THUSAM at NTCIR-11 IMine Task

Cheng Luo, Xin Li, Alisher Khodzhaev, Fei Chen, Keyang Xu, Yujie Cao, Yiqun Liu, Min Zhang, Shaoping Ma
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
c-luo12@mails.tsinghua.edu.cn

ABSTRACT

This paper describes our approaches and results in NTCIR-11 IMine task. In 2014, we participate in subtasks for Chinese/English Subtopic Mining and Chinese Document Ranking. In Subtopic Mining subtask, we mine subtopic candidates from various resources and construct the subtopic hierarchy with several different strategies. In Document Ranking subtask, we rerank the result lists with HITS algorithm and then adopt a pruning exhaustive search algorithm to generate diversified result lists.

Team Name

THUSAM

Subtasks

Subtopic Mining (Chinese, English),
Document Ranking (Chinese)

Keywords

query intent, subtopic mining, document ranking

1. INTRODUCTION

In NTCIR-11, THUSAM group participated in IMine task, including subtopic mining subtask (Chinese and English) and Document Ranking subtask (Chinese).

1.1 Subtopic Mining

For English subtopic mining, we first mine subtopic candidates from different resources and filter the candidates with a sequence of filters. Thus, each candidate is represented as a TF-IDF vector. To mix candidates from different resources, we propose a normalized ranking framework to merge candidates into a single ranked list. Then we cluster the candidates to construct subtopic hierarchy with two different strategies: Bottom-Up approach and Top-Down approach. Bottom-Up method first represents each candidate as a single cluster and then merges clusters to decrease the number of clusters. Top-Down method focuses on combining mined candidates into existing topic hierarchy from Wikipedia.

For Chinese subtopic mining, we propose a 3-step framework to construct topic hierarchy: (1) we extract subtopic candidates from query suggestion, Wikipeida disambiguation items, query facets, similar queries calculated by 'Query2vec' algorithms and some other resources. A Learning-to-Rank algorithm is adopted to rank the subtopics from different resources to pick out the candidates with high quality. (2) We adopt several different algorithms to construct the subtopic hierarchy. The algorithms can be classified into 3 categories: the first approach is called Top-Down algorithm, which means we first organize the First Level Subtopic (FLS) and then classify each candidate into FLSs as Second Level Subtopic (SLS); the second approach is called Bottom-Up algorithm which indicates that we first cluster candidates into groups as SLSs and try to name each cluster with different methods; the third approach called Knowledge Base Aided algorithm organizes the disambiguation items and index items as FLSs and classifies all candidates into FLSs using similar method as Top-Down approach. (3) Moreover, some re-ranking algorithms are used for different runs. The most important re-ranking approaches are based on clicked snippets from search engines. For each topic, we count the frequency of the terms which appear in the clicked search result snippet, and promote the subtopics with high-frequency terms. Previous evaluation results show that our method has improved the D-nDCG [5] values of topics.

1.2 Document Ranking

Given an ambiguous or underspecified query, search result diversification aims to produce a Search Engine Result Page (SERP) that maximizes the probability of satisfying different users' information needs (named subtopics or sub-intents). However, as a maximum coverage problem, search result diversification proves to be NP-hard by previous studies. Several greedy search algorithms such as IA-Select [1] and xQuAD [6] are therefore proposed to find an approximation of the optimal diversified ranking list. Meanwhile, to evaluate diversified search performance, we also need the ideal ranking list as a comparing reference (e.g. to normalize the scores in α-NDCG). Finding the ideal list with known true relevance to different subtopics is also NP-hard in a similar way to search result diversification. Therefore, most studies in diversified search evaluation also use greedy search to generate the ideal list. In both cases, greedy search can generate suboptimal solutions, as it is known to be able to find local optima only.

For example, assume that there are three documents that are all relevant to two subtopics (which have equal importance scores) underlying a particular query. Their relevance scores for the subtopics are shown in Table 1. If we choose the commonly-used weighted sum of document gains for subtopics as evaluation metrics, we can see that greedy search fails to produce optimal results: If the greedy search

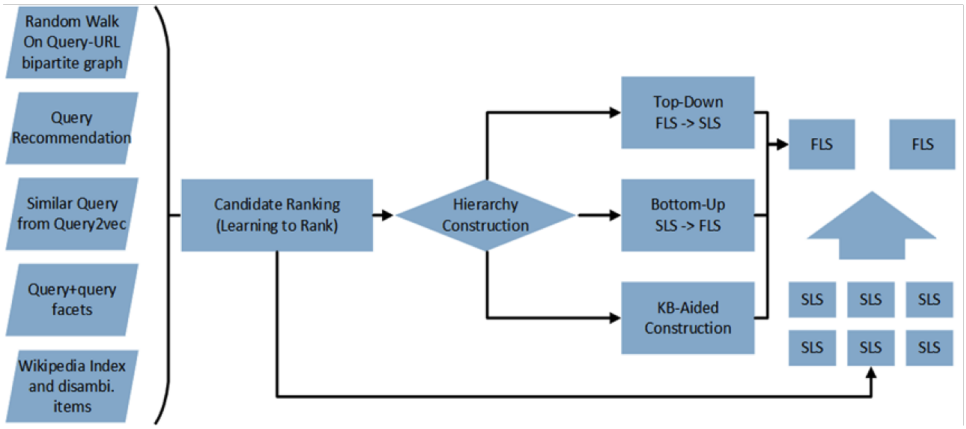

Figure 1: Framework overview for Chinese Subtopic Mining

algorithm is required to produce a list with exactly two documents, either {a, b} or {a, c} will be returned because the diversified gain of a (0.6) is larger than that of b or c (0.5) at the first ranking position. However, if we list all possible result lists with an exhaustive search strategy, we can find better lists b, c and c, b which generate the largest possible weighted sum of document gains. This example shows that the greedy strategy cannot always produce an optimal result list. However, exhaustive search is untractable for online Web search because of its extremely low efficiency.

Table 1: An example for diversified ranking problem.

Document	Subtopic 1 (0.5)	Subtopic 2 (0.5)
a	1	0
b	0	1
c	0.6	0.6

In the Document Ranking subtask, to improve the performance of diversified ranking/evaluation, we adopt a novel search algorithm that can produce better lists than greedy search, while limiting the complexity within a reasonable range. We observe that the NP-hard problem in diversified searches is mainly caused by the fact that a document may be relevant to more than one subtopic. In practice, a large proportion of relevant documents are relevant to only one subtopic. Therefore, a large part of candidate documents can be easily put into several sets of ordered pairs reflecting their relative orders in the optimal diversified result list. Rather than performing a diversified search on all possible ordered lists, a severe pruning will remove all the lists contradicting the determined ordered pairs. This is the idea used in our Pruned exhaustive search algorithm[2]. To further decrease the complexity of the algorithm, we propose a Key Slots search strategy to limit the number of candidate documents based on the fact that most users only focus on the top slots in search result lists. We also introduce a Search Window strategy to divide a large search result space into several smaller search windows, which could be considered as a trade-off between efficiency and effectiveness.

2. ENGLISH SUBTOPIC MINING

2.1 Candidates Mining

2.1.1 *Candidates From Various Resource*

To reveal every concept of a query, we use many resources and develop fusion strategy to combine candidates by multiplying factor of importance and incorporate different resources into a ranked list. In our work, we use similar fusion approach as described in [7].

Most massive set of candidates are obtained from contents of specific HTML tags: for example, *"<title>, <h1>, <h2>, <h3>, <a>"* from top 50 results for each query and its query suggestions (Bing, Google, Yahoo).

In addition to traditional utilization results of commercial search engines (Bing, Google, Yahoo), we further involve:

- Query Completion (Bing, Google, Yahoo)
- Query Suggestion (Bing, Google, Yahoo)
- Google Adwords Keyword Planner
- Wikipedia

2.1.2 *Candidate's Filtering*

All mined candidates are cleaned by a sequence of filters. Each filter focuses on a specific aspect of the candidates. For example, the first filter will remove all the stopwords in articles, the most common English words, question words etc. Also, we adopt stemming methods on candidates to calculate semantic similarity with the help of WordNet Library [3].

We have removed all candidates that do not contain the original query and the ones which contain only the original query. For the term which is shorter than 4 characters in the original query, the filter is extended to match the regular expression describing the possible variations of words formed from letters as abbreviation in case that the stemmed term has more than one extension form.

2.2 Candidate's Presentation

After filtering of each candidates, we remove all the terms which come from the original query. This is based on the assumption that the terms coming from original query carry little additional information and would be noisy when calculating similarity between different candiates.

For example *'Apple tree'* and *'apple store'* has a Jaccard coefficient of 0.33. However, queries *'apple'*, *'Apple tree'* and *'apple store'* should belong to different subtopics. All the candidates are represented as a TF-IDF vector while calculating similarity.

2.3 Resources Based Ranking

We calculate a normalized score for each candidate to merge candidates from different resource into a single list, the normalization strategy is stated as follows:

- For candidates from Google Adwords, we normalize the score of a specific candidate from Google Adword Keyword Planner[1] within other candidates from the same query. In our approach, the score R can be calculated as,

$$R = \frac{c}{2^i} \tag{1}$$

where c is a constant (in our experiment we set $c = 1$) and i denotes to the ranking in Google Adword Keyword Planner.

- For candidates from Wikipedia disambiguation page, the score is calculated as:

$$R = \frac{1}{NS + NH + p} \tag{2}$$

where NS is the total number of subtopics from a disambiguation page; NH is the total number of headers (extracted from html tag *<h2>*) from the disambiguation page's subpages or direct pages; p denotes the candidate's ranking in the page.

- For candidates from the search query, the score can be calculated as following:

$$R = \frac{1}{R_{SE} + T_{source}} \tag{3}$$

where R_{SE} denotes the search engine rank of the related page, $T_{resource}$ represents tag's fine where $T_{title} = 1$, $T_{h1} = 2$, $T_{h2} = 8$ and $T_a = 32$ in our experiment.

We gather all the candidates from different resources and normalize the weight R as the final score of the candidates. This strategy helps us to rank candidates even if it comes from different resources.

2.4 Candidate Subtopic Clustering

There are many duplicates among candidates mined from different resources which carry similar or the same meaning. For example, although *'cherry actors'* and *'cherry cast'* are different literally, they actually belongs to the same subtopic *'cherry movie'* for query *'cherry'*. Thus, it is necessary to cluster candidates into *subtopic clusters*.

Because the candidates are usually short and vague, the snippet information provided by search engines helps us enrich information about the candidates. For each subtopic candidates, we first submit each one to the search engines (Google, Bing, Yahoo) and crawl the top 50 results' snippet information to form an extended document. Then we calculate the Jaccard similarity coefficient between different documents:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4}$$

Where A and B are the two different term frequencies vectors of the two subtopics to compare. We extend this coefficient by considering both the words and their frequencies. (So even if many words retrieve for the two subtopics are the same, but their frequencies is really different, then their similarity will be reduced). We implement this feature because we think that both words retrieved and their frequencies are important feature to know if two subtopics have similar intent or not. So when we calculate the intersection or the union of A and B, we add the average score of their frequencies:

$$J_{ext}(A, B) = \frac{\sum_{i \in A \cap B} \frac{f_{A_i} + f_{B_i}}{2}}{\sum_{i \in A \setminus B} f_{A_i} + \sum_{i \in B \setminus A} f_{B_i} + \sum_{i \in A \cap B} \frac{f_{A_i} + f_{B_i}}{2}} \tag{5}$$

Thus, we create a clustering algorithms using this extended Jaccard Similarity.

2.4.1 Bottom-Up approach

We name it as Bottom-up strategy because we try to decrease the number of clusters by merging similar clusters step by step. It is similar with agglomerative clustering, but we are not going to finish with a single cluster. We propose a two-step strategy: first we represent each candidate as single cluster, then we start to merge similar clusters to decrease the number of clusters and combine candidates definitely related to one topic.

Algorithm 1 Bottom-up hierarchical clustering algorithm with extended Jaccard similarity coefficient

```
Select k (define experimentally)
Create for every subtopic candidate a cluster
for each cluster do
    for each remaining cluster do
        if J_ext similarity of the the two clusters > k then
            Merge clusters
        end if
    end for
end for
Repeat 3 while the similarity between two clusters is above k.
Select j (j < k gap between is defined experimentally)
for each cluster do
    for each remaining cluster do
        if J_ext similarity of the two clusters ≥ k then
            Merge cluster
        end if
    end for
end for
```

2.4.2 Up-Bottom approach

In Up-bottom approach, the main idea that to combine with already existing topics tree (for example, semantic structure from Wikipedia) with mined candidates. If there is a new candidate which does not appear in the existing

[1]https://adwords.google.com.sg/KeywordPlanner

topic hierarchy, we can add it as first level subtopic on the topics tree. The algorithm are formalized in Algorithm 2.

In experiment, we crawl and parse the disambiguation pages from Wikipedia and then merge existing hierarchical structure with the subtopic mined from different resources.

Algorithm 2 Up-bottom hierarchical mixed-classification-clustering algorithm with extended Jaccard similarity coefficient

```
Select k (learnt from training set)
Create for hierarchy cluster based on Wikipedia data
for each cluster do
    if J_ext similarity of the the two clusters > k then
        combine with cluster
    end if
    get best J_ext similarity among all clusters
    if best J_ext < k then
        create new clusters for this candidate on first level
    else
        combine with cluster
    end if
end for
```

2.5 Submitted Runs

We apply methods described above to produce these runs for the English subtopic mining:

- **THUSAM-S-E-1A:** Extraction from multiple resources (all) + tuned bottom-up hierarchical clustering
- **THUSAM-S-E-2A:** Extraction from multiple resources + up-bottom approach

To evaluate our techniques, we make and submit 2 runs. The H-score, F-score, S-score and H-measure D values of the results are shown in Table 2. We can see that THUSAM-S-E-1A performs better in terms of H-score.

We do not expect THUSAM-S-E-1A to perform better. Instead, we have expected THUSAM-S-E-2A, because in our assumption was what hierarchical structure which came from Wikipedia should be better than the hierarchy built by clustering. The official overview shows that the differences among these submitted runs are statistical significantly. This might because that there is a gap between disambiguation items and user intents of a specific query. For example, in Wikipedia, *'cherry'* also means a geographical name in Illinois, however, few people search *'cherry'* for information of that place, most of them search *'cherry'* for information about keyboard or the fruit.

Table 2: Evaluation results for Subtopic Mining English

	H-score	*F-score*	*S-score*	*H-measure*
THUSAM-S-E-1A	0.8065	0.5179	0.4835	0.4380
THUSAM-S-E-2A	0.2634	0.4361	0.4732	0.1203

3. CHINESE SUBTOPIC MINING

3.1 Candidates From Various Resources

We observed that over the internet, there are many services that we can use to help us to disambiguate a query. Indeed, from these resources, we can extract sub-intents of an ambiguous query as well as, for some, interesting information about the sub-intents popularity. The resources we used include: (1) similar queries calculated by random walk algorithm on Query-URL bipartite graph; (2) query recommendation from search engines; (2) similar queries calculated by 'Query2vec' algorithm; (4) query facets offered by NTCIR official, which are extracted from Webpages based on some templates; (5) Wikipedia Indexes and Disambiguous items.

Among all these resources, the query recommendations supplied by search engines are usually of high quality, which means the candidates are relevant to the original. However, the candidates generated by random-walk, 'Query2vec', and query facets usually contains a lot of noise. We need to choose the high-quality candidates via a ranking or filtering process.

3.2 Candidate Ranking with Learning to Rank

We have collected enough subtopic candidates for each query from different external resources, but some of them are of low quality. For example, because the click-through bipartite is highly connected, random walks may result in irrelevant queries: there may exist paths between two completely unrelated queries or URLs. As a result, we need to rank the subtopic candidates for each query according to their correlation and only reserve the candidates with high quality. We adopt learning to rank algorithm to rank the candidates. We use the evaluation results of NTCIR-10 Intent-2 task as training set. In the evaluation results, the subtopics are sorted according to their relevance with the query. So we use the subtopics and their rankings to train the learning to rank models. We use two types of features as below, which represent the similarity between the query and the candidates:

- Text Similarity: length difference, Jaccard similarity, edit distance between the query and the candidate...
- Search Result Similarity: number of shared results in the SERPs of the query and the candidate...

Since we need to submit 5 first level subtopics for each query and 10 second level subtopics for each first level subtopic, so we choose NDCG@50 as the metric to optimize when training. We use 5-fold cross validation and compare the performance of different learning to rank algorithms. The results are shown in Table 3.

From the table we can see that RankBoost performs the best on the testing set, so we choose RankBoost algorithm to train the model and use it to predict the relevance between the candidates and the queries of IMine. We set a threshold θ and reserve the candidates with a score higher than θ. Filtering out candidates of low quality contributes to both classification and clustering, which benefits the construction of two-level subtopic hierarchy.

3.3 Bottom-Up FLS Construction

The main idea of bottom-up FLS construction is to cluster the candidates first and extract n-grams from the titles of

Table 3: Learning to rank results for candidate ranking

Method	NDCG@50 on training set	NDCG@50 on testing set
MART	0.8012	0.6951
RankNet	0.683	0.6675
RankBoost	**0.743**	**0.7303**
AdaRank	0.7049	0.7034
Coordinate Ascent	0.7037	0.676
LambdaMART	0.8274	0.688
ListNet	0.6912	0.6959
Random Forests	0.7896	0.6981

SERPs of the candidates in each cluster. Then we name each cluster with the following two methods. 1. Choose the shortest n-gram that matches a candidate in the cluster. 2. Rank the n-grams with learning to rank and choose the n-gram with the highest weight for each cluster.

3.3.1 Clustering Candidates

First we build feature vector for each candidate. We extract the SERPs of all the candidates from a Chinese commercial search engine and split the titles and snippets of the search results into words. We use TF-IDF of words as the features of the candidates. Since there are about 8000-10000 unique words in the SERPs of the candidates for each query, so the dimension number of feature vectors is about 8000-10000. We try different ways to compute the features. For TF, we try boolean frequency, logarithmically scaled frequency and augmented frequency.[2] For attenuation of the search results in SERP, we try linear attenuation (from 1 down to 0.5 evenly) and 1/r attenuation (divided by the ranking of the result). After computing the feature vectors for the candidates, we cluster the candidates with k-means. Since we need to submit 5 FLS, we choose k as 6 because we believe that some of the candidates cannot be assigned to any clusters and we can group them together and filter them out. We use the evaluation results of Intent-2 as training set. We define the precision of clustering as the ratio between the number of correctly clustered SLS pairs and all the SLS pairs. The highest precision of our methods is 81.6%.

3.3.2 Naming Clusters

We name clusters through extracting n-grams from the titles of SERPs of the candidates in the cluster. We believe that a subtopic is a refinement and complement of the query and its length should be larger than that of the query, so we range n from query.length + 1 to query.length + 10. We reserve the n-gram with the highest frequency for each length so we obtain 10 FLS candidates for each cluster. We adopt two methods to choose FLS from the n-grams. In the first method, we choose the shortest n-gram that matches a candidate in the cluster in the assumption that it can represent the cluster. The advantage of this method is that since the chosen FLS is also a SLS, it is definitely readable. While the disadvantage is that in a cluster, there is not necessarily a SLS that can represent the cluster. In the second method, we use learning to rank to obtain FLS. We use the ground truth of Intent-2 as training set. In the evaluation results, we use all the FLS as positive set and all the SLS not chosen to be FLS as negative set. The features we use are text similarity and search result similarity between the subtopic and the query as before. Since we need to select 5 FLS, so we choose P@5 as the metric to optimize. We try different learning to rank algorithms and the results are shown in Table 4.

[2]http://en.wikipedia.org/wiki/Tf-idf

Table 4: Learning to rank results for naming clusters

Method	P@5 on training set	P@5 on testing set
MART	0.872	0.4758
RankNet	0.3855	0.388
RankBoost	0.4995	0.4593
AdaRank	0.4745	0.453
Coordinate Ascent	0.5277	0.4868
LambdaMART	**0.9317**	**0.5196**
ListNet	0.3886	0.3935
Random Forests	0.8062	0.5144

From the table we can see that LambdaMART performs the best on the testing set, so we choose it to train the model and use it to predict the n-grams of IMine. We choose the n-gram with the highest score as FLS for each cluster. The advantage of this method is that the chosen n-gram is representative for the cluster. But since the chosen n-gram may not match a SLS in the cluster, it is not necessarily readable.

3.4 Top-Down FLS Construction

We can construct FLS via Bottom-Up methods as stated before, however, one of the key challenges in Bottom-Up construction is to name the clusters. Sometimes the n-gram naming method make mistakes, which means that the name is not readable. To solve the readability problem, we proposed a Top-Down FLS construction method. The main idea is to select representative query candidates as FLSs based on both quality and diversity.

During the selecting process, we have to consider two factors when choosing a query. The first factor is that the query should be a high-quality candidate and representative for the other candidates which indicates the same subtopic. The second factor is the novelty compared with the queries which are already chosen, which depends on the previous decisions. Thus, the chosen problem can be reduced to max-cover problem, which is NP-hard.

We design a heuristic greedy select algorithm considering both the relevance and novelty. The training process can be presented as follows.

The performance is evaluated by pairwise error rate on INTENT-2. Thus, we can get a group of parameters which get best performance,which is $a = 0.54, b = 0.28, c = 0.18, d = 0.0$ and the pairwise error rate is as low as 0.144. The application process can be described as follows:

By adopting this process on IMine data, we can get FLSs for each query. The major limitation of Top-Down construction method is that sometimes, the most representative queries may not in our candidate. For example, for query 'samsung projector', 'epson projector', 'sanyo projector' etc, the most representative subtopic is 'projector band', but it is very possible that it is in none of our candidate resource.

3.5 Knowledge Base Aided FLS Constuction

We find that among all the 50 Chinese queries, there are 37 queries which has a corresponding page on Wikipedia

Algorithm 3 *select_train*$(n, Cand, Chosen)$

Require: The number of candidates going to choose n;
The list of candidates which are not chosen yet, C;
The set of candidates which are already chosen,$Chosen$;
Parameters $a, b, c, d \in [0, 1]$, and $a + b + c + d = 1$
Ensure: Parameters a,b,c,d with best performance on Training Set (INTENT-2)

while $n > 0$ **do**
  for all $item \in C$ **do**
    $Score_{item} = a * (1 - AvgSimilarity_{item,chosen}) - b * (item.length/query.length) + c * Rel(item, query) + d * log(item.frequency)$
  end for
  $item_{best} = item$ with highest score
  $Cand$.remove $(item_{best})$
  $Chosen$.remove $(item_{best})$
  $select\_train(n - 1, Cand, Chosen)$
end while

Algorithm 4 *select_application*$(n, Cand, Chosen)$

Require: The number of candidates going to choose n; The list of candidates which are not chosen yet, $Cand$; The set of candidates which are already chosen, $Chosen$, initially, $Chosen$ is empty;
Ensure: $Chosen$ with n candidates in it.

while $n > 0$ **do**
  for all $item \in Cand$ **do**
    $Score_{item} = 0.54 * (1 - AverageSimilarity_{item,chosen}) + 0.28 * (-item.length/query.length) + 0.18 * Relevance(item, query) + 0.0 * log(item.frequency)$
  end for
  $item_{best} = item$ with highest score
  $Cand$.remove $(item_{best})$
  $Chosen$.remove $(item_{best})$
  $select\_train(n - 1, Cand, Chosen)$
end while

or Baidu Baike[3]. Contents on these pages are usually contributed by user crowds and well-organized. For example, for query *'bathtub'*, there are several subtitles on the page from Baidu Baike including: bathtub history, bathtub material, bathtub shopping instructions etc. Thus, we can use these subtitles or disambiguation items as FLSs and then classify other candidates into these FLSs. We name this method as Knowledge Base Aided FLS construction.

3.6 Second Level Subtopic Classification

In the Top-Down hierarchy construction process, after we get FLS subtopics with greedy select algorithm, we have to classified other SLSs into these FLSs. A lot of features between FLS and SLS are extracted for classification:

- Text Similarity (With different weight Attenuation model)
- Shared result number on different Search Result List length (10/20/50/100)

[3]A famous Chinese online encyclopedia (http://baike.baidu.com)

There are 75 features involved in this model and we train a linear regression model (L1-regularized L2-loss support vector classification) and achieve pairwise accuracy 0.59.

3.7 Evaluation Results

Based on the previous described methods, we make different combinations of the methods and submit 5 runs. The descriptions of the 5 runs are as below:

- *THUSAM-S-C-1A*: Cluster the candidates, find n-grams for each cluster, choose the shortest n-gram that matches a candidate in the cluster as its first level subtopic.
- *THUSAM-S-C-2A*: Cluster the candidates, find n-grams for each cluster, choose the n-gram with the highest weight using learning to rank algorithm as its first level subtopic.
- *THUSAM-S-C-3A*: Pick out the wiki disambiguation items as first level subtopics and make classification.
- *THUSAM-S-C-4A*: Pick out the first level subtopics with a greedy selection algorithm and make classification.
- *THUSAM-S-C-5A*: Select first level subtopics with an n-gram learning to rank algorithm and make classification.

The mean values of H-score, F-score, S-score and H-measure are shown in Table 5.

Table 5: Experimental results of Chinese subtopic mining runs

Runtag	*H-score*	*F-score*	*S-score*	*H-measure*
THUSAM-S-C-1A	**0.5527**	**0.5537**	**0.4634**	**0.2773**
THUSAM-S-C-2A	0.4347	0.4498	0.4633	0.2204
THUSAM-S-C-3A	0.3284	0.3744	0.3981	0.1400
THUSAM-S-C-4A	0.3284	0.3744	0.3993	0.1404
THUSAM-S-C-5A	0.4287	0.5040	0.4626	0.2224

From the results we can see that bottom-up FLS construction methods achieve higher H-measures than top-down FLS construction methods. This is reasonable because top-down FLS construction methods largely depend on the selection of first level subtopics. If the selected first level subtopics are not diverse enough, both F-score and S-score will be affected, thus decreasing Hmeasure. While bottom-up FLS construction methods ensure the diversity of first level subtopics through the clustering of second level subtopics.

4. CHINESE DOCUMENT RANKING

4.1 Retrieval Models and Dataset

In the retrieval step, we adopt the same improved probabilistic model and the same retrieval strategies as the ones we used in NTCIR-9 for document ranking. All the retrieval processes are conducted on SogouT-08 dataset and train different parameters are applied in these model or retrieval strategies. These parameters are determined and shown in Table 6.

Table 6: Parameters in retrieval models

part	α_1	k_1	b	ω
Content	0.2	0.6	0.35	0.2
Anchor	0.1	1.6	0.3	0.5
Click	0.1	1.4	0.55	0.3

4.2 Result re-rank with HITS

We adopt HITS[4] to re-rank the baseline search results in the Document ranking subtask, which is similar with our approach in INTENT-10. The main idea that we will re-rank the documents, which are the m-the biggest of either Authority or Hub values, up to the front. Top m documents sorted by either Authority or Hub Value are placed up to the front. Its new rank is determined as follows:

$$R_{new} = R_{old} - R_{old} \times (Authority + Hub) \quad (6)$$

where R_{new} stands for the new rank of the document, and R_{old} is the old one. The new rank is determined by three factors: the original rank, the Authority and the Hub values of the document. The m is set to 40 according to the training results in the TREC 2009 and TREC 2010 diversity task, because top 40 is a stable choice for ERR- IA value. Previous study[8] proved that HITS can stably improve the diversity of the searching result on both the TREC-based dataset and the SogouT dataset.

4.3 Pruning exhaustive search method

Given a list of possible subtopics (or sub-intents) for a particular ambiguous or broad query, both search result diversification in document ranking and the ideal list generation in diversity evaluation can be cast as a maximum coverage problem, which has proven to be NP-hard.

We observe that the NP-hard problem in diversified searches is mainly caused by the fact that a document may be relevant to more than one subtopics. In practice, a large proportion of relevant documents are relevant to only one subtopic: looking into the TREC Web Diversity and NTCIR Intent datasets, we can observe that only about 27% of relevant documents are relevant to more than one subtopics. Therefore, a large part of candidate documents can be easily put into several sets of ordered pairs reflecting their relative orders in the optimal diversified result list. Rather than performing an exhaustive search on all possible ordered lists, a severe pruning can take place, which removes all the lists contradicting the determined ordered pairs.

Based on these observations, we adopted a Pruned exhaustive search algorithm [2] on documents to generate a diversified search result list for each query.

With this algorithm, for each query, we generate a diversified result list based on the subtopic mined with subtopics constructed with Top-Down strategy and N-gram learning-to-rank list.

4.4 Runs and Results

In document ranking subtask, we submit 4 runs with combination of different window size and subtopic mining results.

- **THUSAM-D-C-1A:** Exhaustive search with window size 4. The SM result is from Subtopic N-gram Learning to rank list.
- **THUSAM-D-C-1B:** Exhaustive search with window size 5.The SM result is from Subtopic N-gram Learning to rank list.
- **THUSAM-D-C-2A:** Exhaustive search with window size 4.The SM result is from heuristic greedy select from subtopics.
- **THUSAM-D-C-2B:** Exhaustive search with window size 5.The SM result is from heuristic greedy select from subtopics.

The results of course-grain and fine-grain D#-nDCG over all query topics are shown in Table 7.

Table 7: Experimental results of Chinese document ranking runs

	Coarse-grain results (evaluated with first-level subtopics)	Fine-grain results (evaluated with second-level subtopics)
THUSAM-D-C-1A	**0.6965**	**0.6127**
THUSAM-D-C-1B	0.6943	0.6106
THUSAM-D-C-2B	0.3697	0.2711
THUSAM-D-C-2A	0.3502	0.2623

From the results we can see that both runs with subtopic N-gram learning to rank list perform significantly better than subtopic results from heuristic greedy select, which confirms the subtopic mining results that N-gram learning to rank based subtopic mining approach performs better than heuristic greedy select. Additionally, exhaustive search with window size 4 achieves a higher D#-nDCG value than window size 5.

- THUSAM-D-C-1A: Exhaustive search with window size 4. The SM result is from Subtopic N-gram Learning to rank list.
- THUSAM-D-C-1B: Exhaustive search with window size 5.The SM result is from Subtopic N-gram Learning to rank list.
- THUSAM-D-C-2A: Exhaustive search with window size 4.The SM result is from heuristic greedy select from subtopics.
- THUSAM-D-C-2B: Exhaustive search with window size 5.The SM result is from heuristic greedy select from subtopics.

5. CONCLUSIONS

In this paper, we introduce our approaches for NTCIR-11 IMine task. For subtopic mining subtask, we try to mine candidates from different data resources. The topic hierarchy is construct with different strategies. For Subtopic Mining Chinese subtask, we adopt Bottom-Up, Top-Down and Knowledge base Aided strategies. For Subtopic Mining English subtask, we adopt Bottom-Up and Up-Bottom strategies. Evaluation results have shown that candidates clustering and cluster naming are helpful for subtopic hierarchy construction. For document ranking subtask, we adopt pruning exhaustive search method to generate diversified list.

6. REFERENCES

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.

[2] F. Chen, Y. Liu, J. Li, M. Zhang, and S. Ma. A pruning algorithm for optimal diversified search. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 237–238. International World Wide Web Conferences Steering Committee, 2014.

[3] C. Fellbaum. Wordnet and wordnets. 2005.

[4] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[5] T. Sakai and R. Song. Evaluating diversified search results using per-intent graded relevance. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1043–1052. ACM, 2011.

[6] R. L. Santos, J. Peng, C. Macdonald, and I. Ounis. Explicit search result diversification through sub-queries. In *Advances in information retrieval*, pages 87–99. Springer, 2010.

[7] Y. Xue, F. Chen, A. Damien, C. Luo, X. Li, S. Huo, M. Zhang, Y. Liu, and S. Ma. Thuir at ntcir-10 intent-2 task. In *Proceedings of NTCIR*, volume 10, 2013.

[8] Y. Xue, F. Chen, T. Zhu, C. Wang, Z. Li, Y. Liu, M. Zhang, Y. Jin, and S. Ma. Thuir at ntcir-9 intent task. Citeseer.