THUIR at NTCIR-10 INTENT-2 Task*

Yufei Xue, Fei Chen, Aymeric Damien, Cheng Luo, Xin Li, Shuai Huo, Min Zhang, Yiqun Liu, 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
z-m@tsinghua.edu.cn

ABSTRACT

This paper describes our approaches and results in NTCIR-10 INTENT-2 task. In this year, we participate in subtasks for both the Chinese and English topics. We extract subtopics from multiple resources for these topics, and several subtopic clustering and re-ranking methods are proposed in this work. In Document Ranking subtask, we redefine the novelty of a document and use the new definition to re-rank the retrieved documents. Based on the existing diversification methods, we also try to selectively diversify the search results for the given queries, according to the query types determined by our strategies.

Team Name

THUIR

Subtasks

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

Keywords

query intent, subtopic mining, document ranking

1. INTRODUCTION

In NTCIR-10, THUIR group participates in INTENT task, including the subtopic mining subtask (Chinese and English) and Document Ranking subtask(Chinese).

For English subtopic mining, we apply two different processes for the candidate subtopics coming from external resources (such as Wikipedia, Google Keywords Generator, Search-Engines Suggestion, etc.) and candidate subtopics coming from web pages (extracted through the result snippet information, anchor text and the "h" tag in the top retrieved results of commercial search-engines or our own built search engine based on ClueWeb). For the first one, we introduce a new and efficient way to cluster the subtopics, based on the top result snippet information and Jaccard Similarity Coefficient. For the other one, we use the popular BM25 and Partition Around Medoid algorithms to cluster the subtopics.

For Chinese subtopic mining, the approaches are similar to our work in NTCIR-9. We extract candidate subtopics from query suggestion, Wikipeida, user log and some other resources. A voting strategy is used to rank the subtopics from different resources. Several re-ranking algorithms are used to produce differen runs. The most important and novel re-ranking approaches are based on clicked snippets from search engine. For each topic, we count the frequency of the terms which have appeared in the clicked search result snippets, and promote the subtopics with high-frequency terms. Evaluation results show that this method can improve the D-nDCG values of topics. We also try LDA algorithm on the clicked snippet text to find important terms for subtopics.

In Document Ranking subtask, we develop new diversification methods for document re-ranking. Compared to NTCIR-9, we train new values for parameters of the improved probabilistic model to retrieve documents. We redefine the document novelty in a new way, and based on this new definition, we develop a novel method to diversify the retrieval list. Based on the HITS-based re-ranking algorithm and the D#-measure-based selection algorithm, both of which are used as a separate diversification method in NTCIR-9, we selectively diversify the retrieval results, according to the type of a query. That is when a query is determined by our strategy as the navigational type, the algorithm uses HITS to re-rank the retrieved documents without any diversification and when a query is informational, the algorithm uses the D#-measure-based algorithm to diversify the results. Finally, all the strategies are combined with each other to produce the submitted runs.

2. ENGLISH SUBTOPIC MINING

2.1 External Resource Based Subtopic Mining

We observe that over the internet, there are many interesting services that we can use to help us disambiguate a query. From these resources, we can extract subtopics for an ambiguous query as well as, for some, interesting information about the subtopic popularity. The resources we used are:

- Query Completion (Google, Bing, Yahoo)
- Query Suggestion (Google, Bing, Yahoo)
- Google Insights
- Google Keywords Generator
- Wikipedia (Disambiguation feature)

*This work was supported by Natural Science Foundation (60903107, 61073071) and National High Technology Research and Development (863) Program (2011AA01A205) of China.

2.1.1 Candidate subtopics extraction and filtering

To extract the candidate subtopics, we simply submit the query to these services and scrape the subtopics returned. Many of these subtopics are irrelevant or duplicated, hence we apply a filter to keep only the valid ones. We adopt a large keyword inclusion filter. This filter removes all subtopics that do not contain any of the query words. The original query stop words are discarded such that stop words are not considered in the candidates.

2.1.2 Snippet Based Clustering

Clustering has always been an important aspect in query diversification. The snippet information provided by the search result to summarize the web page, brings us very useful information and contains some important keywords. So we propose here a solution using this feature to cluster our subtopics. For each candidate subtopic, we first submit each one to the search engines (Google, Bing, Yahoo) and crawl the snippets of the top 50 results. Then we set a table with every word found from these snippets and count their frequencies. In order to know wether two subtopics are similar or not, we calculate the Jaccard similarity coefficient:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

Where A and B are the term frequency vectors of two different subtopics to be compared. We extend this coefficient by considering both the words and their frequencies. (Hence even if many words retrieved for the two subtopics are the same, their frequencies is different, which can reduce their similarity). We implement this feature because we think that both words retrieved and their frequencies are important to determine 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:

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

Where f_{A_i} is the frequency of the i-th term in the vector.

We then create a clustering algorithm using this extended Jaccard Similarity.

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

```
1: Select k (define experimentally)
2: Create for every candidate subtopic a cluster
3: for each cluster do
4:     for each remaining cluster do
5:         if J_ext similarity of the the two clusters > k then
6:             Combine clusters
7:         end if
8:     end for
9: end for
10: Repeat 3 while the similarity between two clusters is
    above k.
```

The Jaccard similarity for a cluster is computed as the average similarity between all its candidate subtopics and all the other cluster candidate subtopics.

2.1.3 Resources features based cluster ranking

To rank our clusters, we base our approach on a multi criteria ranking. We use different scores provided by the external resources; for example, in Google Insights or Google Keywords Generator, a score is associated with each term: the popularity for Google Insights and the amount of searches for Google Keywords Generator. So we apply a ranking strategy based on the following features with their weights:

- Jaccard Similarity between the subtopic and the original query: 5%
- Google Insights score: 15%
- Google Keywords Generator score: 75%
- Belongs to the query suggestion/completion: 5%

We also consider that, if a subtopic belongs to the Wikipedia disambiguation feature, then the subtopic is important, and should be granted a better score. In order to compare the subtopic scores with each other, we normalize the scores with respect to the maximum score. For example, the top search in Google Insights or Google keywords generator will have a score of 1. Thanks to this normalization, even if the data come from different resources, we are still able to use them together.

2.2 Top Results Based Subtopic Mining

In the second approach, we propose to find the subtopics directly from the web pages. To get web pages related to the query to disambiguate, we use different search-engines: the commercial ones: Google, Bing and Yahoo, and the one built by Tsinghua University (TMiner) which is based on the Clueweb data. In this way, we are sure to only extract pages that are relevant to the query. We base our approach on a method slightly similar with the one proposed by [4] in NTCIR-9.

2.2.1 Subtopics Candidates Extraction

We first submit a query to the search engines, and get the page results. For TMiner run, we extract the candidate subtopics from different fragments coming from page snippets, page "h1" tags and in-link anchor texts. For the commercial search engine runs, we only extract the candidate subtopics from page snippet (page title and description). We adopt a vector space model to represent each fragment.

$$f = (w_{1,f}, w_{2,f}, \ldots, w_{n,f}) \tag{3}$$

Where $w_{i,f}$ is the weight of a unique word i contained in f.

We remove stop words and query words from the fragments, because they do not help us to distinguish between different fragments.

We then use the BM25 [8] algorithm to evaluate the weight:

$$w_{i,f} = \frac{k_1 + 1)tf_i}{k_1((1-b) + b\frac{dl}{avdl}) + tf_i} \log \frac{N - df_i + 0.5}{df_i + 0.5} \tag{4}$$

Where tf_i is the occurrence of word i in fragment f, and df_i is the number of documents that contain i in the corpus. dl is the length of the fragment f. $avdl$ is the average fragment length for the query. N is the total number of documents in the entire corpus. We experimentally set $k_1 = 1$ and $b = 0.6$.

2.2.2 Candidate Subtopics Clustering

We apply a modified Partitioning Around Medoids (PAM) clustering algorithm to group similar fragments together. Here is the algorithm:

Algorithm 2 Modified partitioning around medoid algorithm

1: Initialize: randomly select k of the n data points as the medoids
2: Associate each data point to the closest medoid. ("closest" here is defined using cosine similarity)
3: **for** each medoid m **do**
4: **for** each non-medoid data point o **do**
5: Swap m and o and compute the total cost of the configuration
6: **end for**
7: **end for**
8: Select the configuration with the lowest cost.
9: repeat steps 2 to 8 until there is no change in the medoid.

The similarity between two fragments is determined using the cosine similarity between their corresponding weight vectors calculated as above using the BM25 algorithm. The PAM algorithm first computes k representative objects, called medoids. A medoid can be defined as the object of a cluster, whose average dissimilarity to all the other objects in the cluster is minimal. After finding the set of medoids, each object of the data set is assigned to the nearest medoid. k is the number of clusters we want to generate and traditionally it is fixed as an input of PAM. However, in our task, it is not suitable as we do not know the number of clusters (intent) a query has; since the number is not predictable. We have to modify the PAM algorithm to make it to be able to decide an appropriate k. We first randomly choose k points as initial cluster medoids. We then assign each points to the closest medoid. If the closest medoid is over a value we set experimentally, then we set this point as a new medoid, and recalculate from the beginning.

2.2.3 Cluster Ranking

We rank the clusters according to their popularity, using the fragment rank inside the commercial search engine or TMiner and the URLs diversity from the different fragments of a cluster. So we give a greater score to the clusters that contains fragments from higher ranked pages and clusters that contains fragments from many different URLs. Here is the formula used to calculate the score for each cluster:

$$Score(c) = \sum_{f \in Frag(c)} (1 - \frac{w(f)}{N}) \quad (5)$$

Where N is the total number of unique URLs in cluster c and $w(f)$ is the number of unique URLs containing fragment f. Learning to rank techniques can also be adopted with sufficient training examples and we would like to add this to our future work.

2.2.4 Cluster Name Generation

From the different fragments, we need to generate a clear name for the cluster. For each cluster, we select the most frequent word and extend it to an n-gram based on the frequency of the other words. We also set that frequency limit experimentally. We keep stop words because they can be interesting to name the intent. Then we check if we need to add the keyword (that we removed from every fragment) or not. This is done by checking whether the original fragments contain the keyword. If more than 50% contains the keyword, we add it, using its position between the most frequent words, in order to place it correctly.

2.3 Resources Fusion

To improve diversity, we combine the subtopics we extracted from both external resources and top result pages. Both datas are coming from two different aspects of the internet: the one, which is from the external resources represents the queries that people are looking for. And the other one, which is mined from the top results pages, shows the information provided by the website owners or participants.

2.3.1 Semantic Similarity Re-clustering

To combine our data, we use a linear combination of the subtopics. After that combination, many subtopics are actually duplicated, so we have to choose a way to re-cluster and re-rank the subtopics. We cluster them according to their semantic similarity, using WordNet and the technique proposed by [6]. Lin describes a method to compute the semantic relatedness of word senses using the information content of the concept in WordNet. We experimentally define a similarity value to decide wether or not two candidates should be clustered together. We choose this system for its speed, but we could also have used both clustering method presented in this paper (Top results Snippet based or Partition around medoid with cosine similarity) to improve the precision. However, as all subtopics are already clustered before the fusion, it is much easier to re-cluster them, and we do not need a accurate clustering method to get good result precision. So we try to use a simple semantic similarity based on WordNet to obtain a tradeoff between speed and performance.

2.3.2 Re-ranking

To re-rank the subtopics, we normalize them by assigning them a percentage of the maximum subtopic score of each query. As a result, every subtopic score is a percentage of its run best subtopic score. After this normalisation we can compares the score of all subtopics and re-rank them.

2.4 Submitted Runs

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

- THUIR-S-E-1A: THUIR-S-E-2A + THUIR-S-E-3A + THUIR-S-E-4A, Linear combination, Semantic similarity based re-clustering
- THUIR-S-E-2A: Extraction from multiple resources (Google Insights, Google Keywords Generator, Query Suggestion/Completion, Wikipedia) + Snippet based clustering.
- THUIR-S-E-3A: Extraction From TMiner top results Snippet, Anchors and H1, BM25, Partition around medoid
- THUIR-S-E-4A: Extraction From Search Engines top results Snippet + Query Suggestion/Completion, BM25, Partition around medoid

Table 1: Experimental result of English subtopic mining runs

Runtag	I-rec@10	D-nDCG@10	D#-nDCG@10
THUIR-S-E-1A	0.4512	0.4775	0.4644
THUIR-S-E-2A	0.4333	0.4795	0.4564
THUIR-S-E-3A	0.4346	0.4726	0.4536
THUIR-S-E-4A	0.4364	0.5062	0.4713
THUIR-S-E-5A	0.4253	0.4893	0.4573

- THUIR-S-E-5A: Extraction From TMiner top results Snippet - BM25 - Partition around medoid + Wikipedia + Official Query Suggestion/Completion; Linear combination; Semantic similarity based re-clustering

2.5 Evaluation Results

To evaluate our techniques, we make different combinations, and submit 5 runs. The D#-nDCG, D-nDCG and I-rec values of the results are shown in Table 3. We can see that THUIR-S-E-4A is the best in terms of D#-nDCG. We do not expect this run to perform best. Instead, we have expected THUIR-S-E-1A, which is the fusion of all our other runs, e.g. THUIR-S-E-2A, THUIR-S-E-3A and THUIR-S-E-5A, to be the best. Even if the fusion run gets a better diversity (I-rec), its relevancy (D-nDCG) is reduced. It is considered that we should choose a more effective re-clustering algorithm than a simple semantic similarity based re-clustering. On the other hand, THUIR-S-E-4A only relies on the top results of commercial search engines, which obtains the best D-nDCG score (relevancy) and a good diversity score (I-rec), implying the best D#-nDCG value. THUIR-S-E-5A is our baseline, and is comparable with THUIR-S-E-1A, THUIR-S-E-3A and THUIR-S-E-4A. We can see that all methods bring a better diversity but not always a better relevancy. The official overview [10] shows that the differences among these submitted runs are not statistical significantly.

3. CHINESE SUBTOPIC MINING

3.1 Candidate Subtopics From External Resources

Similar to English subtopic mining, the Chinese candidate subtopics also come from different external resources.

For each query, we use query suggestions from 4 different search engines. Every search engine returns 10 suggestions, hence there are at most 40 related queries. Obviously, lots of them are reduplicated because different search engines may recommend the same queries. It is reasonable to assume that a query which is recommended by some different search engines is more likely to be an important user intent than a query which is recommended by only one search engine. Based on this assumption, we use the search engines to vote for all related queries. In detail, we rank the recommended queries based on the frequency of each query's appearance in different search engines.

In this step, we ignore the auxiliary words and white spaces in Chinese. We also filter out the queries which are substrings of the given query, since such queries could not be a subtopic of the given query.

We use Chinese Wikipeida to obtain more subtopics in three different ways. In Wikipedia, a title can be associated with more than one Wikipedia topic. There are a lot of disambiguation pages to resolve this kind of conflicts. Different meanings of an ambiguous topic are listed on the disambiguation page. We look up each query in Wikipedia. If it has a disambiguation page, the topics on the page would be regarded as the candidate subtopics.

For each given query, we also compare it with all the topics in Wikipedia. If any topic contains the given query as a substring, this topic is also adopted as a candidate.

On the page of topic in Wikipedia, there is usually an index which summaries different facets of the topic. If we find the given query in Wikipedia as a topic, all the index items on the page will be added into the candidate subtopic collection. Besides Wikipedia, we introduce the index of another Chinese online encyclopedia (hudong.com) as our corpus.

We combine the candidates from online encyclopedias with the ones from query suggestion using the same voting strategy. Different from query suggestion, the candidates from different way of encyclopedia are given different weights in voting.

In Wikipedia, a topic may belong to several categories. If different topics belong to the same category, they should be the same kind of concepts in some way. In our subtopic list, if two subtopics belong to a same category in Wikipedia, we give a penalty on the weight of the reduplicate subtopic.

3.2 Re-ranking Based on Coverage Rate

From the methods described above, we have obtained several candidate subtopics for each query, and a weight assigned for each subtopic. However, for some subtopics extracted for the same query, they share the same weight. Obviously, it is inappropriate to rank these subtopics randomly, so we re-rank them based on the coverage rate between the subtopics and the query.

In most cases, if a subtopic covers more terms of a query, it will more likely be relevant to the query and becomes a subtopic. And when some subtopics cover the same number of terms, a shorter one indicates that it is more concise and describes the subtopic of the query better. So we segment the query into different words and define the *coverage rate* between a subtopic and a query as the ratio between the number of words in the query that appear in the subtopic and the number of all the words in the query. To take the factors of coverage rate and length of the subtopic into consideration, we adjust the weight for each subtopic as follows: $weight_{new} = weight_{old} + 0.05 \times (coverage\ rate) + 0.005/(intent\ length)$. The minimum difference between the original weights is 0.1, so only when the weights of the subtopics are the same do the coverage rate and the length of the subtopic make a difference. The larger the coverage rate and the shorter the subtopic are, the higher weight the subtopic gets.

3.3 Optimizing Intent List by Snippet Click

3.3.1 Snippet Click Model

Looking into users' interaction process with the search engine, it is reasonable to think that users' click through behavior contains clues of their information needs. When user clicks a certain search result, it does not necessarily mean that he is interested in the document due to he has not viewed yet. His first impression mainly comes from the

title and snippet of the search result. We can assume that he/she is interested in the snippets of the corresponding results because these snippets might describe their intents. Dou et al.[3] study the problem of using aggregate clickthrough log, and find that although some individual user clicks are unreliable, the aggregation of a large number of user clicks provides a valuable indicator of relevance preference. Furthermore, we can assume that: 1)on search result level, the snippet of the search result which is clicked more frequently by users reflects more important user intent. 2) If we split the sentences in the snippet into a list of terms, the more often the term appears in snippet, the more probably it is useful to describe the user intent.

Based on these observations, we propose a method utilizing clicked snippet to recognize user's intent and re-rank the list. For each query q, we can crawl its top ten search results to build a snippet document and title document with snippet content and user behavior data. More specifically, in each document, we will make sure that for term t in document

$$freq_{snippet}(t) = \sum_{i=0}^{10} (freq_{snippet_i}(t) * CT_i)$$

$$freq_{title}(t) = \sum_{i=0}^{10} (freq_{title_i}(t) * CT_i)$$

where $freq_{snippet_i}(t)$ and $freq_{title_i}(t)$ and represents the term t's frequency in the i^{th} search result's snippet and title. CT_i means the times that the i^{th} search result is clicked by users.

For each term t in snippet and title, we can calculate its score as follows:

$$TermScore(t) = \sum_{i=0}^{10} (freq_{snippet}(t) + \lambda * freq_{title}(t))$$

λ is the weight of the title. In our experiment, the algorithm gets best performance when $\lambda = 1.2$. In this way, we build a term list L ranked by the score in descending order. We delete the terms which appear in the original query because it brings no additional information to help us understand the user intent.

It should be noted that the term list might be very long and the score might range in a very large scale. The statistics on 99 queries from NTCIR-10 INTENT-2 dataset show that the longest list contains 1079 terms. In some way, the descending order of score means the decreasing order of reliability. Actually we only choose the top k terms. To solve the score range problem, we normalize the top k scores according to their order. Suppose there are k terms in total and the i^{th} score should be:

$$NormScore(t_i) = 1.0 - (\alpha - \beta) * \frac{i}{k}$$

We take α and β as two independent parameters to make sure that the score ranges between α and β. In the experiments, α is set to 1.0 and β is set to 0.5. In the other way the sum of two terms' scores is always larger than one's.

With this list we can optimize the intent list by re-ranking it. For a certain candidate c, its snippet score can be calculated as:

$$SnippetScore(c, k) = \sum_{i=0}^{i<k} NormScore(t_i) * I_{t_i \ in \ c}$$

Suppose $OrigScore(c)$ is the original score in the intent list. It is necessary to combine the original score with the snippet score.

$$Score(t, \lambda, k) = \lambda * OrigScore(t) + (1 - \lambda) NormScore(t, k)$$

We can use this score to re-rank intent candidate list. λ and k are two parameters in this method. Experiments on NTCIR-10 INTENT-2 labelled dataset shows when $\lambda = 0.53$ and $k = 12$, the experiments get the best performance.

3.3.2 LDA on Snippet Click Document

Latent Dirichlet allocation (LDA) is a topic model that is presented by D. Blei, etc.[1] In LDA, a document is viewed as a mixture of different topics. With the LDA model, we can estimate the latent topics and their probability distribution.

In last section, we have introduced the snippet document. It is reasonable to assume that the snippet document contains the most popular subtopics of the query. So we use LDA to estimate the topics of the snippet click document. These topics may correspond to the various intents of the given query.

In LDA, a topic is represented by the probabilities of the words under it. There is no explicit phrase representation of topics. Therefore the topics from LDA can not be added into the intent list directly. We have to bridge the gap between the explicit search intents and the implicit topics of the snippet click document. Algorithm 3 shows the process of transferring the implicit topic to the explicit representation.

Algorithm 3

1: Construct the snippet document d of given query q.
2: Remove all the appearances of given query from d, and get a new document d'.
3: Estimate the latent topics $t_1, t_2, ..., t_n$ of d'.
4: **for** each topic t_k **do**
5: Get two words with the largest probabilities to be generated, denoted by w_{k1} and w_{k2}.
6: **end for**
7: Connect up q to w_{k1} and w_{k2}, and get 4 different phrases.
8: If any of the phrases has appeared in the snippet click document d, add the phrase into the intent candidate list with weight 0.4.

In the algorithm, we use the high-probability words in topic to represent the topic. We remove the given query from the d so that the query itself will not appear as high-probability words. After we get the topics, we connect up q to the words from different topics to make up some phrases. We add a phrase into the candidate intent when it appears in d, so that the selected phrases may be human-readable.

3.4 Evaluation Results

According to the previous described methods, we make different combinations of the methods and submit 5 runs. The runs are introduced in Table 2.

The mean values of D#-nDCG, D-nDCG and I-rec are shown in Table 3. It shows that THUIR-S-C-3A gets the highest D-nDCG value. This run combines basic voting with fine adjustment and random walk query list. These strategies successfully make the important intents to reach high ranks. The I-rec of this run is the lowest. Of all the runs, THUIR-S-C-5A is our baseline. Compared it to THUIR-S-C-1A, THUIR-S-C-2A and THUIR-S-C-3A, we can see that

Table 2: Methods of Runs

Runtag	QS	CR	RW	Snippet	LDA
THUIR-S-C-1A	✓	✓	✓	✓	✓
THUIR-S-C-2A	✓	✓	✓		
THUIR-S-C-3A	✓	✓	✓	✓	
THUIR-S-C-4A	✓	✓	✓*		
THUIR-S-C-5A	✓	✓			

QR: Intent candidates from query recommendations, Wikipedia and Hudong.
CR: Coverage Rate.
RW: Random walk over user click graph.
Snippet: Optimizing with snippet document.
LDA: Optimizing with the LDA result of snippet document.
* Use SogouQ corpus for constructing click graph in stead of the larger corpus.

Table 3: Experimental results of Chinese subtopic mining runs

Runtag	I-rec@10	D-nDCG@10	D#-nDCG@10
THUIR-S-C-1A	0.3839	0.4843	0.4341
THUIR-S-C-2A	0.3839	0.4816	0.4327
THUIR-S-C-3A	0.3786	0.5028	0.4407
THUIR-S-C-4A	0.3792	0.4739	0.4266
THUIR-S-C-5A	0.3892	0.4798	0.4345

all the extra efforts can improve the D-nDCG value, but decrease the I-rec value. In D#-nDCG measure, THUIR-S-C-3A also gets the highest value. THUIR-S-C-4A is a special run which is based on public dataset. It does not perform as well as THUIR-S-C-2A in terms of any measure since the data size in random walk is much smaller. From the Overview paper [10] we can find all these submitted runs are not significantly different in terms of D-nDCG, I-rec and D-#nDCG.

4. CHINESE DOCUMENT RANKING

4.1 Retrieval Models and Dataset

In the retrieval step, we leverage the same improved probabilistic model and the same retrieval strategies as the ones we used in NTCIR-9[12] for document ranking. We also use SogouT dataset to train different parameters in these models or retrieval strategies. These parameters are determined and shown in Table 4.

4.2 Result Re-ranking with HITS

In NTCIR-9, we adopt HITS to re-rank the baseline search results in the Document ranking subtask. It re-ranks the documents that are the m^{th} biggest in terms of either Authority or Hub values up to the front. The new rank is determined based on the original rank, the Authority and the Hub values of the document[5]. It is proved that HITS can stably improve the diversity of the search result on both the TREC-based dataset and the SogouT dataset[12].

Table 4: Parameter values of the improved probabilistic model

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

4.3 D#-measure-based Selection Algorithm

To evaluate the diversified result list, Sakai et al. propose the D#-measure [11]. It computes a global gain for every document by linearly combining the gains with respect to subtopics. It also takes the subtopic recall of the result list into account. Based on this, We provide a D#-measure-based selection algorithm[12]. This algorithm iteratively select a document, which has the currently biggest global gain and subtopic recall, to produce a diversified result list.

4.4 Query Type Identification

Previous work on understanding user search goals has shown that user queries can be classified as informational or navigational[2][9]. If the type of a user query can be identified, appropriate document ranking strategy can be applied to the query type to improve the performance of the search engine. For example, if a user types in "IJCAI13", which is a navigational query, the homepage of the conference should be placed on the top. While an informational query requires different documents on all aspects. In this paper, we use three features mentioned in [7]: n Clicks Satisfied (nCS), top n Results Satisfied (nRS), Click Distribution (CD) to train a decision tree model to identify the query type. The features are described as below:

- nCS: When submitting a navigational type query, the user tends to click a small number of URLs in the results list. The nCS feature is defined as:
  nCS(Query q) = #(Session of q that involves less than n clicks) /#(Session of q)

- nRS: When submitting a navigational type query, the user tends to click the first few URLs in the results list. The nRS feature is defined as:
  nRS(Query q) = #(Session of q that involves clicks only on top n results)/#(Session of q)

- CD: If the goal of a query is navigational, most clicks should be concentrated on a single result. The CD feature is defined as:
  CD(Query q) = #(Click on the most popular result of q)/ #(Click on all results of q)

Based on the three features, we use standard C4.5 algorithm to train a decision tree model to identify the query type. The algorithm is described as below:

4.5 Selective Diversification

With the query type known in advance, we can selectively diversify its search result. We use the way described in Section 4.4 to determine the query type. If a query belongs to the navigational type, then the result list retrieved for the query needs no diversification. In this situation, we use the HITS to re-rank the results retrieved using the strategies described in Section 4.2 to produce the final result list. On the other hand, if the query belongs to the informational type, we use the D#-measure-based selection algorithm to diversify the retrieval result.

Algorithm 4 A decision tree based query type identification

Require: Feature nCS, nRS, CD; Threshold p1, p2, p3, p4;
if nRS $\geq$ p1 **then**
  predict = 1
else
  if nCS $\geq$ p2 **then**
    if CD $\geq$ p3 **then** predict = 1
    else
      predict = 0
    end if
  else
    if CD $\geq$ p4 **then**
      predict = 1
    else
      predict = 0
    end if
  end if
end if

4.6 Diversify Results Based on Novelty

As we know, most queries have not only one subtopic. In fact, how to find subtopics of a query and measure the probability that which subtopic it belongs to as well as the probability that which document can satisfy the subtopics remains a problem. Therefore, we are focused on diversifying the adhoc results to achieve diversity directly. Our main concept of the method is that when deciding the candidate document placed in position k, we select a document that can recommend the most novel information despite all the results before position k. There are two assumptions for this method. One is that in a given adhoc result, all the documents are of high relevance to the query and ranked by the probability that they can meet the user's information needs in the query. The other is that the adhoc result covers a variety of information needs users may have, regardless of the position of each document. So we do not need to search for documents which meet the needs of all kinds of information, but better ranking the documents to cover different information needs in the top results. If the original top n documents of adhoc results are $d_0, d_1, d_2, d_3, \ldots d_{n-1}$, the goal is to re-rank the documents for diversity of top k results. Let S be the current re-ranked list and ω_i be a corpus vector that represents the content of the document d_i. Function $f(d_i, S)$ is used to measure the novel information that document d_i can introduce into the selected documents S. Then our strategy can be described as Algorithm 5.

Algorithm 5

Set $S = d_0$
while $||\omega_i|| > 0$ and $|S| < k$ **do**
  $\omega_i = argmax f(d_i, S)$
  Add d_i to the end of S
end while
for i from 1 to n **do**
  if d_i are not in S **then**
    Add d_i to the end of S
  end if
end for

This algorithm is to select top k results and maintain the relative order of the other results to form a new result list. Following we introduce the two important factors in the algorithm in detail, the corpus vector ω_i and the novelty function f.

4.6.1 Document representation

The corpus vector should describe the content of the document and can be a representative of the document comparing with the other documents in the adhoc list. We try to use anchor, title and text information of all the documents in the adhoc list to form a lexicon separately. Suppose the lexicon contains N terms, then a document can be described as an N-dimensional vector. The dimension N is based on the size of adhoc result list, to be exact, the top n documents of adhoc results. The ith component value of the vector is computed using $tf \cdot idf$ value of the ith term according to the document. See the following,

$$\omega =< w_1, w_2, \ldots w_N >=< tf_1 \cdot idf_1, tf_2 \cdot idf_2, \ldots tf_N \cdot idf_N > \quad (6)$$

We try different n from 500 to 3000 on the queries, finding that $n = 1000$ is a relatively stable range on performance. At the same time, we find that using page-level anchor to build the lexicon is the best in ERR-IA evaluation. Finally we choose the page-level anchor of top 1000 documents in adhoc result list to form the vector and re-rank adhoc results by selecting top $k = 100$ documents according to novelty.

4.6.2 Measure the novelty

The novelty function $f(d_i, S)$ should integrate the differences between d_i and each document in S in terms of the relevance of d_i with respect to the query. We use the cosine value of two vectors to measure the difference between two documents. The novelty of the candidate d_i is computed as follows,

$$f(d_i, S) = \sum_{d_j \in S} \alpha_j + \lambda \times \cos \omega_i, \omega_j > \quad (7)$$

The weight parameter α_j = N- original rank of d_j. We try different λ from 0.01 to 0.3 and find $\lambda = 0.1$ is the best in the result evaluation. Original rank in adhoc list can reflect the quality of the document itself for the query. Through linear combination of the weight parameter and the novelty, the algorithm tends to select the document which can not only be a good answer for the query but also introduce more information than documents before. As a result, it may better satisfy the diverse user's information needs.

4.7 Submitted Results

In Document ranking subtask, we submit 6 runs. One of them is produced using the NTCIR-9 system and its name ends with "R1" as the official required. Other 5 runs are automatically created by our new systems. Table 5 shows the descriptions of each submitted runs and Table 6 shows their evaluation results [10]. The click-based re-rank strategy in Table 5 is the same as the one described in [12].

As Table 6 shows, THUIR-D-C-3A, which is produced by the improved probabilistic model, outperforms the THUIR-D-C-R1, which is produced by the same probabilistic model but with different parameter values, in terms of all the listed measures. All the results that are based on THUIR-D-C-3A are very similar, even though they are produced by different strategies. In fact, these runs are not significantly different [10]. This implies these strategies take limit influences to the

Table 5: Methods of runs

THUIR-D-C-1A	THUIR-D-C-2A + click-based re-rank (large click logs).
THUIR-D-C-2A	THUIR-D-C-3A + novelty-based re-rank.
THUIR-D-C-3A	Retrieve on full text, anchor and click text documents (baseline of 1A and 2A).
THUIR-D-C-4A	Subtopic mining + retrieve on multiple subtopics + decay global gain based diverse results selection.
THUIR-D-C-5B	Official baseline + HITS-based re-rank + novelty-based re-rank + click-based re-rank (SogouQ).
THUIR-D-C-R1	THUIR-D-C-5 retrieval on full text, anchor text and click text, + HITS-based re-rank.

Table 6: Experimental results of Chinese document ranking runs

	I-rec@10	D-nDCG@10	D#-nDCG@10	DIN-nDCG@10	P+Q
THUIR-D-C-1A	0.7288	0.4218	0.5753	0.2868	0.2667
THUIR-D-C-2A	0.7258	0.4201	0.5729	0.2865	0.2663
THUIR-D-C-3A	0.7247	0.4207	0.5727	0.2858	0.2653
THUIR-D-C-4A	0.6731	0.3587	0.5159	0.2611	0.2203
THUIR-D-C-5B	0.6313	0.3571	0.4942	0.2406	0.2298
THUIR-D-C-R1	0.7085	0.4096	0.5590	0.2806	0.2569

retrieval result diversification. Comparing THUIR-D-C-5B, which takes the retrieval result provided by the NTCIR official as its baseline, with the THUIR-D-C-3A-based results, we may conclude that the improvements of the latter results are caused by using different parameter values. [10] also show that these improvements are significant. THUIR-D-C-4A is the only result that is based on the subtopics mined in the Subtopic mining subtask. We leverage the improved probabilistic model to retrieve documents for the subtopics mined in THUIR-S-C-1A, and leverage the strategies described in Section 4.4 to determine the type of each query. THUIR-D-C-4A is then produced using the selective diversification strategy. It is worse than the THUIR-D-C-3A based results.

5. CONCLUSIONS

In this paper, we introduce our approaches for NTCIR-10 INTENT-2 task. For subtopic mining subtask, we try to mine candidates and optimize the ranking of subtopics by using different data, including query suggestion, Wikipedia, user log, search result snippet, etc. Our results have shown that reliable external resources and user behavior data are helpful for mining subtopics. For document ranking subtask, we develop new novelty-based re-ranking method and the selectively diversification method. The experiment results show that these approaches can improve the evaluation values in terms of different kinds of measures.

6. REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[2] A. Broder. A taxonomy of web search. In *ACM Sigir forum*, volume 36, pages 3–10. ACM, 2002.

[3] Z. Dou, R. Song, X. Yuan, and J.-R. Wen. Are click-through data adequate for learning web search rankings? In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 73–82. ACM, 2008.

[4] J. Han, Q. Wang, N. Orii, Z. Dou, T. Sakai, and R. Song. Microsoft research asia at the ntcir-9 intent task. In *Proceedings of NTCIR-9 Workshop Meeting*, pages 116–122, 2011.

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

[6] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th international conference on Machine Learning*, volume 1, pages 296–304. San Francisco, 1998.

[7] Y. Liu, M. Zhang, L. Ru, and S. Ma. Automatic query type identification based on click through information. *Information Retrieval Technology*, pages 593–600, 2006.

[8] S. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, and A. Payne. Okapi at trec-4. In *Proceedings of TREC*, volume 4, pages 73–96, 1995.

[9] D. Rose and D. Levinson. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web*, pages 13–19. ACM, 2004.

[10] T. Sakai, Z. Dou, T. Yamamoto, Y. Liu, M. Zhang, and R. Song. Overview of the ntcir-10 intent-2 task. In *Proceedings of NTCIR-10 Wordshop Meeting*, 2013.

[11] 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.

[12] 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. In *Proceedings of NTCIR-9 Workshop Meeting*, pages 123–128, 2011.