Building a Diversity Featured Search System by Fusing Existing Tools

Jiayu Tang, Thomas Arni, Mark Sanderson, Paul Clough
Department of Information Studies, University of Sheffield, UK
{j.tang, t.arni, m.sanderson, p.d.clough}@sheffield.ac.uk

Abstract

This paper describes our diversity featured retrieval system which are built for the task of ImageCLEFPhoto 2008. Two existing tools are used: Solr and Carrot^2. We have experimented with different settings of the system to see how the performance changes. The results suggest that the system can indeed increase diversity of the retrieved results and keep the precision about the same.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.3.7 Digital Libraries

General Terms

Measurement, Performance, Experimentation

Keywords

Solr, Carrot^2, Image Retrieval, Indexing Expansion, Query Expansion, Diversity, Clustering

1 Introduction

In this paper, we describe how to quickly set up a diversity featured search system by combining existing tools that are available to the public, namely Solr [2] and Carrot^2 [3]. We further describe how to tune the system for better performance in the context of ImageCLEFPhoto 2008.

2 Tools

We use Solr for text search and Carrot^2 for increasing diversity of results.

Solr is a text search server based on the popular search library Lucene [1]. Solr provides some very useful and convenient features. The web-services like API allows users to index documents via XML over HTTP, query it via HTTP GET and receive results in XML format. The fields and field types of documents can be easily defined in the schema.xml file. In this file, users can also specify the Solr "out of the box" tokenizers and token filters to use for indexing and query. In addition, the HTML administration interface gives users comprehensive insight into the system.

Carrot^2 is a open source search results clustering engine. It provides five different algorithms for automatically organising search results into thematic categories. Carrot^2 works as a pipeline of three kinds of components: input components, filter components and visualisation components. Input components obtain search results from a source of choice (e.g. YahooAPI, GoogleAPI, Lucene, Solr, etc.), then filter components apply clustering algorithms to the search results, and finally visualisation components render the clustered results to the user.

3 System Setup

In ImageCLEFPhoto 2008, each image comes with 9 fields of data: DOCNO, TITLE, DESCRIPTION, NOTES, LOCATION, DATE, IMAGE, THUMBNAIL and TOPIC. We decided that TITLE, DESCRIPTION, NOTES, LOCATION are the fields that would provide useful information for text based image retrieval. Therefore, we constructed a new field named TEXT by combining the text from the four fields, and specified it as the default search field in Solr. Solr's default configuration of tokenizers and token filters is used in our experiments, namely WhitespaceTokenizer, SynonymFilter, StopFilter, WordDelimiterFilter, LowerCaseFilter, EnglishPorterFilter and RemoveDuplicatesTokenFilter. More details on each tokenizer and token filter can be found on [2]. After feeding the Solr server with all the 20,000 documents in XML format, we have a running search engine that is able to return a ranked list of documents based on the query submitted by the user. Note that all the fields are indexed by Solr.

In Carrot^2, we construct an input component for acquiring results from the Solr server. Then, a filter component is used for clustering the results, and a visualisation component is used for displaying the clusters and their members. Since ImageCLEFPhoto 2008 assesses the S-recall [5] of the top 20 results of the list to be submitted, we use the following procedure to find the best 20 results from the input/ranked list generated by Solr:

```
repeat
  for each doc in the input list do
    if membership of the current doc not exist in the list of cluster ID then
      add the current doc to output list
      remove the current doc from input list
      add its membership to the list of cluster ID
      if length of output list equals 20 then
        break
      end if
      if length of cluster ID list equals number of clusters found by Carrot$^2$ then
        clear the list of cluster ID
        break
      end if
    end if
  end for
until length of output list equals 20
```

Basically, the above pseudo code chooses documents based on two criteria: 1. appear as early as possible in the input/ranked list, 2. cover as many different clusters as possible. The 20 documents chosen by the above procedure form the list to be submitted for assessment.

4 Experiments

We have submitted 32 runs, all of which are EN-EN-AUTO-TXT, meaning that all the submissions are English-English monolingual runs using fully automated text clustering methods. Among them, there are 8 groups, each of which includes 4 runs with the same system configuration except that different number of documents are used for clustering. In other words, for each group, we vary the number of documents (40, 60, 80 and 100) that are used from the top of the input list for clustering by Carrot^2, in order to get different output lists. For example, a run with top 40 documents applies clustering on the 40 documents and chooses 20 for submission based on the procedure in Section 3.

We increasingly change another 3 kinds of settings of the system to examine how the performances would change. Firstly, for topics whose CLUSTER field is Country, City, State or Location, we specify Carrot^2 to cluster the documents by the LOCATION filed, otherwise by the TEXT field. Secondly, we change the parameters of the clustering algorithm used in Carrot^2. Finally, we apply expansion to the indexing and query stage. In the following, we describe each group of runs.

4.1 Baseline

This group uses the default settings of Solr and Carrot^2. In terms of text retrieval, expansion is not applied during indexing or query. Clustering is applied to the TEXT field for all topics. Carrot^2's default clustering algorithm (Lingo [4]) and default parameters (0.150, 0.775) are used. This group is used as a comparison baseline for other groups.

4.2 Clustering by Location

It seems that for topics that have been specified to be clustered by Country, City, State and Location, the LOCATION filed in each document contains the essential information for clustering. Therefore, such topics are clustered based on the LOCATION field. This group is different from the Baseline group in that clustering on some topics are based on the LOCATION field, while others based on the TEXT field.

4.3 Parameters of Lingo set to (0.05, 0.95)

Lingo [4] is a singular value decomposition based clustering algorithm that has been implemented in Carrot^2. The first parameter 0.05 is the Cluster Assignment Threshold, determining how precise the assignment of documents to clusters should be. Lower threshold assign more documents to clusters and less to "Other Topics", which contains unclassified documents. With a low threshold, more irrelevant documents are also assigned to the clusters. The second parameter 0.95 is the Candidate Cluster Threshold, determining how many clusters Lingo will try to create. Higher values give more clusters. This group is based on group 4.2, but uses (0.05, 0.95) as the parameters.

4.4 Parameters of Lingo set to (0.10, 0.90)

This group is the same as group 4.3 except that it uses (0.10, 0.90).

4.5 Parameters of Lingo set to (0.15, 0.85)

This group is the same as group 4.3 except that it uses (0.15, 0.85).

4.6 Indexing and Query Expansion with (0.05, 0.95)

This group is based on group 4.3, but applies indexing and query expansion in Solr. Specific domain ontologies have been a popular choice for expansion. Cyclopedia websites such as Wikipedia have also been adopted for expansion. Due to limited time, we used neither of the approaches. Instead, we examined the topics and data-set, and then manually built an indexing expansion list and query expansion list, as shown in Appendix. These lists are used as the synonym lists in Solr for indexing and query. In the expansion lists, for lines containing ">=", any of the words before ">=" are replaced by words after ">=" during expansion. For example, "ship, ships => ship, vehicle" will replace any "ship" or "ships" by "ship, vehicle". For lines without ">=", any word from the line will be replaced by the whole set of words from the same line. For example, "USA" or "United States of America" or "US" will be replaced by "USA, United States of America, US".

4.7 Indexing and Query Expansion with (0.10, 0.90)

This group is the same as group 4.6 except that it uses (0.10, 0.90) in Lingo.

4.8 Indexing and Query Expansion with (0.15, 0.85)

This group is the same as group 4.6 except that it uses (0.15, 0.85) in Lingo.

5 Results and Discussions

Figure 1 and 2 depict the precision and cluster recall at 20 of all submitted runs. For example, in Figure 2, group 1 corresponds to the results of cluster recall generated by the group of runs described in Section 4.1.

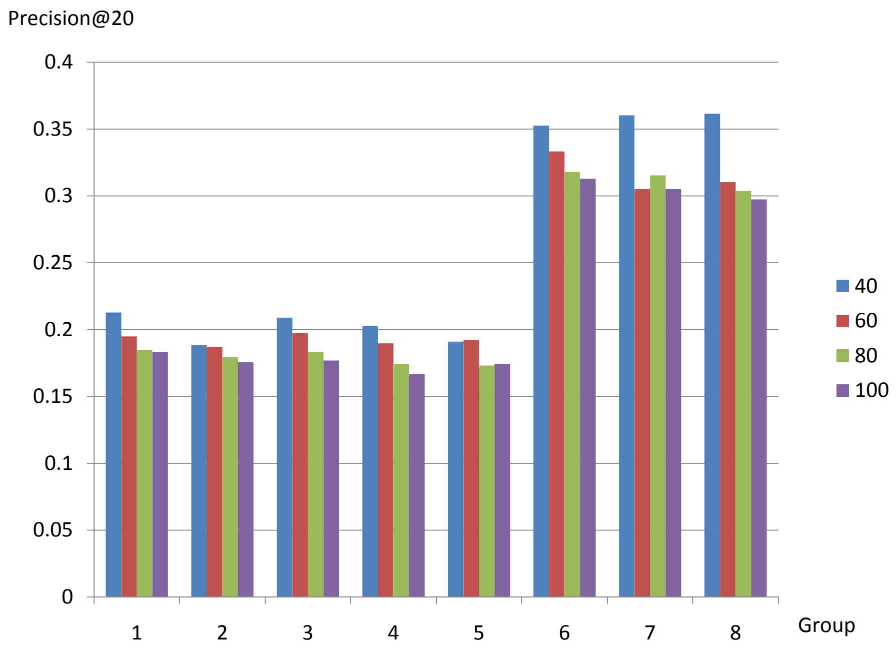

Figure 1: Precision at 20 for all groups of runs.

As shown in Figure 1, group 6, 7 and 8 have clearly higher precisions than the other groups. This is due to indexing and query expansion using the expansion lists in Appendix. As have been mentioned, the expansion lists are built based on an examination of the data-set. For example, in Topic 48 “vehicle in South Korea”, “South Korea” normally means the country, so there is not much ambiguity. However, “vehicle” can mean many things, e.g. car, bus, boat. Intuitively, expanding names of different types of vehicle with the word “vehicle” during indexing will boost the precision, because many images are only annotated with specific vehicle names rather than the word “vehicle”. Therefore, after indexing expansion, “car” becomes “car vehicle”. Similarly, we expanded specific animal names with the word “animal”, so “fish” becomes “fish animal”.

In terms of cluster recall, we can see in Figure 2 that different parameters of the clustering algorithm (Lingo) have led to different performances. It is a little surprising that group 2 (Section 4.2) performed worse than group 1 (Section 4.1), the reason of which needs to be examined. In addition, it seems that low Cluster Assignment Threshold (i.e. more documents are clustered) and high Candidate Cluster Threshold (i.e. more clusters are created) give better cluster recall. In our experiments, (0.05, 0.95) gives the best results: group 3 is better than group 4 and 5; group 6 is better than group 7 and 8.

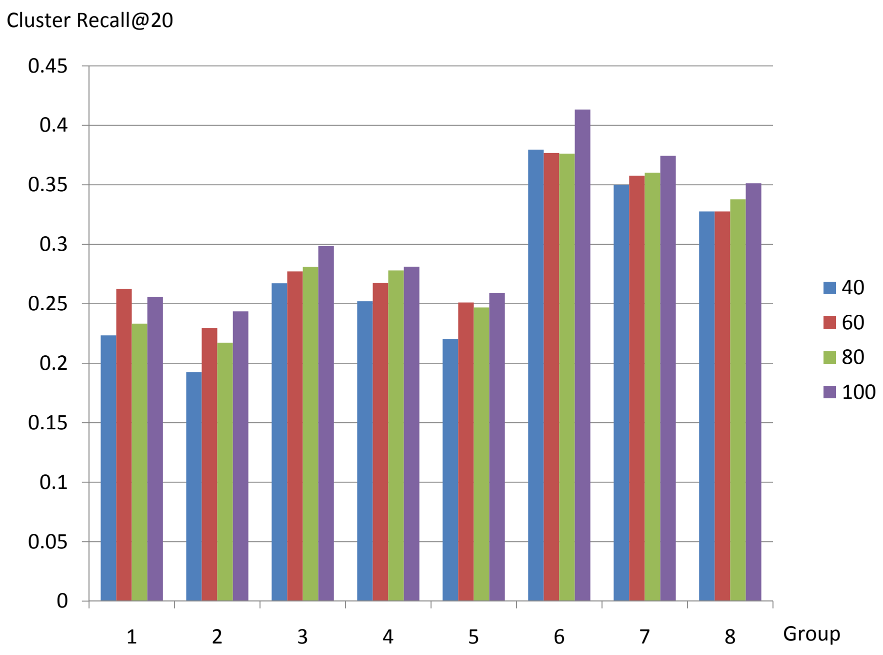

Figure 2: Cluster recall at 20 for all groups of runs.

By comparing the two figures, it can also be noticed that groups with the same settings of Solr have very similar precisions, no matter what settings of Carrot^2 were used. For example, with different parameters of Lingo, the precisions of group 6, 7 and 8 are relative stable, but the values of cluster recall vary. This can be seen as an evidence that the diversity featured retrieval system can make the results more diverse while maintaining the precision.

6 Conclusions and Future Work

This paper has described our submissions to ImageCLEFPhoto 2008. We have changed 4 kinds of settings: the field used for clustering, the number of images used for clustering, indexing and query expansion, and parameters of the clustering algorithm. The results suggest that indexing and query expansion can fairly improve precision. Appropriately chosen clustering method can increase diversity of the results while keeping precision almost the same.

As we have mentioned, the performance of group 2 is a little against our initial anticipation. It would be interesting to find out why. On the other hand, we plan to build an automatic expansion approach using resources such as ontologies, rather than using the manually built expansion lists.

7 Acknowledgements

Work undertaken in this paper is supported by the EU-funded TrebleCLEF project (Grant agreement: 215231) and by Tripod (Contract No. 045335).

References

[1] Apache lucene project. http://lucene.apache.org. [Visited 23/07/08].

[2] Apache solr project. http://lucene.apache.org/solr/index.html. [Visited 23/07/08].

[3] Carrot^2 project. http://project.carrot2.org/. [Visited 23/07/08].

[4] S. Osinski, J. Stefanowski, and D. Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Proceedings of the International Conference on Intelligent Information Systems*, pages 359–368, Zakopane, Poland, 2004.

[5] Cheng Xiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 10 – 17, Toronto, Canada, 2003.

A Appendix

A.1 Indexing Expansion List

```
# vehicles
ship, ships => ship, vehicle
cutter, cutters => cutter, vehicle
train, trains => train, vehicle
rail, rails => rail, vehicle
locomotives, locomotive => locomotive, vehicle
wagon, wagons => wagon, vehicle
tractor, tractors => tractor, vehicle
bus, buses => bus, vehicle
car, cars => car, vehicle
forklift, forklifts => forklift, vehicle
boat, boats => boat, vehicle

    # animals
Alligator, Alligators => Alligator, animal
Turtle, Turtles => Turtles, animal
Ducks, Duck => Duck, animal
Dolphins, Dolphin => Dolphins, animal
Fish => Fish, animal
Whale, Whales => Whale, animal
Pelicans, Pelican => Pelicans, animal
blowfish => blowfish, animal
shoal => shoal, animal
Shark, Sharks => Shark, animal
Crocodile => Crocodile, animal
orcas => orcas, animal
Angelfish => Angelfish, animal
kangaroo, kangaroos => kangaroo, animal
wallaby, wallabies => wallaby, animal
koalas, koala => koala, animal
wombats, wombat => wombat, animal
quokkas, quokka => quokka, animal
platypuses, platypus => platypus, animal
possums, possum => possum, animal
Tasmanian devils, Tasmanian devil => Tasmanian devil, animal
gull => gull, animal
flamingo, flamingos => flamingo, animal
anacondas, anaconda => anacondas, animal
ocelot, ocelots => ocelot, animal
```

Penguin, Penguins => penguin, animal
condors, condor => condor, animal
monkey, monkeys => monkey, animal
bird, birds => bird, animal
iguana, iguanas => iguana, animal
snail, snails => snail, animal
toucan, toucans => toucan, animal
lion, lions => lion, animal
llama, llamas => llama, animal
snake, snakes => snakes, animal
Tortoise, Tortoises => Tortoise, animal
parrot, parrots => parrot, animal
Booby, Boobies => Booby, animal
horse, horses => horse, animal
sea lion, sea lions => seal, animal

# sports
football => football, sport
surf => surf, sport
motorcycle => motorcycle, sport
race => race, sport

# people
fans, fan => fan, people

# water
river => river, water
lake => lake, water

# stone, rock
rock => rock, stone
brick, bricks => brick, stone

A.2 Query Expansion List

# countries
USA, United States of America, US

# synonym
church, churches, cathedral, cathedrals
oxidised, rusty
observing, watch, see
football, soccer
match, game
accommodation, room

# common sense
swimming => swimming in the water
drawings, drawing => drawing, line, petroglyph
prize, prizes => prize, medal
water => water, sea