social Image Search

Sourav S Bhowmick Aixin Sun Ba Quan Truong
School of Computer Engineering, Nanyang Technological University, Singapore
assourav|axsun|bqtruong@ntu.edu.sg

ABSTRACT

Despite considerable progress in recent years on *Tag-based Social Image Retrieval* (TagIR), state-of-the-art TagIR systems fail to provide a systematic framework for end users to ask why certain images are not in the result set of a given query and provide an explanation for such missing results. However, as humans, such *why-not* questions are natural when expected images are missing in the query results returned by a TagIR system. Clearly, it would be very helpful to users if they could pose follow-up why-not questions to seek clarifications on missing images in query results. In this work, we take the first step to systematically answer the why-not questions posed by end-users on TagIR systems. Our answer not only involves the reason why desired images are missing in the results but also suggestion on how the query can be altered so that the user can view these missing images in sufficient number. We present three explanation models, namely *result reordering*, *query relaxation*, and *query substitution*, that enable us to explain a variety of why-not questions. We present an algorithm called WINE (**W**hy-not quest**I**on a**N**swering **E**ngine) that exploits these models to answer why-not questions efficiently. Experiments on NUS-WIDE dataset demonstrate effectiveness as well as benefits of WINE.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search Process*

Keywords

Social Image, Flickr, Tag-based image search, Why-not questions, Explanation models

1. INTRODUCTION

The prevalence of digital photography devices (*e.g.,* digital cameras, mobile phones) and increasing popularity of social image sharing platforms and applications (*e.g.,* Flickr, Picasa, and Instagram) have made huge volume of images available online. Many of these online images are socially tagged by their uploaders or viewers and are searchable by tags, which are free-form keywords. Consequently, techniques to support *Tag-based Social Image Retrieval* (TagIR) for finding relevant high-quality images using keyword queries have recently generated tremendous research and commercial interests. In simple words, given a keyword query (or search query) which expresses a user's information need, a TagIR search engine returns a ranked list of images where the images annotated with the most *relevant* tags to the query are ranked higher.

(a) Q(*Rome*) (b) Q(*Rome Colosseum*)

Figure 1: Search results of the two queries in Example 1.

Most existing efforts in TagIR attempt to improve its search accuracy or diversify its search results so as to maximize the probability of satisfying users' search intents [15, 16]. Despite the recent progress towards this goal, it is often challenging to generate high quality search results for a search query which can satisfy search intents of different users. This is because the search intents of users are not always precise and hence difficult to interpret accurately. Often, desired images may be unexpectedly missing in the search results. However, state-of-the-art TagIR systems lack explanation capability for users to seek clarifications on the absence of expected images (*i.e.,* missing images) in the result set. Consider the following set of user problems[1]:

Example 1. Ann is planning a trip to Rome to visit its famous landmarks. She issues a search query `"Rome"` on a tag-based social image search engine. The top-20 matches to her query retrieved by the search engine are depicted in Figure 1(a). Expectedly, many images of Rome's famous landmarks appear as top result matches, such as the *Spiral Stairs*, the *Gallery of Map*, and the *Sistine Chapel*. However, surprisingly, there are no images related to the *Colosseum*, a famous landmark of Ancient Rome, in the top-100 results. So why is it not in the result set? Note that expanding the query by adding the keyword `Colosseum` to it would garner images related to *Colosseum* (Figure 1(b)). However, Ann would

[1] All search results presented in our examples are obtained using a TagIR system following the best performing configuration in [13] on NUS-WIDE data collection.

MM'13, October 21–25, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2502081.2502098.

(a) Q(*lake*) (b) Q(*lake Hangzhou China*)

Figure 2: Search results related to Example 2.

lose images of other interesting landmarks, depriving her to get a bird eye view of different attractions of Rome. ∎

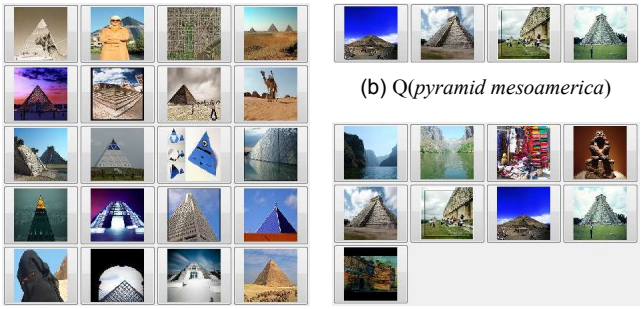
EXAMPLE 2. Bob has returned from a trip to China. He really enjoyed the scenic *Xi Hu lake* in *Hangzhou* city of the *Zhejiang* province. However, Bob has forgotten its name. Hence, he posed the following query to retrieve images related to *Xi Hu lake*: **"lake Hangzhou Zhejiang China"**. Surprisingly, no result is returned by the search engine! Why not? Note that searching for **"lake"** alone is ineffective as Bob primarily wants images of *Xi Hu lake* and not other lakes. In fact, the query **"lake"** returns more than 4000 images, many of these are irrelevant (Figure 2(a)). ∎

EXAMPLE 3. Carlos, a young archaeologist researching on Mesoamerican culture, hopes to find images related to their pyramids. He submits the query **"pyramid"** which returns mostly images related to Egyptian and Louvre pyramids (Figure 3(a)). So why are Mesoamerican pyramids not in the result set? Perplexed, Carlos expands the query by adding the keyword **"Mesoamerica"**, hoping to retrieve relevant images. However, only four images are now returned and among them, only two are really relevant to Mesoamerican pyramids (Figure 3(b)). Are there only two images of Mesoamerican pyramids in the image collection? Thinking that his modified query may be too strict, Carlos now removed the keyword **"pyramid"** from the query. However, only five additional results are returned now and none of these additional images are relevant (Figure 3(c)). So why not more images related to Mesoamerican pyramids can be retrieved? ∎

There is one common thread throughout these problems encountered above, despite the differences in search queries: the user would like to know why certain images are missing in the top-*m* result set of a given query (Examples 1 and 2) or not there in sufficient number (Example 3) and suggestion on how his/her query can be altered effectively to view these missing images in sufficient number. In this paper we refer to this problem as the WHY NOT? problem in TAGIR (formally defined in Section 3). Note that in the above examples, one cannot expect the users to sift through the underlying social image dataset to seek for explanation when they encounter missing results.

At a first glance, it may seem that any large-scale social image search engine (*e.g.,* Flickr) may facilitate answering these original queries more effectively simply because they have very large collection of social images compared to the NUS-WIDE data collection used in the aforementioned examples. For instance, the query in Example 2 returns several images related to *Xi Hu lake* when posed directly on Flickr[2]. Unfortunately, users' expectations are just too diverse to eliminate the WHY NOT? problem in Flickr (detailed in Section 7). For example, consider the query **"pyramid"** in Example 3 directly on Flickr. It only retrieves a single image related to Mesoamerican pyramid in its top-50 result set! In fact, even in large image collection provided by web search engines (*e.g.,* Google, Bing) where data associated with images are not as sparse as social images, the WHY NOT? problem is still encountered. For example,

[2]All results related to Flickr and Google Images are last accessed on July 14th, 2013.



(a) Q(*pyramid*) (b) Q(*pyramid mesoamerica*) (c) Q(*mesoamerica*)

Figure 3: Search results of the three queries in Example 3.

suppose we issue the query **"Manchester city"** on *Google Images*. Interestingly, majority of top-50 images retrieved by it are about the *Manchester City Football Club*! For a user who is not a soccer fan, this result is unsatisfactory due to lack of images related to the city of Manchester.

Our initial investigation shed some light on the possible reasons for the WHY NOT? problem. First, the desired images may be ranked very low in the search results because the same keyword query may express very different search intents for different users. The top-ranked images may be considered relevant by some users but not by others. For instance, reconsider Example 1. The reason Ann could not see the images related to *Colosseum* is because they are ranked too low. The first Colosseum image is ranked 217-*th* in the result set and Ann is unlikely to explore more than 100 images to search for *Colosseum*.

Second, the set of tags associated with images may be noisy and *incomplete*. Note that tags are not from controlled vocabulary and are assigned by users with various different motivations. Consequently, not all keywords mentioned in the search query may appear as tags in relevant images. For instance, consider Example 2. A user may not annotate an image related to *Xi Hu lake* with the tag **Zhejiang** even if she tags it with **Hangzhou**. In fact, none of the images related to *Xi Hu lake* are tagged with **Zhejiang** in the underlying image collection! However, it is unrealistic to expect a user to be aware of this fact.

Third, the query formulated by the user may be too restrictive due to the user's limited understanding of the data collection. That is, there may be a mismatch between the tags that the user *expects* to be associated with her desired images and the *actual* tags that annotate these images in the data collection. In Example 3, Carlos failed to retrieve sufficient number of images annotated with the tag **Mesoamerica** because it is rarely used in tagging images in the image collection. However, Carlos is unlikely to have this knowledge or possess the skill to alter the query to retrieve his desired images.

Clearly, it would be very helpful to Ann, Bob, and Carlos if they could simply pose a follow-up why-not question to the TAGIR engine to seek an explanation for desired missing images and suggestions on how to retrieve them. In this paper, we propose a novel framework called WINE (**W**hy-not quest**I**on a**N**swering **E**ngine) to address this problem. We assume that a user specifies a why-not question in WINE using a *why-not tag*. For instance, in Example 1, after receiving the results as in Figure 1(a), Ann may use a why-not tag **Colosseum** to seek some images related to *Colosseum* to appear in the result list. Similarly, in Example 2, Bob may issue a why-not tag **lake** to specify why he cannot view images related to lakes in *Hangzhou*. Our proposed framework *automatically* generates explanation to a why-not question and recommends *refined* query, if necessary, whose result may not only includes images re-

lated to the search query but also to the why-not question. *To the best of our knowledge, this is the first systematic effort to formulate and address the* WHY NOT? *problem in* TAGIR.

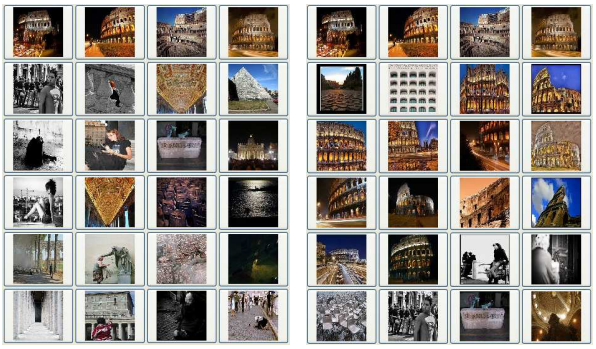
We propose three complimentary explanation models, namely *result reordering*, *query relaxation*, and *query substitution*, that are designed to address three different scenarios of the WHY NOT? problem (detailed in Sections 3.3 and 4-6). Going a step further, we also describe techniques that provide a user useful suggestions on how to alter her query in order to retrieve these desired images. A key feature of WINE is its portability as it is orthogonal to the result retrieval and ranking mechanism of the underlying search engine.

It may seem that the WHY NOT? problem can be addressed by leveraging existing search techniques such as query expansion, query suggestion, and search result clustering. Unfortunately, this is not the case. Specifically, these techniques assume that the search engine captures a user's information need in one way or another by analyzing either the underlying data including click-through data or by presenting the search results in a more organized manner. However, unlike a why-not question in the WHY NOT? problem, these techniques *do not* accept explicit feedback from a user when she is unsatisfied with the search results. Consequently, by modifying a query without such user input, a search engine may generate answers to a modified query that do not agree on the user's original search intent. For instance, in Example 1, if we add the why-not tag `Colosseum` to **"Rome"** (based on query expansion) then the search intent morphs from "famous landmarks of Rome including Colosseum" to "Colosseum in Rome". Consequently, as depicted in Figure 1(b), such query expansion leads to loss of images of interesting landmarks in *Rome* other than *Colosseum*. Notably a why-not question should not alter the original search intent. As another example, consider the query **"pyramid"** in Example 3. An existing query suggestion technique may recommend the tags **"Egypt"**, **"Louvre"**, **"Giza"**, **"Sphinx"**, etc., reflecting the commonly associated concepts to *pyramid*. Clearly, such suggestion not only modifies the search intent of Carlos, but also fails to address his why-not question. That is, without the explicit why-not tag **"Mesoamerica"**, state-of-the-art query suggestion models may fail to speculate that Carlos' interest is in *Mesoamerican pyramid*.

2. RELATED WORK

Tag-based Social Image Retrieval (TAGIR). The research efforts in TAGIR can be broadly categorized into three types, namely *indexing*, *scoring*, and *ranking*. *Indexing* an image in TAGIR involves two complementary tasks: (i) determining the set of tags best describing the image [3, 18], and (ii) quantifying how accurately each of these tags objectively describe the visual content of the image (also known as *tag relevance* [8]). *Scoring* aims to compute a *relevance score* between a keyword query and a tagged image. Relevance score computation typically considers the matching score between the keyword and the image tags and other factors derived from search logs and user click-through data [7,12]. *Ranking* determines the order in which search results (*e.g.,* the top-ranked images) are presented to the searcher. The default image ranking is solely based on the relevance score [8]. More sophisticated methods may diversify the search results so as to satisfy the different possible search intents expressed by the same keyword query [15, 16].

It may seem that our proposed explanation models are closely associated with search results re-ranking [9] and query expansion/substitution [1, 17]. However, search results re-ranking techniques rely on *positive* user inputs (*e.g.,* a user selects relevant results from the initial search results) to improve user satisfaction [5, 9] whereas WINE allows a user to specify *explicit negative* feedback through a why-not tag instead of initial search results.



(a) $\alpha = 0.4$ (b) $\alpha = 0.9$

Figure 4: Reordered results for query (*Rome*) with why-not tag *Colosseum*.

Naturally, results re-ranking techniques cannot be effectively applied to address all why-not questions because there are very few positive results or even no results for users to provide any input.

Zha *et al.* [17] adopts the query suggestion model from IR into TAGIR. Consequently, the suggested tags in [17] are selected to be closely related to the query but distinct among each other. Furthermore, a few representative images are retrieved for each suggested tag to assist users in understanding the suggested tags. However, when there are very few results matching a user query, the co-occurrence based technique used in this scheme becomes unreliable. For instance, Example 2 demonstrates a case where the query is too selective so that adding new tags into the query fail to improve the result quality. Another example is the query **"Leningrad"** (Q_{26} in Section 7) which returns only five images. Consequently, in contrast to WINE, it fails to suggest the query **"saintpetersburg"** instead in order to retrieve more images. In addition, WINE provides clear explanation for reasons behind such suggestion.

Why-Not Questions in Databases. More germane to this work are recent efforts in the database community to provide automatic explanation to a why-not question [2, 6, 14]. To answer why-not questions (*i.e.,* why some expected data items are not shown in the result set) on relational databases, multiple answer models have been proposed, including: *database modification*, *manipulation identification*, and *query modification*. However, these approaches cannot be trivially extended to TAGIR setting due to fundamental differences in data model and query processing technique. Particularly, this is because: (i) the social image data in TAGIR is not represented using relational structure and (ii) there is no concept of query plan, which was exploited by these models, in the context of TAGIR.

3. THE WHY NOT? PROBLEM

We begin by introducing some terminology that is necessary to the understanding of the WHY NOT? problem. Then, we formally define the problem that we are addressing in this paper. Lastly, we introduce the explanation models and present the WINE algorithm.

3.1 Terminology

Given a social image collection $\mathcal{D}$ and the set of all tags $\mathcal{T}$ in $\mathcal{D}$, each image $d \in \mathcal{D}$ is user-annotated by a set of tags $T_d \subseteq \mathcal{T}$. The set of images annotated by tag t is denoted as $D(t)$. Given $T \subseteq \mathcal{T}$, we denote the set of images annotated by all tags in T as $D(T)$.

Given a search query Q with n query tags $t_1, \ldots, t_n$, let $R(Q, \mathcal{D})$ (or simply $R(Q)$ when the context is clear) denote the image search result list of Q on $\mathcal{D}$. For simplicity, we shall assume $t_i \in \mathcal{T}, \forall (i = 1, 2, \ldots, n)$. Each result image $d_r \in R(Q)$ is annotated by query tags $t_1, \ldots, t_n$ and is associated with a *relevance score*, denoted as $rel(d_r, Q)$. $R(Q)$ is assumed to be listed by descending order of

the relevance scores. Recall that WINE is independent of the results ranking technique used to support TagIR. Hence, we can use any state-of-the-art relevance score computation technique [7, 12, 13] for computing $rel(d_r, Q)$. By abusing the notation of lists, we denote the list of images in $R(Q)$ as the image set $D(Q)$. We use $R_m(Q)$ or $D_m(Q)$ interchangeably to denote the top-m images in $R(Q)$.

3.2 Problem Statement

A user asks a why-not question of the form "Why do (top-m) results of my query Q not contain sufficient number of images related to S." The predicate S is defined using a *why-not tag* t_w. So the aforementioned question is equivalent to the following form "Why do (top-m) results of my query Q not contain sufficient number of images related to tag t_w." For example, consider the query in Example 1. Here the query is `"Rome"` and the user Ann wishes to know why the top-100 results of her query do not contain any images related to *Colosseum*. Hence, t_w = `"Colosseum"`.

Note that in this paper we focus on a single why-not tag. While more complex collection of why-not tags could be allowed in theory, our initial analysis of users suggested that such tag collection is often difficult to interpret as the semantics of the why-not question may become unclear (*e.g.,* view images with all why-not tags vs some why-not tags). It is even more challenging when the why-not tags themselves do not appear in the data collection[3].

When attempting to answer a why-not question during social image search, we have three known pieces from which to draw information: the query Q, the search results $R(Q)$ with their relevance scores, and the question t_w. An answer to a why-not question should exploit these information to return to the user potential reasons for the image(s) of interest is missing from the (top-m) results. Further, it should suggest solutions that will enable the user to retrieve and view these missing image(s) along with other relevant results. This leads to a formulation of the Why Not? problem.

Definition 1. *Let Q be a search query on a social image database $\mathcal{D}$, and $R_m(Q)$ be the set of top-m ranked search results, where $0 < m \leq |R(Q)|$. Let t_w be the why-not question and α be a configurable parameter indicating the ratio of desired number of images related to t_w in $R_m(Q)$. Then the goal of the* **Why Not?** ***problem*** *is to find an answer $\mathbb{N}$ comprising (a) the reason for $\frac{|R(t_w) \cap R_m(Q)|}{|R_m(Q)|} < \alpha$, and (b) a suggestion, if executed, ensures any one of the followings: (i) $\frac{|R(t_w) \cap R_m(Q)|}{|R_m(Q)|} \geq \alpha$, or (ii) $\frac{|R(t_w) \cap R_m(Q')|}{|R_m(Q')|} \geq \alpha$, or (iii) $\frac{|R(t'_w) \cap R_m(Q')|}{|R_m(Q')|} \geq \alpha$, where Q' is a modification to Q based on the suggestion and t'_w is a closely related tag to t_w.*

3.3 Explanation Models

We propose the following three explanation models to tackle the Why Not? problem.

Result Reordering Model: Intuitively, in this explanation model we reorder the search results $R(Q)$ so that images related to the why-not tag t_w in $R(Q)$ appear in the top-m results. A key advantage of this approach is that it is simple, fast and is particularly well-suited when the relevant images are in $R(Q)$ but ranked too low (*e.g.,* Example 1). Sample output of promoting the why-not tag `Colosseum` in Example 1 is depicted in Figure 4(a).

Query Relaxation Model: In this model, we identify the set of *selective* tags in Q which are responsible for filtering majority of the relevant images related to t_w from $R(Q)$. These tags can then be removed from Q. Note that this model is suitable when the following two conditions are satisfied: (i) there are few images related to t_w in $R(Q)$; but (ii) there are a large number of images related to t_w in the image collection $\mathcal{D}$. Clearly, the former condition is reasonable since if there are enough images in $R(Q)$, result reordering is a simpler and more effective solution. The latter condition is intuitive because if there are too few relevant images in the image collection, it is impossible for large number of images related to t_w appearing in the result set. For example, consider Example 2. This model identifies that `Zhejiang` is a *selective* tag and advises Bob to remove it from the original query in order to view images related to *Xi Hu lake*. Figure 2(b) depicts the results containing images of *Xi Hu lake* for the relaxed query `"lake Hangzhou China"`.

Query Substitution Model: This explanation model is suitable when there are too few relevant images annotated by t_w in $\mathcal{D}$ (as in Example 3). Consequently, it is impossible to either interpret which images are relevant to the why-not tag or deduce its relations with other tags using $\mathcal{D}$ alone. Therefore, we leverage an external data source (*Wikipedia*) to interpret the semantics of t_w and use this knowledge to infer some *closely related* tags to t_w in $\mathcal{D}$. These tags can then be used as a new query to retrieve the desired images. For instance, consider Example 3. The query `"pyramid Mesoamerica"` is modified to `"pyramid Maya"` after identifying `maya` to be most closely related to `mesoamerica` using Wikipedia. This new query generated 27 query results and many of which are images of Mesoamerican pyramid.

3.4 The WINE Algorithm

Algorithm 1 outlines the procedure for answering the why-not question using the aforementioned explanation models. It takes five inputs, namely, the original query Q, the why-not tag t_w, the number of displayed results m, a set of Wikipedia articles $\mathcal{W}$, and a user-configurable *desired image ratio* α. Note that α represents the fraction of images related to t_w that the user intends to see in her top-m results. Obviously, the why-not question is formulated by the user as α is too low for her satisfaction in the current result set. Hence, the goal of WINE is to explain to the user why there are insufficient images related to t_w in $R_m(Q)$ and provide her suggestion, if executed, will ensure that there are sufficient images related to t_w in $R_m(Q)$ *without* modifying the user's search intent in Q. Towards this goal, Algorithm 1 chooses among the three explanation models by comparing $\alpha \times m$ with two variables s_1 and s_2. Specifically, s_1 is the number of images in both $R(Q)$ and $R(t_w)$ and s_2 is the number of images in $R(t_w)$. Obviously, $s_1 \leq s_2$. We now describe the explanation model selection process.

If $s_1 > \alpha \times m$ (Line 4), then $R_m(Q)$ contains fewer than $\alpha \times m$ results related to tag t_w but $R(Q)$ contains many such images at lower ranks. This scenario is illustrated using a Venn diagram in Figure 5(a). The gray shaded circle represents the images in $R_m(Q)$ which are displayed to the user. The black shaded area represents images related to t_w that appear in $R(Q)$ but not in $R_m(Q)$. Observe that the intersection between $R_m(Q)$ and $R(t_w)$ is small but the intersection between $R(Q)$ and $R(t_w)$ is large enough. In this case, a simple but effective solution is to first notify the user that images related to t_w are lowly ranked (Line 5) and then invoke the *Result Reordering* model (Reorder method in Line 7) to reorder the results (elaborated in Section 4). Note that the reordering ensures that $\frac{|R_m(Q) \cap R(t_w)|}{|R_m(Q)|} \geq \alpha$. The effect of reordering is illustrated in Figure 5(b) where there are now more images with tag t_w in $R_m(Q)$ (the striped area). Notice that the size of $R_m(Q)$ remains unchanged (always equal to m).

If $s_1 \leq \alpha \times m$ but $s_2 > \alpha \times m$ and Q is a multi-tag query (Line 9), then the scenario indicates the followings. First, there are insufficient images in $R(Q)$ related to t_w so that, no matter how we reorder $R(Q)$, we cannot propel sufficient number of images related

[3]This issue can be addressed by building a *query interpretation module* on top of WINE to interpret the why-not question and allow WINE to evaluate multiple why-not tags using the proposed explanation models. However, this is orthogonal to the problem.

Algorithm 1: Algorithm WINE.

Input: A set of n query tags $Q = \{t_1, \ldots, t_n\}$ with search result $R(Q)$, why-not tag t_w with search result $R(t_w)$, set of Wikipedia articles $\mathcal{W}$, the number of displayed results m, the desired image ratio α

Output: A list of images $R(Q_{new})$ addressing the why-not question t_w

```
1  Set flag = false /* true if user accepts a suggestion */;
2  s1 ← |R(Q) ∩ R(tw)|;
3  s2 ← |R(tw)|;
4  if s1 > m × α then
       // m × α < s1 ≤ s2
5      Send notification ℕ;
6      if flag = true then
7          R(Qnew) ← Reorder(R(Q), tw, R(tw), m, α);
8      return R(Qnew)
9  else if s2 > m × α and Q is multi-tag query then
       // s1 ≤ m × α < s2
10     (ℕ, T′) ← Relax(Q, tw);
11     if flag = true then
12         Qnew = Q\T′;
13         Generate results R(Qnew);
14         return R(Qnew)
15 else
       // s1 ≤ s2 < m × α
16     (ℕ, t′w) ← Substitute(Q, tw, β);
17     if flag = true then
18         Modify Q to Qnew based on t′w;
19         Generate results R(Qnew);
20         return R(Qnew)
```

to t_w to the top-m results. Second, if $s_2 > \alpha \times m$, then there are sufficient number of images annotated with t_w in $\mathcal{D}$ which can be potentially retrieved using queries *other than* Q. Consequently, if Q is multi-tag, then it can be relaxed to Q_{new} to retrieve more results related to t_w. This situation is illustrated in Figure 5(c). Unlike in Figures 5(a)-(b), the intersection between $R(Q)$ and $R(t_w)$ is small (black area) but $R(t_w)$ remains large. So, it is reasonable to relax Q to a new query Q_{new} such that its intersection with $R(t_w)$ is larger. Such query relaxation process is invoked by the RELAX procedure (Line 10). Specifically, this procedure efficiently identifies the *selective* tags in Q and suggest to the user to remove them in order to retrieve result set $R(Q_{new}) \supseteq R(Q)$ which contains more images related to t_w. Hence, its output consists of a notification $\mathbb{N}$ that suggests the removal of a *selective tagset* T' from the query. If the user accepts this suggestion ($flag$ is set to `true`) then Q is modified to Q_{new} by removing these selective tag(s) and a new result set $R(Q_{new})$ is generated based on the modified query (Lines 11-13). We elaborate on the RELAX procedure in Section 5.

Lines 16-20 reflect the case when $s_2 \leq \alpha \times m$. In this case, $\mathcal{D}$ contains too few images tagged with t_w. Furthermore, Q simply cannot be relaxed (a tag cannot be removed from Q) as it is possible for Q to be single-tag query. Figure 5(d) illustrates such situation when $R(t_w)$ is too small. However, as tags are inherently noisy, it is possible to retrieve images related to t_w but annotated by *closely-related* tags. We use an external source $\mathcal{W}$ (a set of *Wikipedia* articles), to find the most *closely related* tags t'_w in $\mathcal{D}$ to t_w and Q such that it is associated with sufficiently large number of images in $\mathcal{D}$. The intuition behind this approach is that since t_w and t'_w are closely related, an image relevant to t'_w is likely to be relevant to t_w as well. These steps are encapsulated in the SUBSTITUTE procedure which returns a notification $\mathbb{N}$ that suggests closely related tag t'_w for refining Q (Line 16). Figure 5(d) depicts such situation where $R(t'_w) \cap R(Q_{new})$ is significantly larger than $R(t_w) \cap R(Q)$. If the user accepts the suggestion, then the algorithm substitutes the original query with Q_{new} and generates the new results $R(Q_{new})$ (Lines 17-19). Note that we ensure $|R(Q_{new})| \geq \alpha \times m$ to avoid retrieving too few results. We shall elaborate on this step in Section 6.

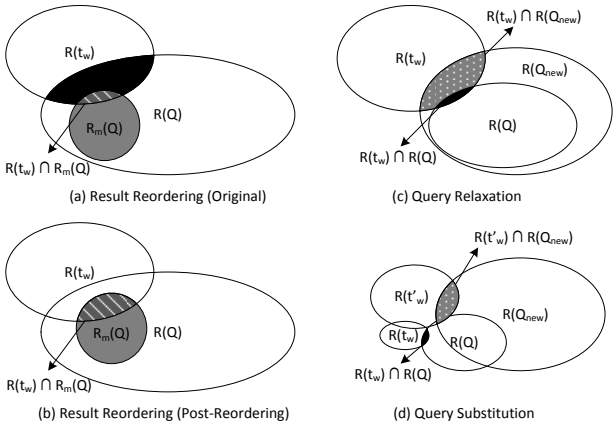


Figure 5: Venn diagram representation of the three models.

Notice that both the query relaxation and query substitution models aim to increase the size of $R(t_w) \cap R(Q)$ (the black area in Figure 5) while the main objective of our solution is to ensure that $R(t_w) \cap R_m(Q)$ (the shaded area in Figure 5) exceeds $\alpha \times m$. Therefore, for these models result reordering is applied (if necessary) when $R(Q_{new})$ is generated (Lines 13 and 19).

4. THE RESULT REORDERING MODEL

In short, the result reordering model reranks the original result images so that there are at least $\alpha \times m$ images relevant to the why-not tag appearing in the top-m results $R_m(Q)$. Given the query Q, each result image $d \in R(Q)$ is assigned a new score by *linearly combining* the relevance score $rel(d, t_w)$ of the why-not tag t_w and the original score $rel(d, Q)$ as follows.

$$rel_w(d, Q, t_w) = (1 - \theta) \times rel(d, Q) + \theta \times rel(d, t_w) \qquad (1)$$

where $\theta = \frac{rel(d_0,Q)}{rel(d_0,Q)+rel(d_w,t_w)}$, d_0 is the $(1-\alpha)m$-th results in $R(Q)\backslash R(t_w)$ and d_w is the $\alpha \times m$-th result in $R(Q) \cap R(t_w)$. Note that we assume $rel(d, t_w) = 0$ when $d \notin R(t_w)$.

Informally, Equation 1 ensures that there are $\alpha \times m$ results with tag t_w in $R_m(Q)$ by ensuring that the $\alpha \times m$-th result in $R(Q) \cap R(t_w)$ appears at the m-th result in $R_m(Q)$ while maintaining the relative order among all results in $R(Q)\backslash R(t_w)$ and among all results in $R(Q) \cap R(t_w)$. The *desired image ratio* $0 \leq \alpha \leq 1$ indicates how many results in the top-m displayed results need to be replaced by images relevant to t_w. In other words, α indicates the user's level of dissatisfaction to the current result list $R(Q)$. Let us illustrate the intuition behind α with an example. Reconsider Example 1 and the why-not question **"`Colosseum`"**. If we set α to a low value[4] (*e.g.,* $\alpha = 0.4$) then we can retrieve images related to `Colosseum` as well as other landmarks as top results (Figure 4(a)). On the other hand, if the value of α is high (*e.g.,* $\alpha = 0.9$) then more images related to `Colosseum` will be laid out as top results (Figure 4(b)). That is, appropriate value of α is driven by specific user's need.

Algorithm 2 outlines the procedure to realize the result reordering model. The key aspect here is that it is designed to avoid sorting $R(Q)$ again as it incurs a cost of $O(|R(Q)| \log(|R(Q)|))$ which can be

[4] In WINE a user does not need to specify any numeric value to set α. Rather she simply drags a slider in the GUI to specify different α values and interactively view the changes in the results (result reordering is very fast as we shall demonstrate in Section 7).

potentially expensive when $R(Q)$ is large. Observe that only the scores of images in $R(Q) \cap R(t_w)$ are modified. Thus, only these images need to be re-sorted. Also notice that $|R(Q) \cap R(t_w)|$ is usually small since users typically ask the why-not question when $|R(Q) \cap R(t_w)| \ll |R(Q)|$. Accordingly, after computing θ (Lines 1-3), we partition $R(Q)$ into two groups, namely L_1 and L_2, corresponding to $R(Q) \cap R(t_w)$ and $R(Q) \backslash R(t_w)$, respectively (Lines 5-10). Each image d in L_1 is assigned a new score $rel_w(d, Q, t_w)$ computed by Equation 1 (Line 7) and re-sorted based on this score (Line 11). Note that the relevance score of each image in L_2 is the same as that in the original query results. Lastly, these two sorted lists are merged into the final result list L (Line 12). The time complexity of Algorithm 2 is $O(|R(Q)| + |R(Q) \cap R(t_w)| \log(|R(Q) \cap R(t_w)|))$.

5. THE QUERY RELAXATION MODEL

While result reordering model can efficiently display relevant images with the why-not tag t_w, it is ineffective when there are too few images with tag t_w in the result set. Notice that arbitrarily removing one or more query tags may not be effective as the query may become too generic leading to the retrieval of too many irrelevant results (*e.g.,* query **"lake"** in Example 2). Hence, a technique to effectively identify the "culprit" query tag(s) in a query for removal is desirable. The query relaxation model is proposed to address this issue. We begin by introducing the notion of *selective tagset* which we shall be exploiting subsequently.

5.1 Selective Tagset

We first introduce *relevance-aware cardinality* and *relevance-aware selectivity* to facilitate exposition. The *relevance-aware cardinality* of a set of query tags $T \subseteq Q$, denoted as $card(T)$, is the weighted sum of the relevance scores of all images annotated by all tags in T. The relevance score used here is the one associated with the why-not tag ($rel(d, t_w)$). Then, $card(T)$ is formally defined as:

$$card(T) = \sum_{d \in \mathcal{D}} rel(d, t_w) \tag{2}$$

Intuitively, a set of query tags T is *selective* when removing a single tag from T would generate *significantly larger* size of query results. Accordingly, given T, we define a child set of T as a $(|T|-1)$-subset of T[5]. We denote the set of all child sets of T as $children(T)$. Then, the *relevance-aware selectivity* of T, denoted as $select(T)$, is defined as follows.

$$select(T) = \frac{\frac{\sum_{T' \in children(T)} card(T')}{|children(T)|}}{card(T)} \tag{3}$$

Informally, a *selective tagset* of a multi-tag query Q is a set of tags $T_{max} \subset Q$ which has maximum relevance-aware selectivity among all possible tagsets in Q. If there are more than one set then the tie is broken arbitrarily. Formally,

Definition 2. *Let $Q = \{t_1, t_2, \ldots, t_n\}$ be a search query where $n \geq 2$. Let t_w be the why-not tag. Then $T_{max} \subset Q$ is a* ***selective tagset*** *iff $\nexists\, T' \subset Q$ such that $select(T') > select(T_{max})$ where $T' \neq T_{max}$.*

5.2 The RELAX Algorithm

We now discuss the algorithm to efficiently detect the selective tagset of a multi-tag query Q. Obviously, computing selectivity is straightforward when the cardinalities of all 2^n subsets of Q are known. The brute-force approach for this task requires traversing all images with tag t_w for each subset resulting in a time complexity of $O(2^n D(t_w))$. Clearly, this is inefficient. Hence, we propose an algorithm that scans $D(t_w)$ only once to detect the selective tagset.

[5]A k-subset of a set is a subset with exactly k elements.

Algorithm 2: Algorithm Reorder.

Input: Result images $R(Q)$, why-not tag t_w with search results $R(t_w)$, α, the number of displayed results m
Output: A reordered list of result images L

1 $d_0 \leftarrow$ the $[(1-\alpha) \times m]$-th result in $R(Q) \backslash R(t_w)$;
2 $d_w \leftarrow$ the $[\alpha \times m]$-th result in $R(Q) \cap R(t_w)$;
3 $\theta \leftarrow \frac{rel(d_0, Q)}{rel(d_0, Q) + rel(d_w, t_w)}$;
4 Initialize two empty temporary list L_1 and L_2;
5 **for** *each result image* $d \in R(Q)$ **do**
6 **if** $d \in R(t_w)$ **then**
7 $rel_w(d, Q, t_w) = (1-\theta) \times rel(d, Q) + \theta \times rel(d, t_w)$;
8 Add d to L_1 with score $rel_w(d, Q, t_w)$;
9 **else**
10 Add d to L_2 with score $rel(d, Q)$;
11 Sort all images in L_1 by their assigned scores descendingly;
12 $L \leftarrow merge(L_1, L_2)$;
13 **return** L

Algorithm 3: Algorithm Relax.

Input: A set of query tags $Q = \{t_1, \ldots, t_n\}$ with search result $R(Q)$, why-not tag t_w with search result $R(t_w)$
Output: The most selective subset $T' \subset Q$, Notification $\mathbb{N}$

1 Initialize the cardinality of each subset of Q to 0;
2 **for** *each image* $d \in D(t_w)$ **do**
3 $T \leftarrow Q \cap T_d$;
4 Increase $card(T)$ by $rel(d, t_w)$;
5 **for** $i = 1 \rightarrow n$ **do**
6 **for** *each subset* $T \subseteq Q$ **do**
7 **if** $t_i \in T$ **then**
8 $T' \leftarrow T \backslash \{t_i\}$;
9 $card(T') \leftarrow card(T') + card(T)$;
10 **for** *each subset* $T \subseteq Q, T \neq \emptyset$ **do**
11 $select(T) = \frac{\frac{\sum_{T' \in children(T)} card(T')}{|children(T)|}}{card(T)}$;
12 $T_{max} \leftarrow$ the subset T with maximum relevance-aware selectivity;
13 Generate notification $\mathbb{N}$ based on T_{max};
14 **return** $(\mathbb{N}, T_{max})$

Observe that based on Equation 2, an image d contributes its relevance score to the cardinality of a tagset T when $T \subseteq T_d$. Therefore, $T \subseteq Q \cap T_d$. Let $exactCard(T')$ be the sum of relevance scores $rel(d, t_w)$ of all images $d \in D(t_w)$ such that $Q \cap T_d = T'$. Then Equation 2 can be rewritten as:

$$card(T) = \sum_{T \subseteq T' \subseteq Q} exactCard(T') \tag{4}$$

That is, the contributing images to $card(T)$ are split into $2^{|Q|-|T|}$ groups, each corresponds to a superset T' of T. Notice that $(Q \cap T_d)$ is unique for all d. Therefore, $exactCard(T')$ can be computed for all sets $T' \subseteq Q$ in a single scan over $D(t_w)$.

Each subset T of $Q = \{t_1, \ldots, t_n\}$ can be encoded as an n-bit bitarray such that the i-th bit is 1 *iff* $t_i \in T$. Using bitarray encoding, a set is a superset of another set *iff* its bitarray supersedes the other set's bitarray[6]. Furthermore, the bitarrays can be arranged in an n-dimension hypercube such that two bitarrays are connected when they are different in only one bit. The sums are then computed using the Hypercube algorithm [11], a popular algorithm in parallel computing. In brief, given an n-dimension hypercube with 2^n nodes, each is labeled with an n-bit bitarray and assigned a value, the hypercube algorithm computes the sum of all values and stores

[6]Bitarray b_1 supersedes bitarray b_2 when $b_1 \wedge b_2 = b_2$

that sum at a node (assumed to be 00...0). The computation consists of n steps. At step i, the current sum of values at all nodes whose i-bit is 1 is added to the current sum of values of the corresponding nodes whose i-bit is 0. After n-steps, the total sum of all values in the hypercube is at node 00...0.

Not that although we do not employ parallel computation for finding the selective tagset, the Hypercube algorithm has a valuable property that can also be exploited for sequential computation. After n steps, the value at each node a is the sum of the original values at all nodes whose bitarrays supersede a's bitarray. Specifically, we can compute the cardinality of *all* subsets of Q using Equation 4 by a single n-step hypercube process having complexity $O(2^{n-1}n)$.

Algorithm 3 realizes the above intuitions to compute the selective tagset. Lines 1-4 compute the $exactCard(T)$ of each subset T of Q while Lines 5-9 add them into *card* following Equation 4 and the Hypercube algorithm. Lines 10-11 compute the selectivity score from the cardinality score following Equation 3. Line 12 simply finds the subset of Q with maximum relevance-aware selectivity. Based on this subset, the algorithm generates the explanation $\mathbb{N}$ for the user (Line 13). Notice that the relevance-aware selectivity of *all* query tag subsets is computed in the aforementioned steps. Consequently, we can easily extend our approach to notify the user with top-k most selective tagsets instead of the only most selective tagset and allow her to select the desired tagset for removal. The time complexity of the algorithm is $O(n \times 2^{n-1} + |D(t_w)|)$. Notice that in practice, n is usually small and $n \ll D(t_w)$.

6. THE QUERY SUBSTITUTION MODEL

The query relaxation model discussed in the preceding section is not effective when there are very few images (if any) associated with a why-not tag in the entire image collection. In this case, even the most relaxed query fails to retrieve sufficient number of desired results (Example 3). In this section, we present the query substitution model to tackle such scenario.

The lack of existence of sufficient images matching the why-not tag poses two intertwining challenges. Firstly, the user-specified why-not tag cannot be leveraged directly for generating explanation to the why-not question as it is unlikely that the user wishes to see a very small number of result images (if any) associated with this tag. Secondly, while the desired images are likely to be annotated by some closely related tag(s) to the why-not tag, it is difficult to find these related tags using traditional tag co-occurrence or tag relevance measures as both of these require sufficiently large number of matching images to be effective.

In order to address the aforementioned challenges, we resort to the external source *Wikipedia* to interpret the why-not tag. Specifically, we shall use it to measure the *strength* of relationship between tags to find a *closely related* tag t_c to the why-not tag. It is then used in lieu of the why-not tag to guide the why-not question answering process as well as modification of the query.

In particular, t_c must satisfy the following three conditions: (a) it must annotate sufficiently large number of images in the image collection; (b) it should be *closely related* to the why-not tag t_w; and (c) it should be *related* to the query keywords Q. The first condition is obvious. Otherwise t_c would offer little benefit compared to the why-not tag. The remaining conditions ensure necessary semantic relationship between t_c, t_w, and query tag(s). Notice that we use the phrase *closely related* for the second condition whereas *related* for the last. This is because t_c must be closest to the why-not tag. We now formally quantify the relatedness of a selected tag t_c.

DEFINITION 3. *Given a query $Q = \{t_1, \ldots, t_n\}$, a why-not tag t_w, and a term t_c, the* ***tag relatedness*** *of t_c, denoted as $\Phi(t_c)$, is defined as follows:*

$$\Phi(t) = (1-\beta)\frac{\sum_{1\le i\le n} sim(t_c, t_i)}{n} + \beta * sim(t_c, t_w)$$

where $\beta \in [0,1]$ and sim denotes tag similarity of a pair of tags.

Algorithm 4: Algorithm SUBSTITUTE.

Input: A set of n query tags $Q = \{t_1, \ldots, t_n\}$ with search result $R(Q)$, why-not tag t_w with search result $R(t_w)$, a configurable parameter β

Output: The closely related tag t_c, Notification $\mathbb{N}$

1 Initialize $(n+1)$ sorted lists $s_0, s_1, \ldots, s_n$, each stores the most similar tags to $t_w, t_1, \ldots, t_n$, each neighbor tag also has large enough images in the image collection $\mathcal{D}$;
2 Initialize weights w_i with $w_0 = \beta$ and $w_i = \frac{1-\beta}{n}$ for $1 \le i \le n$;
3 Initialize a heap H with at most k tags, when it is full, the minimum is removed;
4 **for** *each* $i \in [0, n]$ **do**
5 $t \leftarrow s_i.next()$ whose score is in $score_i(t)$;
6 **for** *each* $j \in [0, n], j \ne i$ **do**
7 $score_j(t) \leftarrow WLM(t, t_j)$;
8 $score(t) = weightedSum(w_0, score_0(t), \ldots, w_n, score_n(t))$;
9 Add t to H;
10 $thresDesc_i \leftarrow w_i \times (s_i.peekNext() - s_i.last())$;
11 $threshold \leftarrow weightedSum(w_0, s_0.last(), \ldots, w_n, s_n.last())$;
12 **while** $threshold > H.min$ **do**
13 $i \leftarrow arg\,\min_{i\in[0,n]} thresDesc_i$;
14 $t \leftarrow s_i.next()$ whose score is in $score_i(t)$;
15 **for** *each* $j \in [0, n], j \ne i$ **do**
16 $score_j(t) \leftarrow WLM(t, t_j)$;
17 $score(t) = weightedSum(w_0, score_0(t), \ldots, w_n, score_n(t))$;
18 Add t to H if $score(t) > H.min$;
19 $thresDesc_i \leftarrow w_i \times (s_i.peekNext() - s_i.last())$;
20 $threshold \leftarrow weightedSum(w_0, s_0.last(), \ldots, w_n, s_n.last())$;
21 Related tag $t_c \leftarrow H.max$;
22 Generate notification $\mathbb{N}$ based on t_c;
23 **return** $(\mathbb{N}, t_c)$

Clearly, in the above definition the key challenge is to compute *tag similarity* as we cannot simply compute it using tag co-occurrence or tag relevance measures.

6.1 Tag Similarity Computation

We compute tag similarity by adopting the *Wikipedia Link Measure* (WLM), an efficient and highly accurate technique to measure the similarity between two Wikipedia articles using hyperlinks [10]. Computing tag similarity using WLM consists of two steps as in [10], *disambiguation* and *article similarity*. The disambiguation process corresponds to the mapping process from each tag to each article. Specifically, this step exploits two heuristics: *dominant meanings* and *mutual disambiguation*. While a tag can have many meanings, it usually has only a few dominant meanings frequently used in most cases. For example, the tag `pyramid` probably refers to either the geometric shape pyramid or a building with that shape, but rarely refers to the arena at *Memphis*. *Mutual disambiguation* means that when two tags are used together, they are probably related to each other. For instance, in Example 3 since `pyramid` and `mesoamerica` are used together, `pyramid` probably refers to a building and not a geometric shape.

The *article similarity* $docSim(a_1, a_2)$ between two articles a and b is computed using WLM as:

$$docSim(a,b) = 1 - \frac{\log(max(|A|,|B|)) - \log(|A \cap B|)}{\log(|W|) - \log(min(|A|,|B|))}$$

where A and B are the sets of all articles that link to a and b, respectively, and W is the entire Wikipedia. The intuition behind the above formula is that two related articles tend to be referred to by lots of common articles. Finally, the tag similarity of a tag pair is computed as the article similarity of their disambiguated articles.

Let us illustrate the process using the tags `pyramid` and `mesoamerica`. In Wikipedia, `mesoamerica` only matches to a single article about *Mesoamerica* region whereas `pyramid` can match to multiple articles. Among them, only pyramid shape and pyramid building are dominant meanings. The article similarity between the *Mesoamerica* article and the two *pyramid* articles are then computed and compared. The larger one is the similarity between *Mesoamerica* and *pyramid* building. Thus, they are the disambiguated meanings of `mesoamerica` and `pyramid`, respectively. Hence, their similarity is the similarity score between the two tags.

6.2 The SUBSTITUTE Algorithm

We now discuss the algorithm to locate a tag with highest tag relatedness value efficiently and refine the query Q accordingly. Observe that the tag relatedness of t_c is effectively a linear combination of its similarities to all the query tags and the why-not tag which can be pre-computed and pre-sorted. Hence, we can cast our problem as the *combining fuzzy grade problem* which can be solved efficiently using the Threshold Algorithm (TA) [4].

Algorithm 4 outlines the procedure to identify a tag t_c with highest tag relatedness value. Specifically, we *extend* the TA as follows. Recall from Definition 3, the similarity to the why-not tag has largest impact on the overall score. Therefore, it is reasonable to proceed on the list storing the why-not tag's similarities faster instead of same speed as in TA. To this end, for each sorted list, we can estimate the threshold decrease when the next processed node is select from this list and choose the list with maximum threshold decrease. This optimization ensures the quickest termination.

Let us elaborate on Algorithm 4 now. Line 1 initializes $(n + 1)$ sorted lists containing the most similar tags to the why-not tag and the query tags. Each list is assumed to support three functions *next*() to proceed the list, *last*() to look at the last tag seen by sorted access and *peekNext*() to peek the next tag but does not proceed the list. Line 2 initializes the weights following Definition 3 which shall be used in the *weightedSum*() function to compute the weighted sum of all similarity scores. Line 3 initializes a min-heap H which automatically discards the minimum score tag when a tag with larger score is added. That is, H stores the top-k most related tags. Lines 4-10 simply process the first tag within each list similarly to the steps of TA. Note that processing the first tag of all lists is required to initialize the threshold (Line 11). Lines 13-20 are repeatedly executed until the *threshold* is smaller than the minimum value of H. Within each loop, the next processed tag is retrieved from the list with maximum threshold decrease (Line 13). Lines 14-20 are similar to Lines 5-11. Observe that since H stores top-k related tags, we can easily extend our approach to notify the user with top-k closely related tags instead of most closely related tag and allow her to select the desired tag for query modification.

The worst case time complexity of Algorithm 4 is basically the time complexity of TA which is $O(|\mathcal{T}|^{n/(n+1)}k^{1/n+1})$ where $\mathcal{T}$ is the set of all tags in the image collection $\mathcal{D}$ and n is the query size. Theoretically, in the worst-case, Algorithm 4 may iterate through all tags in $\mathcal{T}$. However, in practice, it is very unlikely. Since k is usually small, the top-k related tags to the why-not tags and the query tags are very likely to be very similar to either the why-not tag or the query tags. Consequently, Algorithm 4 usually terminates quickly and the execution time is very short. Additionally, for each tag (*i.e.,* keyphrase) in Wikipedia, we only need to sort and store a small list of similar tags. This space optimization is crucial since there are more than three million articles in Wikipedia.

Table 1: Sample queries for evaluating result reordering model

Id	Query [WHY NOT tag]	$\|R(Q)\|$	$\|R(t_w)\|$	$\|R(Q) \cap R(t_w)\|$	$F(Q, t_w)$
Q_1	Rome [colosseum]	909	20	18	7
Q_2	music [classical]	2213	119	27	6
Q_3	sea [fish]	9016	2279	556	6
Q_4	Paris [Louvre]	2338	2279	178	4
Q_5	Paris [arcdetriomphe]	2338	190	30	2
Q_6	Paris architecture [cathedral]	250	967	25	0
Q_7	sky clouds [bird]	8525	4031	132	1
Q_8	sky clouds [plane]	8525	2014	345	1
Q_9	newyork [statueofliberty]	3822	46	25	3
Q_{10}	sunset sunrise sky [horizon]	238	1195	43	40

7. EXPERIMENTS

WINE is implemented in Java 1.7 using Lucene 3.0.3 as the underlying index engine[7]. All experiments are conducted on an Intel Xeon X5570 machine with 12GB memory. In our experiments, unless specified otherwise, we set the number of displayed images $m = 50$ and the desired image ratio $\alpha = 0.2$.

7.1 Experimental Setup

Dataset. Since off-the-shelf image search engines (*e.g.,* Flickr) typically disallow full access to their data and some statistics (*e.g.,* relevance scores) which are required by WINE, we cannot evaluate WINE directly on top of such search engines. Hence, we are confined to conduct experiments using the NUS-WIDE dataset containing 269,648 images from Flickr. All tags provided in the dataset are used in our experiments without filtering. The underlying TAGIR system used in our experiments follows the best performing configuration in [13] for multi-tag queries to facilitate evaluation of the query relaxation and query substitution models. For query substitution, we used the English Wikipedia dump released on 30 January, 2010[8] which contains 3.2 million articles and more than 266 million hyperlinks.

Query Set. We asked 14 volunteers (undergraduate students in computer science and business majors) each to evaluate a subset of more than 500 randomly chosen queries. A volunteer may issue a why-not tag if the results returned by a query do not completely satisfy his/her search intent. 114 queries received why-not tags from the volunteers. Note that some queries (*e.g.,* Q_7, Q_8) received multiple why-not tags resulting in 125 query why-not tag pairs. Then, for each query and why-not tag pair, we asked the volunteers to label which result images are relevant in the original *and* modified result set, taking into account both the original query and the issued why-not tag. Out of these 125 queries, 60, 36, and 29 queries invoked the result reordering, query relaxation, and query substitution models, respectively. Due to space limitation, here we chose to report 30 queries (10 queries per model) as described in Tables 1-3. These queries are chosen based on multiple factors. First of all, for these queries most volunteers demonstrated low satisfaction. Secondly, we ensure that the number of keywords in these queries vary from 1 to 4. Note that the average number of keywords in a search query is 2.2 [13]. Thirdly, we ensure that the number of images satisfying these queries vary significantly (*i.e.,* $R(Q)$ is varied from 0 to 9016). Last but not the least, majority of these queries fail to return sufficient number of result images annotated by the why-not tags even when they are posed directly on Flickr. Specifically, among the top-50 Flickr results [9], the number of images annotated

[7]The video of WINE is available at `http://youtu.be/A42i2geQZVk`.

[8]`http://download.wikimedia.org/enwiki/20100130/`.

[9]The setting used is the default setting of Flickr search available at `http://www.flickr.com/search/?q={query}`.

Table 2: Sample queries for evaluating the query relaxation model

Id	Query [WHY NOT tag]	$\|R(Q)\|$	$\|R(t_w)\|$	$\|R(Q) \cap R(t_w)\|$	Selective tag(s)	New query	$\|R(Q_{new})\|$	$\|R(Q_{new}) \cap R(t_w)\|$	$F(Q, t_w)$
Q_{11}	Apple Computer [iPhone]	152	196	4	computer	apple	1318	97	0
Q_{12}	Apple Computer [laptops]	152	37	8	computer	apple	1318	15	3
Q_{13}	Las Vegas Nevada [casino]	44	140	5	las nevada	vegas	224	54	24
Q_{14}	sea summer [surfing]	592	310	6	summer	sea	9016	81	1
Q_{15}	sea summer [tornado]	592	628	1	summer	sea	9016	13	0
Q_{16}	sea summer [carnival]	592	348	1	sea	summer	5055	76	4
Q_{17}	christmas night [santaclaus]	146	44	3	night	christmas	1400	31	1
Q_{18}	christmas night [firework]	146	48	1	christmas	night	8806	28	0
Q_{19}	sky cloud rain [rainbow]	92	1245	7	cloud	skyrain	384	33	2
Q_{20}	lake Hangzhou Zhejiang China [lake]	0	4336	0	zhejiang	lake hangzhou china	8	8	43

by the why-not tag t_w is reported in the last columns of Tables 1-3 ($F(Q, t_w)$). Observe that it is smaller than 10 ($\alpha < 0.2$) for all but four queries. *This demonstrates the need for why-not questions even in very large social image collection such as Flickr.*

Performance Metric. We conducted experiments to evaluate the *effectiveness* and *efficiency* of the three explanation models. The *effectiveness* is measured by *Precision@K* ($P@m$), which is the ratio of the relevant images among the top-m retrieved images for a query ($m = 50$ in our evaluation). To the best of our knowledge, this is the first work to answer why-not questions in TAGIR. Hence, there is a lack of benchmark dataset for the evaluation. The labels annotated by the volunteers are used as ground-truth labels.

7.2 Effectiveness of the Explanation Models

The sample queries and related features used for effectiveness study are reported in Tables 1-3. Notice that $|R(Q) \cap R(t_w)|$ and $|R(t_w)|$ for each query satisfies the condition necessary to invoke the corresponding model with $m = 50$ and $\alpha = 0.2$. The $P@50$ of all queries are reported in Figure 6 (for the time being, the reader is requested to ignore the bars related to Coo and PRF).

Result Reordering. Observe that the original results of Q_1-Q_{10} have relatively high $P@50$, reflecting the fact the user is generally capable of forming suitable queries for their needs. However, we also notice that their $P@50$ remains imperfect and for some queries' it is as low as 0.4. Such imperfection reflects the fact the some of users' expectations have not been met. For instance, in Q_3 for the query `sea`, the user expects some images related to fishes in the sea. But these images are missing in the original results. In summary, our results clearly show that the simple result reordering model can improve or at least maintain the $P@50$ for all sample queries and, for some queries, bring the precision close to perfect (Q_7, Q_8, Q_{10}).

Query Relaxation. Expectedly, we observe significant increase in result sizes ($|R(Q_{new})|$) for this model (Table 2) for Q_{11}-Q_{20}. By observing the changes in result sizes, it is clear that the query relaxation model successfully identifies and removes the selective tag(s) from the original query. For instance, for Q_{19}, a user expects to see `rainbow` when searching for rainy skies but rainbows usually only appear in clear rather than cloudy sky. Similarly, in Q_{13} and Q_{20}, `Vegas` and `Hangzhou` are common names for **"`Las Vegas`"** and **"`Zhejiang`"**, respectively. Hence, the latter tags are not necessary. We also observe that query relaxation helps to rectify user's knowledge. For Q_{11}, although *Apple Computer* was a former name of *Apple Inc.*, it has dropped the word "Computer" when *iPhone* products were launched (as suggested by our solution). Figure 6 reports significant increase in $P@50$ values for most queries after query relaxation ($Q_{11} - Q_{20}$). Particularly, Q_{14}, Q_{17}, Q_{19} and Q_{20} reach near perfect value.

Query Substitution. Table 3 reports the top-3 most closely related tags based on Wikipedia and the suggested new queries. Note that in an interactive TAGIR system, a user can select one of these related tags for new query formulation. In our evaluation, we always choose the most closely related tag (top-1) for query substitution. For instance, the why-not tag **"`mesoamerica`"** in Q_{21} is replaced by **"`maya`"** to form a new query **"`pyramid maya`"**. Note that a user may also use the query tag as why-not tag as in Q_{26} to reflect that images retrieved by tag **"`leningrad`"** are not what the user expects for **"`leningrad`"**. Note that `Leningrad` is the former name of `Saint Petersburg` and, due to its recent popularity, there are significantly more images related to the modern `Saint Petersburg` than to the historical `Leningrad` in Flickr. Hence, using tag **"`saintpetersburg`"** will retrieve more relevant images.

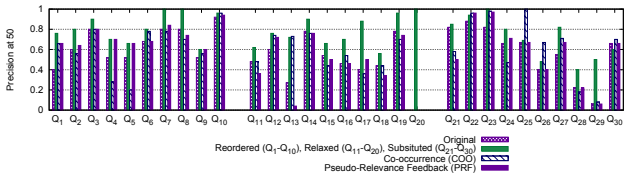


Figure 6: $P@50$ of WINE, COO, PRF against original. WINE: Reordered (Q_1-Q_{10}), Relaxed ($Q_{11}-Q_{20}$), Subsituted ($Q_{21}-Q_{30}$)

The $P@50$ values of the original results and the results of the substituted queries are reported in Figure 6 (Q_{21}–Q_{30}). Among the 10 sample queries, six underwent significant increase in $P@50$ after query substitution, three do not have significant changes in $P@50$ (Q_{21}, Q_{22} and Q_{25}) while the remaining one suffers a slight drop in $P@50$ (Q_{30}) but the result qualities are not necessarily worst. Q_{30} highlights a situation where the image collection indeed has too few relevant data on `picasso`. Consequently, all suggested queries are not truly similar to `picasso`. Nevertheless, we notice that the drop in $P@50$ is small (6%).

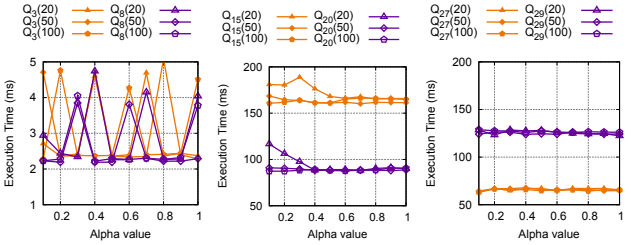
7.3 Comparison with Query Expansion

Next, we compare WINE with two popular query expansion techniques based on *tag co-occurrence* and *pseudo-relevance feedback*, denoted as Coo and PRF, respectively [1]. In brief, Coo expands a query by finding the top-N tags which co-occurs with most query keywords, where tag co-occurrence is measured using Jaccard coefficient. PRF expands the query by analyzing the top-100 results from the original query and finding the top-N most significant tags in these images. The tag significance is measured as KL-divergence between their distribution in the top-ranked images and the whole collection. We set $N = 5$.

The $P@50$ of Coo and PRF are reported in Figure 6 which clearly shows the superiority of WINE. For example, consider Q_{11} and Q_{12} which are identical queries (`apple computer`) except that the follow-up why-not questions are different (`iphone` versus `laptops`). Thus, Coo and PRF detect same expanded tags for both Q_{11} and Q_{12} but their results on Q_{12} is clearly better than on Q_{11}. It is because, our dataset (NUS-WIDE) is built when iPhone is newly introduced but laptops and macbook have had a long history. Thus, both Coo and PRF can only expand using tags `macbook` or `laptop` but not `phone` or `iphone`. Meanwhile, WINE can exploit the why-not tag and correctly modify the queries on both cases. Similarly,

Table 3: Sample queries for evaluating the query substitution model

Id	Query [WHY NOT tag]	$\|R(Q)\|$	$\|R(t_w)\|$	$\|R(Q) \cap R(t_w)\|$	Top-3 most related tag(s)	New query	$\|R(Q_{new})\|$	$F(Q, t_w)$
Q_{21}	pyramid [Mesoamerica]	605	9	4	maya teotihuacan aztec	pyramid maya	27	1
Q_{22}	skating [winters]	248	9	0	winter spring summer fall snow	winter skating	62	7
Q_{23}	Balkan [Serbian]	62	9	2	serbia montenegro bulgaria	balkan serbia	51	6
Q_{24}	pagoda lake [Zhejiang]	35	6	0	hangzhou jiangsu guangxi	hangzhou	20	4
Q_{25}	Lombardy lake [brescia]	3	8	0	lakegarda verona trento	lakegarda	13	10
Q_{26}	Leningrad [Leningrad]	5	5	5	saintpetersburg moscow kiev	saintpetersburg	21	9
Q_{27}	Persian city [Persepolis]	11	9	0	fars shiraz yazd iranian safavid	fars	22	4
Q_{28}	Japan [ninja]	3647	55	1	ninja katana samurai	ninja	55	0
Q_{29}	India [bangalore]	2938	0	0	bangalore karnataka chennai	bangalore india	85	1
Q_{30}	painting [picasso]	1294	74	0	cubism painter impressionism	cubism	595	0



(a) Result Reordering(b) Query Relaxation(c) Query Substitution.

Figure 7: The impact of α and efficiency of WINE.

for Q_{21}, both Coo and PRF fail to speculate that the user's intent could be on `Mesoamerican pyramid` while for Q_6, both cannot recognize that `cathedral` architecture is a crucial part of `Paris architecture`. Furthermore, when the original results are irrelevant, PRF also suffers from query drift problem [1]. For example, PRF expands Q_{13} with tags `jet`, `flying`, and `flight`. Since WINE uses explicit user feedback (*i.e.,* the why-not tag), it does not suffer from these problems and offer better results. However, we also notice that, for queries where original results are fairly good (*e.g.,* Q_{10}, Q_{22}, Q_{23}), query expansion have comparable performance with WINE. For example, for Q_{10}, query `sky sunrise sunset` are expanded with tags `clouds`, `sun`, `blue`, and `sea` and images with both `sky` and `sea` tend to have a `horizon` (why-not tag).

7.4 Effect of α

In this experiment, we measure the effect of α and m on the efficiency of our proposed algorithms. We select two sample queries for each model and execute them using the corresponding model across different values of α and for $m \in \{20, 50, 100\}$. All queries are selected so that the conditions to execute their corresponding models are met. Figure 7 reports results of our experiment. We use $Q_i(j)$ to denote query Q_i for $m = j$. From the results, it is clear that our proposed algorithms are efficient across the whole spectrum of α and m and both these parameters do not greatly influence the query performance. It is worth noting that WINE *is efficient even if we consider very large image collection* as it is not very sensitive to the underlying dataset size. WINE focuses only on the top-m result images or the cases when the result size is too small. In both these scenarios, the number of images it needs to process is fractionally small compared to the underlying data.

8. CONCLUSIONS & FUTURE WORK

We have proposed a novel framework called WINE for explaining why-not questions on query results in a TAGIR system. WINE exploits three explanation models to automatically generate explanation for a user's why-not question as well provide suggestion for effective query modification to retrieve desired images. We empirically demonstrated that the techniques presented herein produce good quality response to user's why-not questions. Despite the proven effectiveness of WINE in handling a wide variety of why-not questions, it suffers from two limitations. First, WINE assumes that search results of a query are returned as a ranked list of images. While this is true for a broad spectrum of image search engines, more sophisticated search engines may organize the result images in form of clusters or summaries and directly return them to the user. It is interesting to explore how a user can pose why-not questions on them. Second, our query substitution model leverages Wikipedia to discover closely-related keywords of a why-not tag. Here we assume that such keywords indeed exists in Wikipedia. However, in certain cases (*e.g.,* images related to recent events) such closely-related tag may exists in the image collection but not in Wikipedia. We are currently exploring how WINE can be extended to address these limitations. Additionally, we intend to study the WHY NOT? problem in the context of social video search (*e.g.,* Youtube). In summary, the results of this paper are an important first step in this regard.

Acknowledgements: Aixin Sun and Ba Quan Truong were supported by Singapore MOE AcRF Tier-1 Grant RG13/10.

9. REFERENCES

[1] C. Carpineto, R. Giovanni. A Survey of Automatic Query Expansion in Information Retrieval, *ACM Comput. Surv.*, 44(1), 2012.
[2] A. Chapman and H. V. Jagadish. Why not?, *In ACM SIGMOD*, 2009.
[3] T.-S. Chua, et al. NUS-WIDE: a real-world web image database from National University of Singapore, *In ACM CIVR*, 2009.
[4] R. Fagin, A. Lotem, M. Naor. Optimal aggregation algorithms for middleware, *In PODS*, 2001.
[5] J. He, E. Meij, Maarten de Rijke. Result diversification based on query-specific cluster ranking, *JASIST*, 62(3), 2011.
[6] Z. He, E. Lo. Answering Why-not Questions on Top-k Queries, *In ICDE*, 2012.
[7] V. Jain, M. Varma. Learning to re-rank: query-dependent image re-ranking using click data, *In WWW*, 2011.
[8] X. Li, et al. Learning Social Tag Relevance by Neighbor Voting, *IEEE Trans. Multimedia*, 11(7), 2009.
[9] C. Liu, et al. Post-rank reordering: resolving preference misalignments between search engines and end users, *In CIKM*, 2009.
[10] D. Milne, I. A. Witten. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links, *Proc. AAAI Workshop on Wikipedia and Artificial Intelligence*, 2008.
[11] David A. Padua. Encycl. of Parallel Computing, Springer, 2011.
[12] G. Smith, et al. Evaluating Implicit Judgments from Image Search Clickthrough Data, JASIST, 63(12), 2012.
[13] A. Sun, S. S. Bhowmick, et al. Tag-based social image retrieval: An empirical evaluation, *JASIST*, 62(12), 2011.
[14] Q.-T Tran, C.-Y Chan. How to ConQueR why-not questions, *In ACM SIGMOD*, 2010.
[15] R. H. van Leuken, et al. Visual diversification of image search results, *In WWW*, 2009.
[16] M. Wang et al. Towards a Relevant and Diverse Search of Social Images, *IEEE Trans. on Multimedia*, 12(8), 2010.
[17] Z.-J. Zha et al. Visual Query Suggestion, *In ACM MM*, 2009.
[18] G. Zhu, et al. Image tag refinement towards low-rank, content-tag prior and error sparsity, *In ACM MM*, 2010.