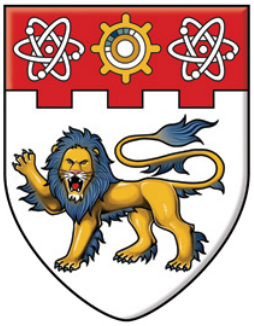

NANYANG TECHNOLOGICAL UNIVERSITY

CSC415: Information Retrieval

An interactive photo search interface of Singaporean Art

Le Quang Hoa	U1020208K
Jesper Norberg	N1301992E
Zkriya Rakhimberdiyev	U1021923E
Suryadi Hartanto Tjandra	U1120125J

Examiner: Assoc prof Hoi Chu Hong

Abstract

This project aims to provide a search engine specialised in finding images of Singaporean Art relevant to the query. To achieve this end, we crawled data fram Flickr, indexed and ranked it using Solr and Jetty, and then finally created a user interface for interacting with the search engine. We then evaluated our program's performance using two customized experiments to determine how well we performed compared to Flickr in regards to both average document relevancy in regards to the main topic as well as average image relevancy in regards to a provided query related to the main topic. For both our experiments, our search engine performed better than Flickr. With our primary goal fulfilled, we consider the project a success.

Contents

1 Introduction **1**

2 Problem Definition **1**
2.1 Problem description 1
2.2 Related Work in literature 2

3 Approach **2**
3.1 Crawling Flickr 2
3.2 Designing schematisation 3
3.3 User Interface 7

4 Experiments **8**
4.1 Experiment 1 8
4.2 Experiment 2 9
4.3 Results analysis 10

5 Discussion of Pros and Cons **11**
5.1 Pros 11
5.2 Cons 11

6 Conclusions **11**
6.1 Summary of project achievements 11
6.2 Future Directions for improvements 11

7 References **12**

1 Introduction

When approaching the task of determining the problem that the project is to solve using information retrieval, we wanted to address a niche topic previously unaddressed, whilst making it limited enough to be processed in a moderate amount of time using a single computer. In the end, the topic we decided on was Singaporean Art. At the moment, the sites dedicated to Singaporean art all have a specific museum, or specific type of art that they refer to. There is no site which provides general content from all artforms and venues. With this project, we seek to fill this void.

The Arts represent an important topic, central to any society since before written word was created. Art in itself provides a language that cuts across arbitrary borders such as race, culture and economy. It allows individuals a means for self-expression, allowing the inner world to extend into reality. While doing so, it develops higher order thinking skills, such as analysis, evaluation and problem finding, in other words your creative and critical thinking.[1] This actually extends into other fields, enhancing both social skills and academic prowess.[2] Of course, it also provides a great source of personal enjoyment.

With our project we want to provide Singaporeans with an interface for browsing the art of the city. Not limiting ourselves to exhibitions or certain types of art, we allow for everything the Singaporean themselves consider to be art to be represented. In order to determine this, we have specified a number of attributes which can be accredited to art, and then retrieving related information, such as image and description, from the website Flickr.

2 Problem Definition

2.1 Problem description

The problem given is to implement a complete search engine. The engine in question can be divided into three major stages; 1. Crawling, 2. Indexing and Ranking, and 3.Querying.

For crawling, a variety of websites were provided as possible targets for information retrieval. Out of these, we decided to use Flickr, as it best corresponded to the goal of our project; providing pictures relevant to the subject Art. It also provides the most metadata that can be accessed through their API.[5]

To crawl the data, we were to use the site's provided API, given they existed. As this project crawled Flickr for data, this was the API used.[3] We did not end up needing any third party libraries for the crawling.

The entire dataset crawled was required to be at least 10,000+ records, eg. in our case 10,000+ images from Flickr.

For indexing, we were strongly recommended to use Solr and Jetty for indexing and ranking, and so we did. Several suggestions were given for possible innovation in regards to indexing and ranking, and we ended up implementing several of the suggestions, as discussed in section 3.

For querying, we were to provide a simple yet friendly User Intercae (UI) for handling the user's queries. Our ways of implementing this is further discussed in section 3.

2.2 Related Work in literature

Abbasi et al[6] conducted a study to measure the effectiveness of Flickr tags to search for landmarks. E.K.Chung and J.W.Yoon[7] also performed an analysis of the differences between Flickr tags and search queries. Both of their results showed that using Flickr tags alone is not sufficient to produce accurate search results. In our experiment, we are trying to search images not only based on tags, but also other datas available, such as titles and description.

Lacic et al[8] has performed a study on Apache Solr, and concluded that Solr performs well as a text-based search server.

Text-based image retrieval (TBIR) is used instead of content-based image retrieval (CBIR). This is based on the papers by T. Pavlidis[**?**]. The existence of the 'semantic gap', the lack of corellation of the interpretation of the image with the pixel values of the image, causes CBIR to be unreliable in detecting images that have similar interpretation, but different values. This is unsuitable to our topic, art. Many art forms can be categorized into the same type (sculpture, painting, dance, etc), but have dissimilar appearance.

Our project is to build text-based image retrieval system based on image metadatas using Apache Solr with art as the topic.

3 Approach

3.1 Crawling Flickr

We are using the Flickr API to crawl for data. The data provided by the API will first need to be processed in order to be used.

Flickr's Java API provided various information, such as photo ID's, photo descriptions, tags, geolocations, etc.

After studying the information, we identified a list of fields we considered relevant for our users, and through which we will find documents. The fields in question we have used are shown in table 1.

Field	Value
id	6983691429
title	ION orchard
description	ION Orchard is poised to become the...
location	1.304883,103.831795
flickr url of photo	http://www.flickr.com/photos/kazeeee/3814375563/
posted date	2009-08-12T16:03:04Z
tags	ion, orchard, singapore
thumbnail url of photo	http://farm4.static.flickr.com/35/b182ccc9c9_s.jpg

Table 1: Data types used

Next, we performed the ollowing procedures in order to index our data set: 1. Converted the data set from its native format into a JSON format, which is supported by Solr. 2. Configured Solr to apply transformations to the text in the document during indexing. 3. Added documents to Solr using the HTTP POST request method.

To provide an example, here is how a document would look after a conversion into JSON format:

```
{
  "tags": [
    "ion",
    "orchard",
    "singapore",
  ],
  "id": "6983691429",
  "title": " ION Orchard",
  "thumbnail": " http://farm4.static.flickr.com/35/b182ccc9c9\textunderscores.jpg ",
  "location": "1.304883,103.831795",
  "description": " Orchard is poised to become the \&quot;centre of ... ",
  "date": "2009-08-12T16:03:04Z",
  "url": " http://www.flickr.com/photos/kazeeee/3814375563/"
}
```

3.2 Designing schematisation

We use the 'id' field in order to uniquely identify each document in the index. The fields 'title', 'description' and 'tags' are useful for searching for text information. The fields 'posted', 'date' and 'location' are helpful for ordering the documents by methods other than direct similarity, such as distance from a specific point. All of these fields except for 'thumbnail' were chosen to be indexed and stored. The 'thumbnail' field was still stored for displaying photo in search results, but does not need to be indexed. The thumbnail is the only image we show, its link gives the full size image on Flickr.

```
Schema.xml
 <fields>
   <field name="id" type="string" indexed="true" stored="true" required="true"/> #A
   <field name="tags" type="text_flickr_soft" indexed="true"
        stored="true" multiValued="true"></field> #B
   <field name="title" type="text_flickr" indexed="true" stored="true"/>#B
   <field name="description" type="text_flickr" indexed="true" stored="true"/> #B
   <field name="location" type="loc_flickr" indexed="true" stored="true"/>#B
   <field name="date" type="tdate" indexed="true" stored="true"/> #B
   <field name="url" type="string" indexed="true" stored="true"/> #B
   <field name="thumbnail" type="string" indexed="false" stored="true"/> #C
   <field name="spell" type="textSpell" indexed="true"
        stored="false" multiValued="true"/> #D
   <field name="catch_all" type="text_flickr" indexed="true"
        stored="false" multiValued="true"/> #E
<fields/>
   <uniqueKey>id</uniqueKey>
   <copyField source="tags" dest="catch_all"/>
   <copyField source="title" dest="catch_all"/>
   <copyField source="description" dest="catch_all"/>
   <copyField source="url" dest="catch_all"/>
   <copyField source="title" dest="spell"/>
   <copyField source="description" dest="spell"/>
```

The #A field is used to uniquely identify the documents in the index The #B fields define a name and type and whether they are indexed or not The #C field is not indexed but is stored for display purposes The #D spell field will be analyzed using a field type called "textSpell" The #E catch_all field should not be stored as it is populated from another field

"catch_all" and "spell" are copy fields. Copy fields allow us to populate one field from one or more other fields. For example, "catch_all" field contains the text from tags, title, and description field. It helps users find documents using a single field.

Next, we will describe what text analysis tools we used for building our index. Our data contains html markup, hashtags, hyphened terms, etc.

```
<fieldType name="text_flickr" class="solr.TextField"
positionIncrementGap="100">
 <analyzer type="index"> #A
  <charFilter class="solr.HTMLStripCharFilterFactory"/>
    <charFilter class="solr.PatternReplaceCharFilterFactory"
        pattern="([a-zA-Z])\1+"
        replacement="$1$1"/>
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.WordDelimiterFilterFactory"
        generateWordParts="1"
        splitOnCaseChange="1"
        splitOnNumerics="1"
```

```
      stemEnglishPossessive="1"
      preserveOriginal="0"
      catenateWords="1"
      generateNumberParts="1"
      catenateNumbers="0"
      catenateAll="0"
      types="wdfftypes.txt"/>
 <filter class="solr.StopFilterFactory"
      ignoreCase="true"
      words="lang/stopwords_en.txt"
      />
 <filter class="solr.LowerCaseFilterFactory"/>
 <filter class="solr.ASCIIFoldingFilterFactory"/> <filter class="solr.KStemFilterFactory"/>
</analyzer>
<analyzer type="query"> #B
 <charFilter class="solr.PatternReplaceCharFilterFactory"
      pattern="([a-zA-Z])\1+"
      replacement="$1$1"/>
 <tokenizer class="solr.WhitespaceTokenizerFactory"/>
 <filter class="solr.WordDelimiterFilterFactory"
      splitOnCaseChange="0"
      splitOnNumerics="0"
      stemEnglishPossessive="1"
      preserveOriginal="0"
      generateWordParts="1"
      catenateWords="1"
      generateNumberParts="0"
      catenateNumbers="0"
      catenateAll="0"
      types="wdfftypes.txt"/>
 <filter class="solr.LowerCaseFilterFactory"/>
 <filter class="solr.ASCIIFoldingFilterFactory"/>
 <filter class="solr.StopFilterFactory"
      ignoreCase="true"
      words="lang/stopwords_en.txt"
      />
 <filter class="solr.SynonymFilterFactory" #C
      synonyms="synonyms.txt"
      ignoreCase="true"
      expand="true"/>
 <filter class="solr.KStemFilterFactory"/>
</analyzer>
</fieldType>
```

#A is an analyzer for indexing documents, #B is an analyzer for processing user queries, and #C is synonym prossing which is performed on the query side, not during indexing

HTMLStringCharFilterFactory was used to remove HTML markup from text. Then PatternReplaceCharFilterFactory was applied to filter characters using regular expressions. To configure this factory, we defined two attributes: pattern and replacement. The pattern is regular expression that identifies sequences of repeated letters. For example, word "cooool" will be changed to "cool". WhitespaceTokenizerFactory splits the text on whitespaces. Next, WordDelimeterFilter splits a token into sub-words using various customized rules and preserves special characters on hashtags. Finally, we used a filter based on Krovetz Stemmer: KStemFilterFactory. This stemmer is less aggressive in its transformations than other stemmers like the PorterStemmer. For example, for words "requirements" and "wedding" it will produce "requirement" and "wedding" respectively.

We configured our search engine to provide suggestion for misspelled words. As we mentioned earlier, we have a spell field in our schema. For that field we did not apply stemming. As a result in our dictionary we got many word variations which help in suggesting accurate results.

Each photo in our dataset has geographical identification metadata. It consists of longitude and latitude coordinates from where the photo was taken. We used this data to allow for sorting results by a distance from a geographical position of the user. Moreover, the user is able to perform range queries based on posted dates of the photos.

For our application, we utilized a field faceting mechanism. Field faceting is the process of requesting the unique values found in a particular field, along with the number of documents in which they are found when a search is performed. For example, in application we facet upon the 'tags' field. The query looks as follows:

```
http://localhost:8983/solr/collection1/select?q=statue&wt=json&indent
=true&facet=true&facet.field=tags
```

.

The results of the faceting:

```
"facet_counts":{
 "facet_queries":{},
 "facet_fields":{
 "tags":[
 "singapore",2692,
 "sculpture",584,
 "asia",463,
 ...
]},
 "facet_dates":{},
 "facet_ranges":{}},
```

When the user selects "sculpture" tag and clicks on it, we add it as a filter to our subsequent query. Hence our query will be represented as follows:

```
http://localhost:8983/solr/collection1/select?q=statue&fq=tags%3Asculpture&w
t=json&indent=true&facet=true&facet.field=tags
```

.

As a result we narrowed down the search results with another filter:

```
"facet_counts":{
 "facet_queries":{},
 "facet_fields":{
 "tags":[
 "singapore",578,
 "gardens",252,
 "chinese",245,
 ...
]},
 "facet_dates":{},
 "facet_ranges":{}},
```

If the user runs a third search, we will add yet another filter and then he/she will narrow it down even further.

We used Solr's default Similarity class for computing similarity between documents and query. First, it makes use of a Boolean model to filter out any documents that do not match the query. Then, it utilizes a Vector Space model for scoring, drawing the query as a vector, as well as an additional vector for each document. The similarity score for each document is based upon the cosine similarity between the query vector and that document's vector. For Vector Space scoring model, following components are calculated:

- tf – term frequency in document. It measures how often a term appears in the document
- idf – inverse document frequency. It measures how often the term appears in the document across the index.
- coord – number of terms in the query that were found in the document
- lengthNorm – measure of the importance of a term according to the total number of terms in the field
- queryNorm – normalization factor, so that queries can be compared
- boost(index) – boost of the field at index-time
- boost(query) – boost of the field at query-time ...

3.3 User Interface

Our web application is written in JavaServer Pages and consists of only one page. For our UI we used a Bootstrap framework. It is equipped with a

bunch of features that facilitates the development of websites. Moreover, we utilized MapBox's maps to display our results on the map. Our application and MapBox's maps interact using GeoJSON objects. GeoJSON is a format for encoding geographic data. In order to talk to Solr we used SolrJ API. It enables developer to add, update, and query Solr more conveniently. For example, we query solr using following code:

```
String url = "http://localhost:8983/solr";
 SolrServer server = new HttpSolrServer( url );
 SolrQuery solrQuery = new SolrQuery();
solrQuery.setFacet(true);
 ...
```

And we get our results using:

```
QueryResponse rsp = server.query(solrQuery);
```

4 Experiments

We have designed two experiments in order to evaluate the performance of our search engine.

4.1 Experiment 1

The first experiment is to test for the mean comparative ratio between our site and Flickr. To test for this, we perform the following: Out of the group of documents which contain "Art", which is our key word, how many of these contain the word X, where X will be a variety of related values, such as statue, or drawing. We do this for both Flickr and our search engine. We then calculate the relevance ratios for each of these terms, how big a part of the "Art" group is X? We then compare the ratios between our site's ratios and Flickr's ratios, and then take the mean average out of them. The resulting value we dub the mean comparative ratio between our site and Flickr. It denotes how close in matching we are to Flickr. If the values are close to identical, it implies we've retained the integrity of the Flickr search engine, and if they aren't, it implies some change has happened. The results from our experiment is presented in table 2.

Query	Our hits	Flickr hits	Our ratio	Flickr ratio	Comparative ratio
Art	4873	15950	1:1	1:1	1:1.00
Conceptual Art	6	10	1:812	1:1595	1:1.96
Architecture Art	1036	1577	1:5	1:10	1:2.00
Buildings Art	990	961	1:5	1:17	1:3.40
Photography Art	554	1554	1:9	1:10	1:1.11
Painting Art	252	550	1:19	1:29	1:1.53
Drawing Art	51	174	1:96	1:92	1:0.96
Acrobatics Art	0	85	0	1:188	undefined
Circus Art	6	14	1:812	1:1139	1:1.40
Street Performance Art	72	237	1:68	1:67	1:0.99
Dance Art	300	786	1:16	1:20	1.1.25
Music Art	318	709	1:15	1:22	1:1.47
Musical Art	186	327	1:26	1:49	1:1.88
Opera Art	48	321	1:102	1:50	1:0.49
Film Art	91	562	1:54	1:28	1:0.52
Theatre Art	236	806	1:21	1:20	1:0.95
Museum Art	2095	3388	1:2	1:5	1:2.50
Statue Art	360	389	1:14	1:41	1:2.93
Gallery Art	457	497	1:11	1:32	1:2.91
Fashion Art	59	166	1:83	1:96	1:1.16
Design Art	559	704	1:9	1:23	1:2.56
Sculpture Art	921	1049	1:5	1:15	1:3.00
Mean value					1:1.64

Table 2: Experiment 1

4.2 Experiment 2

The second question we attempt to answer experimentally is: How accurate are our queries? If we search for a term, how many of the top 20 hits display the expected result? For example, if we search for "statue", how many of the 20 most relevant hits actually depict a statue? As mentioned in section 3, relevance is determined by cosine similarity. The results from the experiment follows in table 3.

Query	No rel. hits on our site	No rel. hits on flickr	Comparative ratio
Art	20/20	20/20	1:1.00
Conceptual Art	3/6	7/10	1:1.40
Architecture	18/20	18/20	1:1.00
Buildings	19/20	20/20	1:1.05
Photography	16/20	19/20	1:1.19
Painting	12/20	4/20	1:0.33
Drawing	15/20	6/20	1:0.40
Acrobatics	13/20	5/20	1:0.38
Circus	11/20	6/20	1:0.55
Street Performance	20/20	6/20	1:0.30
Dance	3/20	7/20	1:2.33
Music	15/20	1/20	1:0.07
Musical	20/20	1/20	1:0.05
Opera	18/20	11/20	1:0.61
Film	1/20	1/20	1:1.00
Theatre	5/20	3/20	1:0.60
Museum	17/20	4/20	1:0.24
Statue	20/20	12/20	1:0.60
Gallery	10/20	8/20	1:0.80
Fashion	8/20	10/20	1:1.25
Design	3/20	5/20	1:1.67
Sculpture	19/20	12/20	1:0.63
Mean value	0.67%	0.44%	1:0.66

Table 3: Experiment 2

4.3 Results analysis

For experiment 1, we notice that the mean comparative ratio is quite far from the expected 1:1. In fact, it is 1:1.64. This means that for an average art related search term, Flickr gives on average 64% greater chance for a query to be unrelated to one of the primary areas of art. This is likely due to the fact that Flickr contains duplicates, which are removed when we crawl the data. This also explains the difference in number of hits, our 4873 hits contain the same amount of unique documentss as Flickr's 15950. As providing a site for finding art, being biased toward typical art forms is a good thing, and indeed one of our goals with the search engine.

For experiment 2, the mean value of the comparative ratio is $1{:}0.66 = 1.51{:}1$. In other words, our search engine is 52% more likely to show an image relevant to the query. This is likely due to filtering away all non-art related content. This makes finding relevant images much easier, which was indeed one of our primary goals.

5 Discussion of Pros and Cons

5.1 Pros

We leveraged built-in Solr tools to deal with nuances of our dataset. Specifically, we used the PatternReplaceCharFilterFactory with a regular expression to collapse repeated characters down to a maximum of two to deal with terms like "sooo". We used the WhitespaceTokenizer and WordDelimeterFilter to preserve leading and @ characters on hashtags. We also used stemming and synonym expansion to improve our search application's matching capabilities. As a result, we believe that our solution is effective for analyzing our dataset using only built-in Solr's tools.

The obvious strength of our search engine is that for its designated topic Art it does perform better than Flickr both in regards to document relevancy in regards to the main topic as well as average image relevancy in regards to a provided query related to the main topic. It provides a platform for those interested in Singaporean art to achieve greater accuracy with their queries.

5.2 Cons

In our application we compare images based on their textual descriptions. The descriptions are generally provided by the users contributing the images in question, who often only provide very brief and sometimes even inaccurate details. These issues of poor image description can greatly affect image retrieval effectiveness based on textual metadata. Without proper textual description of an image, it is difficult to reliably match the image using only text queries. Thus it is hard to retrieve the relevant images with high accuracy.

6 Conclusions

6.1 Summary of project achievements

Our primary goal was to provide a search engine which allowed users to retrieve pictures related to Singaporean art. As both our experiments show that our search engine both provides better relevancy to standard art forms and better relevancy in terms of query result, we consider the primary goal to be fulfilled. With that in mind, we label the project a success.

6.2 Future Directions for improvements

There is always room for improvement. First and foremost, we could widen our search engine to also look for images on other websites. This would allow for an even greater spread in results.

Another possibility for improvement would be to further develop the indexing and ranking algorithms. Examples of this would be ****TODO: provide one or two examples of possible algorithmic improvements (if any)****

7 References

[1] G. Spruce. "Teaching and learning for critical thinking and understanding." Routledge publishing, 2002.

[2] Benefits of Music and Arts. University of Michigan. http://sitemaker.umich.edu/356.murphy/benefits_of_music_and_art_education (retrieved 2014-04-11)

[3] Flickr API. http://www.flickr.com/services/api/ (retrieved 2014-04-11)

[4] T. Chong. "The State and the New Society: The Role of the Arts in Singapore Nation-building". Asian Studies Review, Vol. 34 Issue 2, p131-149, June 2010.

[5] M. Batko et al. "Crawling, indexing, and similarity searching images on the web". In Proceedings of SEDB '08, the 16th Italian Symposium on Advanced Database Systems, Mondello, Italy, 2008, pages 382–389.

[6] Abbasi et al. "Exploiting Flickr Tags and Groups for Finding Landmark Photos" in 31th European Conference on IR Research, ECIR 2009, Toulouse, France, April 6-9, 2009. Proceedings, pp. 654-661.

[7] Chung, EunKyung, and JungWon Yoon. "Categorical And Specificity Differences Between User-Supplied Tags And Search Query Terms For Images. An Analysis Of "Flickr" Tags And Web Image Search Queries." Information Research: An International Electronic Journal 14.3 (2009): ERIC. Web. 14 Apr. 2014.

[8] S.Patil and R.Regunathan."Search Engine Optimization of E-Commerce Website using Apache SOLR". International Journal of Advanced Research in Computer Science, vol. 4, no. 8, May-June 2013.

[9] T. Pavlidis. "Limitations of Content-based Image Retrieval" [Online]. Available: http://www.theopavlidis.com/technology/CBIR/PaperB/vers3.htm