MSRA Columbus at GeoCLEF 2006

Zhisheng Li[1], Chong Wang[2], Xing Xie[2], Wei-Ying Ma[2]

[1]Department of Computer Science, University of Sci. & Tech. of China, Hefei, Anhui, 230026, P.R. China
zsli@mail.ustc.edu.cn
[2]Microsoft Research Asia, 4F, Sigma Center, No.49, Zhichun Road, Beijing, 100080, P.R. China
{chwang, xingx, wyma}@microsoft.com

Abstract

This paper describes the participation of Columbus Project of Microsoft Research Asia (MSRA) in the Geo-CLEF 2006 (a cross-language geographical retrieval track which is part of Cross Language Evaluation Forum). For location extraction from the corpus, we employ a gazetteer and rule based approach. We use the MSRA's IREngine as our text search engine. Both text indexing and geo-indexing (implicit location indexing and grid indexing) are considered in our system. We only participated in the Monolingual GeoCLEF evaluation (EN-EN) and submitted five runs based on different methods, including MSRAWhitelist, MSRAManual, MSRAExpansion, MSRALocal and MSRAText. In MSRAWhitelist, we expanded the unrecognized locations (such as former Yugoslavia) to several countries manually. In MSRAManual, based on the MSRAWhitelist, we manually modified several queries since these queries are too "natural language" and the keywords of the queries seldom appear in the corpus. In MSRAExpansion, first we use the original queries to search the corpus. Then we extract the locations from the returned documents and calculate the times each location appears in the documents. Finally we will get the top 10 most frequent location names and combine them with the original geo-terms in the queries. However, this may introduce some unrelated locations. In MSRALocal, we do not use white list or query expansion method to expand the query locations. We just utilize our location extraction module to extract the locations automatically from the queries. In MSRAText, we just utilize our pure text search engine "IREngine" to process the queries. The experimental results show that MSRAManual is the best run among the five ones and then the MSRAWhitelist approach. MSRALocal and MSRAText perform similarly. The MSRAExpansion performs worst due to the introduced unrelated locations. One conclusion is that if we only extract the locations from the topics automatically, the retrieval performance does not improve significantly. Another conclusion is that automatic query expansion will weaken the performance. This is because the topics are too difficult to be handled and the corpus may be not large enough. Perhaps, the automatic query expansion may perform better in the web-scale corpus. And we find that if the queries are formed manually, the performance will be improved significantly.

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; H.2.3 [**Database Management**]: Languages – Query Languages

General Terms

Measurement, Performance, Experimentation

Keywords

Location name extraction and disambiguation, geographic focus detection, implicit location, geographic information retrieval.

1. Introduction

In common web search and mining, location information is usually discarded. However, people need to consider locations all the time, such as eating, traveling and shopping. The goal of the Columbus Project is to utilize the location information into web search and mining to better facilitate users to acquire the knowledge related to locations and suggest a location-aware way to organize the information.

The main technologies in Columbus Project are listed below:

- Location Extraction and Disambiguation – a fundamental problem of developing technologies for extracting and disambiguating locations from the documents, including texts and web pages;
- Geo-Indexing and Geo-Ranking – with extracted locations, developing indexing and ranking methods for search, such as spatial indexing and integrating text ranking into spatial ranking;
- Mining geographical knowledge – mining the relationships between common knowledge and location information;
- Visualization for search results or mining results – developing better technologies for visualizing the geo-information on map or virtual earth for easy browsing, navigating and information acquiring.

This is the first time for Columbus project to participate in GeoCLEF 2006 [7]. GeoCLEF is aimed at providing necessary platform for evaluating the geographic information retrieval. We only participated in the Monolingual GeoCLEF evaluation (EN-EN) and submitted five runs based on different methods.

The rest of the paper is organized as follows. We present a description of our system in section 2 and depict our monolingual GeoCLEF experiments (EN-EN) in section 3. Section 4 presents the experimental results and analysis. Conclusions and future work are discussed in Section 5.

2. Geographic Information Retrieval System Description

2.1 Geo Knowledge Base

We use internal geographic Database as our basic gazetteer. This gazetteer contains basic information about the locations all over the world, including location name, location type, location importance and the hierarchical relationship between the locations. We utilize this gazetteer to extract the locations, to disambiguate the locations and detect the focuses of documents. Besides this gazetteer, we also use some other resources to improve the performance, including stop word list, person name list, white list and location indicator list. The method to generate the stop word list is in 2.2.3. The white list and location indicator list is maintained manually, while the person name list is downloaded from the internet. And we integrated all these resources as a Geo Knowledge Base.

2.2 Location Extraction Module

Location Extraction module aims to extract locations and also disambiguate them from unstructured text. We manually composed the rules to address this task. Figure 1 is the workflow of the location extraction. It includes several parts: text parsing, geo-parsing, geo-disambiguating and geo-coding.

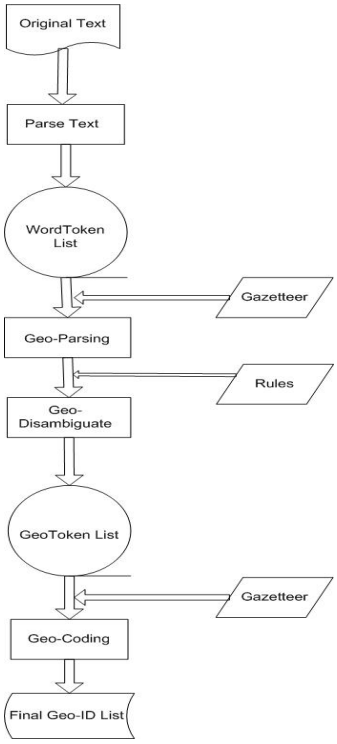


Figure 1. Workflow of location extraction

2.2.1 Text-parsing

In this step, we use a simple text-parsing method to parse the original text into a single-space separated word-token string. All useless tags and symbols are removed.

2.2.2 Geo-parsing

There are several ways to find the location candidates in this step. We use a gazetteer-based approach and only find the candidates that appear in the gazetteer. We follow the longest match principle. For example, if "New York" is detected, the substring "York" is not reported. In order to match the location names in the text with the ones in our gazetteer fast, we utilize a parsing-prefix table which is a list used to store the prefixes of the candidates of locations in the text. During the process, we need to maintain the parsing-prefix table until the longest location name is found. An example of location extraction can be seen in figure 2.

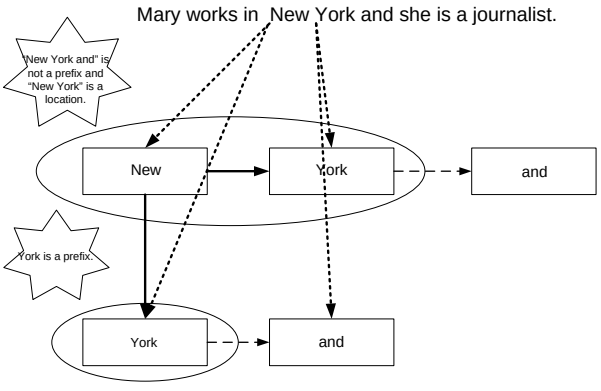


Figure 2. An example of location extraction. "New York" and "York" are prefixes in the prefix table. When the next word-token is "and", "New York and" is not a prefix and "New York" is a location, then the updating of prefix-table stops and "New York" returns as a location candidate.

For updating the parsing-prefix table, we need to create a location-prefix hash-table, which stores all the possible prefixes of the location names in the gazetteer, e.g. the prefix for "New York" are "New" and "New York". Then in the matching process, we just need to look up the word-token in the location-prefix hash-table to check whether it's a prefix of a location. If the word-token is a prefix, it indicates that a location may be found and we insert it into the parsing-prefix table. The geo-parsing algorithm is as follows.

Our algorithm scans the word token list from the beginning by gazetteer-lookup operation. Once a possible prefix is met, the parsing-prefix table will be created and the corresponding word token will be inserted into it. If the next word token will be a new prefix together with the existed prefixes in table, the prefix table will be updated. Otherwise, if a prefix in the table has been a location name, the update operation will stop and we will get the longest location name from the parsing-prefix table and transform it to a geo token, at the same time the prefix table will be cleared. Finally, the algorithm will stop when it meet the end of the word token list.

In this algorithm, we utilize the location-prefix table during the location-extraction process so that the text can be parsed linearly. The impossible location names, whose prefixes are not in the location-prefix hash-table, won't be looked up. So the dictionary-lookup operation times decrease significantly and the geo-parsing speed will be much faster than previous approaches.

2.2.3 Geo-Disambiguation

The ambiguity of our gazetteer is very serious, e.g. "is", "all" are also place names. The reason for disambiguating is that many different locations have the same name and there are some location names which are not used as locations, such as person names. We employ a rule-based approach and utilize stop word list, person name list and white list to disambiguate the candidates.

Our stop word list is generated from the corpus of the GeoCLEF 2006 data sets, which are collections of newspapers from LA Times (Year 1994) and Glasgow Herald (Year 1995). We applied following criteria to obtain the stop word list:

- Names that appeared too frequently, but always appeared as lower-case. Then these location names should be put into the stop word list, e.g. "is", "all".
- The location or its entity type is not very important, but it appears too frequently. It has a large probability to be non location word. Like "Leaf" is a river but in most of the cases, it should be a stop word.

Besides the automatic-generated stop words, we also maintain a manual stop word list. The manual stop word list contained the stop words that cannot be generated by the method described before.

The white list contained some historic names, organization names or other alias that do not exist in our gazetteer, like "Korea", "former Eastern Bloc", "former Yugoslavia", "Middle East". We map these words to multiple EntityIDs in our gazetteer, for example: Korea-> South Korea (ID: 134) + North Korea (ID: 131). Through the mapping, the white list can be treated nearly the same as the common items in the gazetteer, only lack of hierarchy relationships.

We also applied a confidence-based framework in the geo-disambiguation. We assign each location name a confidence score. First we run the rules in the following orders, and then combine the confidence scores in different steps.

- Stop word rule: we remove the location candidates if they appear in the stop word list.
- Location Indicator rule (xx means locations): such as xx city, and there is a rule list with corresponding confidence scores.
- Non-Location Indicator rule (mainly person indicator rules): such as Mr., and there is also a rule list with corresponding confidence scores.
- Complete Person Name rule: If the location candidate is a person name in the Person Name list, and from the context we find that there is a whole person name there, this location candidate is removed.
- Default Popular Person Name rule: If the location candidate is a very popular person name, and there are no positive rules to indicate it is a location, this candidate is removed.
- Default rule: If there are no strong rules to remove the location candidate, it will be assigned a moderate score according to population, importance and location hierarchical level.

2.2.4 Geo-Coding

In this step, after we disambiguated the location names, we need to assign the exact meanings to them. Many different places have same names, e.g. "London" is the capital in England, but also a place in United States of America. Figure 3 shows a

visual illustration of Geo-Coding. We utilize the heuristic methods to determine the exact meanings of the disambiguated locations.

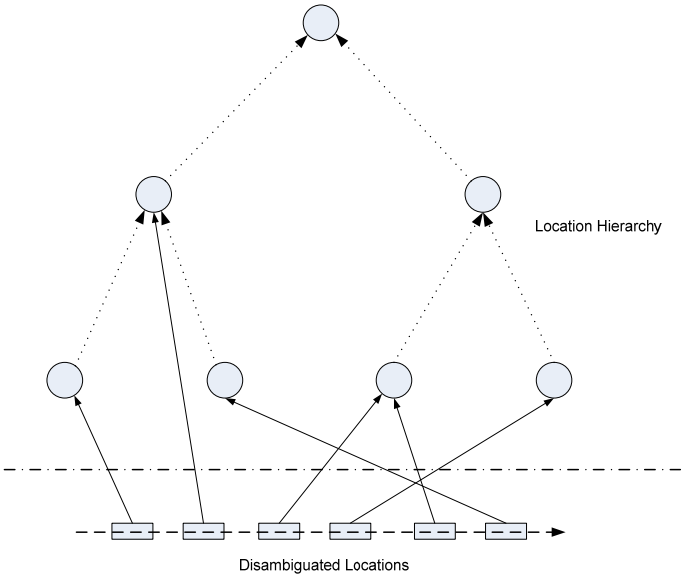


Figure 3. A visual illustration of Geo-Coding

2.3 Geographic Focus Detecting Module

When the locations' exact positions are determined, we want to get the focus of the documents. We adopted the algorithm described in [5]. Its main idea is to accumulate the score of each node in the hierarchical tree from bottom to up, and to sort all the nodes whose score are not equal to zero. Then we can get a list of focuses about the documents. The location with the biggest score is the most possible focus. For example, a document, mainly talking about Redmond economics, also has some relationship with Seattle.

2.4 Geo-Indexing Module

The indexing module of a GIR system usually composes of two parts: text index and geo-index. On the other hand, a geographical query usually consists of three parts: textual terms, spatial relationship and geographical terms. For example, one query in GeoCLEF 2006 is "Snowstorms in North America", where "Snowstorms" indicates what the user intends to know, "in" represents the spatial relationship and "North America" is the scope of the area that the user is interested in. We can retrieve a set of documents related to the textual terms (Snowstorms) through the text index and another set of documents whose geographical focuses are related to the query area through geographical index. These two sets will be merged and ranked according to a combined ranking function. In our system, we suppose the query location input is through text, not map.

In our system, explicit locations and implicit locations [9] are indexed together and different geo-confidence scores are assigned to them. The advantage of this mechanism is that no query expansion is necessary and implicit location information can be computed offline for fast retrieval.

In [6], the authors concluded that the most common geographical relationship used in queries is "in". Actually, if no spatial relationship exists in the query, we can safely assume the relationship is "in". Also, when a user wants to know information about "Independence movement in Quebec" (a query in GeoCLEF 2006), it means he/she wants to get the information covered by the query geographical area "Quebec", and would feel uncomfortable if a GIR system returns documents about "Independence movement in Canada" or "Independence movement in North America". Therefore, in our indexing algorithm, the implicit locations will not be expanded to lower levels because users usually don't need information about upper locations when they seek for information. When "Canada" appears in one document, the document ID shouldn't be appended to the inverted lists of "Quebec" or other children locations of "Canada". In order to obtain the implicit locations, we first adopt the focus detecting algorithm described in [5]. Afterwards we add the ancestors of these explicit focuses as implicit locations, but with lower confidence scores.

In our system, we adopt two types of geo-indexes: one is called focus-index, which utilizes the inverted index to store all the explicit, and implicit locations of documents (see Figure 4); the other is called grid-index, which divides the surface of the Earth into 1000 × 2000 grids. The documents will be indexed by these grids according to their focuses. The reason for adopting grid-index is that some topics in GeoCLEF 2006 can't be solved only by focus-index due to the spatial relationship other than "in" (such as "near").

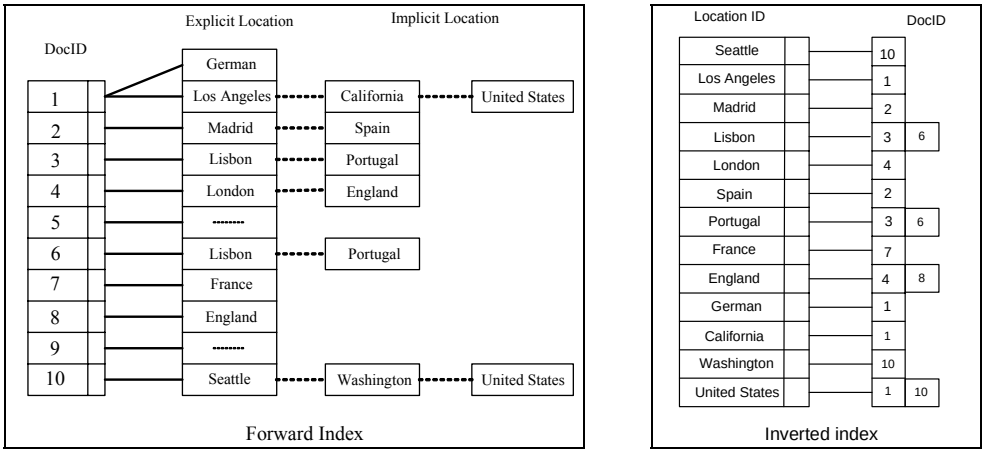


Figure 4. Inverted index of both explicit and implicit locations.

2.5 Geo-Ranking module

For the ranking module, we adopt IREngine, developed by MSRA, as our basic search engine. Then we integrated the geo-ranking module into it. The basic ranking function for text-ranking module is BM25. The ranking algorithm is as follows.

For focus-index, the matched docID list can be retrieved by looking up the locationID in the inverted index. For grid-index, we can get the docID list by looking up the grids that the query location covers. We first retrieve two lists of documents relevant to the textual terms and the geographical terms respectively, and then merge them to get the final results. A combined ranking function $R_{combined} = R_{text} \times \alpha + R_{geo} \times (1- \alpha)$, where R_{text} is the textual relevance score and R_{geo} is the geo-relevance score. Experiments show that textual relevance scores should be weighted higher than geo-relevance scores.

3. Monolingual GeoCLEF Experiments (English - English)

In table 1, we show all the five runs submitted to GeoCLEF. When the topic field is "Title", we just use the EN-title element of the topic to generate the query of the run. When the topic field is "Title + Description", this means that the EN-title and EN-desc are both used in the run. When the topic field is "Title + Description + Narrative", this means that EN-title, EN-desc and EN- narr are all used. And the "Description" field in Table 1 gives a simple explanation of the methods used in the runs. Priorities are assigned by us, where priority 1 is the highest and 5 the lowest.

Table 1. Run Information

Run-ID	Topic Fields	Description	Priority
MSRAWhitelist	Title	using geo knowledge base	1
MSRAManual	Title + Description	using geo knowledge base and manual query construction	2
MSRAExpansion	Title + Description	using query expansion	3
MSRALocal	Title	without geo knowledge base and query expansion	4
MSRAText	Title + Description + Narrative	using pure text	5

In MSRAWhitelist, we used the EN-title elements of the topics to generate the queries. For some special queries, e.g. "Credits to the former Eastern Bloc", "Arms sales in former Yugoslavia", "Eastern Bloc" and "former Yugoslavia" are not in our gazetteer, so we utilized the geo knowledge base to get the corresponding sovereigns of these geo-entities. Then we can make a white list manually for the geo-terms of these queries.

In MSRAManual, we generated the queries with EN-title and EN-desc elements of the topics. However for some queries, e.g. "Wine regions around rivers in Europe", "Tourism in Northeast Brazil", if we just input the "wine regions" or "Tourism", very few documents will return. So we manually modified the textual terms of such queries, e.g. "Tourism" -> "tourism tourist tour traveling travel".

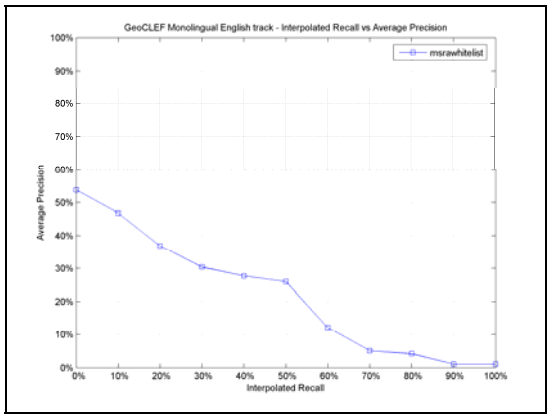
In MSRAExpansion, we generated the queries with EN-title and EN-desc elements of the topics. Different from MSRAManual, the queries are automatically expanded based on the pseudo-feedback technique. First we use the original queries to search the corpus. Then we extract the locations from the returned documents and calculate the times each location appears in the documents. Finally we will get the top 10 most frequent location names and combine them with the original geo-terms in the queries. For example, "former Yugoslavia" will be expanded to be "Croatia, United States, Russia, Sarajevo, Balkan, Edinburgh, Tuzla, Belgrade, Germany, Zagreb". Then we can see "Croatia, Sarajevo, Belgrade, Zagreb, Tuzla" are really parts of "former Yugoslavia", however, unrelated locations, such as "United States", also appeared. The reason is that they appeared together with "former Yugoslavia" frequently in the corpus and it's difficult to find a criterion to remove them.

In MSRALocal, we used the EN-title elements of the topics to generate the queries. And we do not use geo knowledge base or query expansion method to expand the query locations. We just utilize our location extraction module to extract the locations automatically from the queries.

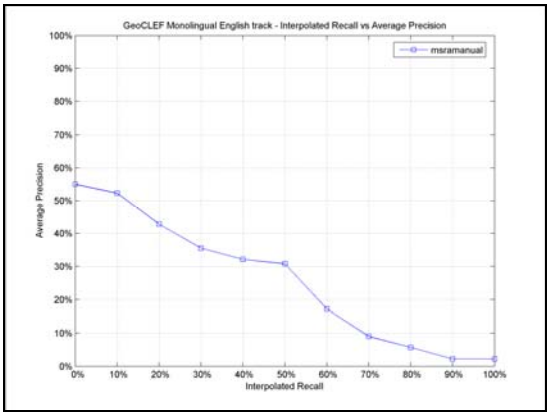
In MSRAText, we generated the queries with EN-title, EN-desc and EN- narr elements of the topics. We just utilize our pure text search engine "IREngine" to process the queries.

4. Results and Discussion

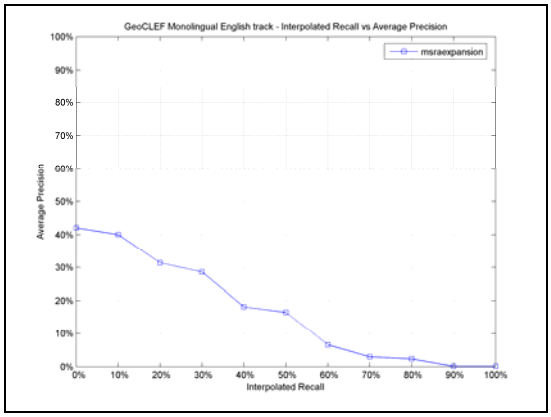
Figure 5 is the interpolated Recall vs Average Precision for the five runs. MSRAManual is the best run among the five ones as expected, because the queries are modified manually. The worst one is MSRAExpansion. The reason is that many unrelated locations are added to new queries after pseudo feedback for some topics.



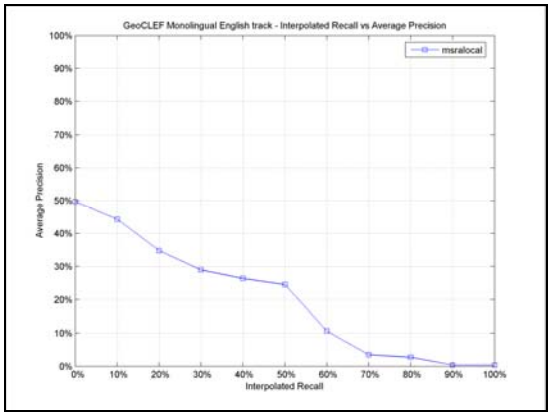
MSRAWhitelist



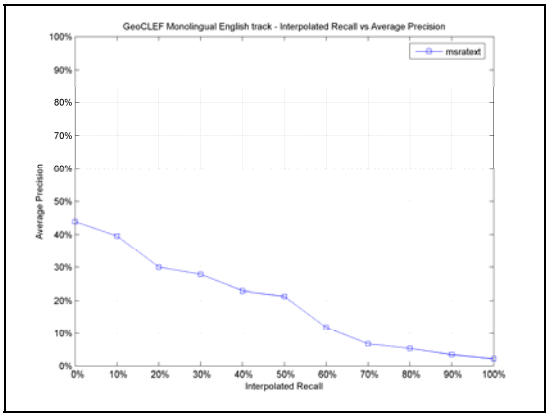
MSRAManual



MSRAExpansion



MSRALocal



MSRAText

Figure 5. The Interpolated RECALL vs AVERAGE PRECISION for all five runs

Table 2 shows the MAP and R-Precision for all the runs. MSRAManual outperforms the other runs in both MAP and R-Precision. The performances for MSRALocal and MSRAText are very similar to each other. It indicates that only extracting locations from the topics as geo-terms without any expansion won't improve MAP much. However, MSRAWhitelist outperforms than MSRALocal by about 9% in MAP. The reason is that we only made white list for several topics and it won't affect the MAP much.

MSRAManual improves the performance significantly compared with MSRALocal. For example, for the GC045 "Tourism in Northeast Brazil", the MAP in MSRALocal is 0.0%, but in MSRAManual it's 26.96%. This is because, as discussed in Section 3, very few documents contain the keyword "Tourism". We manually modified "Tourism" to "tourism tourist tour traveling travel" and improved the performance of this query significantly. For the GC037 "Archeology in the Middle East", the MAP is from 6.45% to 27.31%, and the reason is similar to that of GC045.

The MAP of MSRAExpansion drops a little compared with MSRALocal, since the expansion introduced unrelated locations. Perhaps, if we have a large corpus, the performance of the expansion approach would improve, since the top-ranked documents might include more relative locations.

Table 2. MAP & R-Precision for five runs

RUN-ID	MAP	R-Precision
MSRAWhitelist	20.00%	23.52%
MSRAManual	23.95%	25.45%
MSRAExpansion	15.21%	18.53%
MSRALocal	18.37%	22.45%
MSRAText	18.35%	21.23%

5. Conclusions

One conclusion is that if we only extract the locations from the topics automatically, the retrieval performance does not improve significantly. Another conclusion is that automatic query expansion will weaken the performance. This is because the topics are too difficult to be handled and the corpus may be not large enough. Perhaps, the automatic query expansion may perform better in the web-scale corpus. And we find that if the queries are formed manually, the performance will be improved significantly.

In the future we will improve the system in the following aspects: 1) geographical indexing and ranking; 2) automatic query processing; 3) geographical disambiguation algorithm; 4) visualization for the results.

6. Reference:

[1] E. Amitay, N. Har'El, R. Sivan and A. Soffer. Web-a-where: Geotagging Web Content. SIGIR 2004.

[2] Y.Y. Chen, T. Suel and A. Markowitz. Efficient Query Processing in Geographical Web Search Engines. SIGMOD'06, Chicago, IL, USA.

[3] B. Martins, M. J. Silva and L. Andrade. Indexing and Ranking in Geo-IR Systems. GIR'05, Bremen, Germany.

[4] A.T. Chen. Cross-Language Retrieval Experiments at CLEF 2002. Lecture Notes in Computer Science 2785, Springer 2003.

[5] C. Wang, X. Xie, L. Wang, Y.S. Lu and W.Y. Ma . Detecting Geographical Locations from Web Resources. GIR'05, Bremen, Germany.

[6] M. Sanderson and J. Kohler. Analyzing Geographical Queries. GIR'04, Sheffield, UK.

[7] GeoCLEF 2006. http://ir.shef.ac.uk/geoclef/

[8] C.B. Jones, A.I. Abdelmoty, D. Finch, G. Fu and S. Vaid. The SPIRIT Spatial Search Engine: Architecture, Ontologies and Spatial Indexing. Lecture Notes in Computer Science 3234, 2004.

[9] Z. S. Li, C. Wang, X. Xie, X. F. Wang and W.Y. Ma. Indexing implicit locations for geographic information retrieval. GIR'06, Seattle, USA.