Nanjing University at TREC Dynamic Domain 2016: Improvement of MDP Model

Department of Software Engineering, Nanjing University

fengliu@nju.edu.cn

Abstract

This paper describes the joint submission by Nanjing University of TREC 2016 Dynamic Domain track. We submitted one run for the main track. In our view, there are two principal components involved in the search process, the user and the search engine. And the Dynamic Domain process can be separated into two phases: a traditional static information retrieval phase to generate an initial set of documents, followed by a dynamic information retrieval phase which takes user feedback into account to improve the results. We developed an algorithm that combines a dynamic search algorithm and a re-ranking baseline in the whole phase. During the search, we collect the relevance feedback from the simulator and use them to train our model and update its attribute. Our algorithm terminates itself automatically when a certain number of iteration has been finished or no other relevant documents can be retrieved.

1. Introduction

The Dynamic Domain challenge is a dynamic searching project that iteratively enhances the retrieval results after the first feedback set is generated. Our goal is to get better feedback from a simulating user called "jig" through as few iterations as possible. In this project, we have two search areas, namely EBOLA and POLAR. The data set we get is the CBOR object, which contains a number of topics, each of that has several unequal subtopics. Data structure for CBOR is similar to JSON and it consist of key pairs, which is shown below. We need to interact with the user to get feedback, and give articles which meet the user's interest. All the articles we get are crawled from the web page in the data set given.

```
{   'key': 'ebola-03cad6ee34e9dc0aeb77e4c5d31aad2aa41f6ad819f23b8504612d6e6de8a18c',
    'request': {   'body':    None,
                   'client':   { '...': '...' },
                   'headers': [ [ 'Accept-Language': 'en-US,en' ], [ '...', '...' ] ],
                   'method':  'GET'
    },
    'response': {  'body':    '<!DOCTYPE html> <html lang=\'en\' class=\'js-disabled\'> <head> ... </html>',
                   'status':  '200',
                   'headers': [ ['Content-Type', 'text\/html' ], [ '...', '...'] ],
    },
    'timestamp': 1421064000L,
    'url': 'http://www.nature.com/news/ebola-1.15750',
    'indices': [
        { 'key': 'crawl', 'value': 'ebola' },
        { 'key': '...', 'value': '...'},
    ],
    'features': [
        { '...': '...' }, { '...': '...' }
    ],
}
```

Figure 1. An Example of the CBOR Data Format

In each interaction, we give the user five articles, then "jig" will give feedback for each article. In the feedback, we can know whether "jig" is interested in these articles. If so, "jig" will give the relevance between the article and subtopics related to the topic. It helps us to determine the articles returned to user for the next iteration.

We use Indri and LDA initialize our iteration. At start, user gives a topic and we give the first feedback by Indri and LDA, which contains five articles. We adjusted the percentage of influence to consequence

of Indri and to get a best result in first interaction. After receiving the feedback from user, we take advantage of MDP model and NRE algorithm to return the next five articles until the termination condition is met and the iterative process ends. The termination condition we set is the number of iterations reaches five or there are three consecutive iterations which results are not positive.

2. Related Works

2.1 Indri

Indri is a new search engine from the Lemur project. From the Initial Search Phase to the Dynamic Refinement Phase, Indri is used as our information retrieval process base line. Especially, the Initial Phase which is very essential during the whole process is almost fully dependent on Indri.

2.2 Latent Dirichlet Allocation(LDA)

LDA is an algorithm that takes in a distribution of words and discovers the aggregations of topics it implies. In our system, we use LDA to the task of discovering five different topic aggregations from 30 documents returned by Indri from the current query. Next, the system builds five expanded queries by adding the 5 most principal words in the distribution of each topic to the initial query, one for each of the five topics aggregations. It then does a new Indri search with each of the five new queries, and keeps the top document of each one as the set of five LDA recommendations.

2.3 Markov Decision Process (MDP)

MDP is a widely used decision model which constructs the states set S and action set A for all related agents in the decision process. In our system, we use MDP by two steps. One is training process. The other is used in searching. The action of agents will effect the states which making the states transform uncertainly. The feedback of the actions will impact the agent's choice on the action. In our system the query Q is modeled as the set of states S. The updating of the weight of the relevancy is considered the set of actions A. The result of one query will effect the result of the next query.

3. Initial Search Phase

The first phase consists in an information retrieval process to obtain an initial set of five documents based on the user's original query, which is a stage we regard as the most important part because the dynamic refinement phase is fully dependent on the feedback the jig system returns. If the first feedback does not get what the user what, it will lead to an "false start".

In order to get the best feedback from the jig, we decided to use Indri as our base. Firstly, we transferred the data TREC provided into TRECTEXT/TRECWEB format which can help Indri build a retrieval model. Meanwhile, we pre-processed the corpus by using the porter stemmer algorithm over the whole corpus with the library NLTK and then we also cleaned the HTML tag and stop words with NLTK. After the model was built up. It is not difficult to get 5 different documents through Indri.

At the very beginning, Indri was used to fetch the first five docs. The result turned out to be good. However, a classic IR system can only give us the top 5 words by keywords, which contrast to our initial purpose, trying to find more related and user-interested subtopics so that the diversification of the results need to be the main point. It is necessary for us to discover topics present within the result set and return one representative document per topic. In order to do this, we found that a Latent Dirichlet Allocation (LDA) algorithm for topic modelling. This LDA algorithm can help us re-rank the documents and give us the final result of the initial phase.

Many strategies have been tried to get the best set of five documents. Firstly, Indri was used as the base and to get 30 docs returned by Indri to build a list. Then we use LDA algorithm to divide these 30 docs into 5 groups, and in each group we rank the 6 passages so that we get 5 different subtopics created. So each top doc from the five groups can get into a group and become the set of the first result. However, it turns out to be not good because of the bad practice of clustering. And then a new way was adopted. We extracted the key words of these 30 docs through LDA and generate a key word list. We divided the key words into 5 groups. Nest, the system builds five expanded queries, one of each of the five topics groups, by adding the five most probable words in the distribution of each topic group to the current query. It then runs a new Indri search with each of the five new queries, and keeps the top document of each one as the set of five docs to return.

However, it worked not as good as we expected. The Indri base that counts most in the process. We found

that the Indri direct result docs have some docs in common with the method using LDA. We thus decided to create a weighted combination of these two sets. The re-partition of t weights is 50% and 50% at first. After several test, we found that Indri produced better results than LDA. Consequently, we changed the weights to 70%,30%. After the re-rank, the top 5 docs were submitted.

4. Dynamic Refinement Phase

The dynamic refinement phase starts with a set of five documents, provided to the users, derived from the initial phase or from a previous iteration of this refinement phase. The user will interact with the system by providing feedback indicating which documents and which passages they find.) We use Named Entity Recognition (NER) algorithm to recognize the most potentially informative keywords of the feedback, contributing to reformulate the query to find additional relevant documents.

The dynamic refinement process of our system takes the important information from the set of top five documents of 30 documents retrieved. to improve the result There are two possible situations the system could be in. The first is if the user marked at least one document as relevant which means that we can consider that we are in the good area of the domain to search. The second situation is the opposite situation, when there are zero relevant documents in the set of five results.

4.1 Positive Feedback Situation

In the first situation, our dynamic refinement algorithm will take advantage of the positive feedback information to generate a new query. We used the extraction of the most informative words as we considered as a high informative part of the feedback. Because of the significant difference between passages length, taking the whole passage can't be an option in most situations. In order to focus on the most informative words, we use Named Entity Recognition (NER) to capture named entities on the feedback. We considered that something named in the feedback indicates that this entity can have compatible significance or meaning for the author for the related subtopic for the potential new topic which are helpful for the diversification and specialization. We use the words obtained from the above process to expand the initial query. We thought that the positive feedback will help us refine the query.

4.2 "False start" Situation

In the second or "false start" situation, the user has marked none of the top five documents of the 30 previously returned by the initial phase as relevant. Fortunately, because of the good performance of the initial phase, the "false start" situation is rare. However, We considered that it is important for our system to be able to deal with the rare but adverse situation and because of the considering of time whether the system can find the new five document is influential. In order to overcome this adversity quickly, we abandon the top five documents and select the second top five document. If the set of the second top five document is also disappointing, we continue to abandon it and choose the new set by the ranking given by the initial phase until we find the marked documents from the 30 documents. Obviously, after 5 abandoned, the set will be empty which means no one document is be marked by the user. If that happened, it would be very difficult. This situation is difficult for two reasons. The first one is that we are in a bad area totally, and it might be difficult to find a relevant document. The second one is that we do not find documents and we lose time in the process of retrieving documents. The CubeTest metric take time into account and, as a consequence, ending at this moment should be the optimal.

5. Information Retrival Process

After reformulating the query, we continued into the search part based on the Markov Decision Process (MDP) to find 5 new documents related with the query return to the user. A DR process and an IP process combine the one iteration process. The result of the IP processor is sensitive to the result of the previous query and is influential to the next DR and IP process.

The information retrieval process occurs after each dynamic refinement process in our system. This process will return 5 different most related documents to the user and then may start a new DR process. With the new query, we continue into the search part which is based on the Markov Decision Process (MDP). MDP is widely used decision model which construct the states set S and action set A for all related agents in the decision process. The action of agents will effect the states which making the states transform uncertainly. The feedback of the actions will impact the agent's choice on the action. In our system the query Q is modeled as the set of states S. The updating of the weight of the relevancy is considered the set of actions A. We return the Top 5 documents ranked based on the value of

accumulating benefits which are calculated by the weight of the relevancy to the user. Then The feedback given by the user will influence the reformulation of query in DR process effecting on the weight of the relevancy, aka, the next action which means these are the feedback of the previous action and that will influence the next action, the updating process that directly lead to the result of the next searching result. In this method, The previous result of query will influence the result of a new IP process and also the new result will effect the next query.

The IP processing is divided into four parts as following.(all the symbol in the below share the same meaning with the training process):

a). Updating the $p_s(t \mid d)$ according to the query.

the processing of updating is the same with the updating of $p_s(t \mid d)$ according to the query.

b) calculate the relevance between one document and the current query. The details are as follow

1. if i =1, the first search in this session then the relevance:

$$Score(q_1, d) = \log p(q_1 \mid d)$$

2. else i > 1, then the relevance:

$$Score(q_i, d) = \log p(q_i \mid d) + \alpha \sum_{t \in q_{theme}} [1 \quad p(t \mid d_{i-1}^*)] \log p_{us}(t \mid d)$$
$$\beta \sum_{\substack{t \in +\Delta q \\ t \in d_{i-1}^*}} p_{us}(t \mid d_{i-1}^*) \log p_{us}(t \mid d) + \varepsilon \sum_{\substack{t \in \Delta q \\ t \quad d_{i-1}^*}} idf(t) \log p_{us}(t \mid d)$$
$$\delta \sum_{t \in \ \Delta q} p_{us}(t \mid d_{i-1}^*) \log p_{us}(t \mid d)$$

$$P(q_i \mid d) = 1 - \prod_{t \in q_i} (1 - P_s(t \mid d))$$

In this formula, $p(q_i \mid d)$ is represent the Instant Benefit.. And the $P_s(t \mid d)$ is the obtained in the training process and has been updated by the feedback. α, β, ϵ, δ are the discount factors of each action. According to the empirical data, we set the α to 2.15, β to 1.75, ϵ to 0.07, δ to 0.42.

c) calculate the relevance between the document and the entire session.We use the formula as follow.

$$Score_{session}(q_n, d) = \sum_{i=1}^{n} r^{n-1} Score(q_i, d),$$
$$Score(q_1, d) = \log p(q_1 \mid d)$$

We set the discount factor of MDP to 0.8 and taking into account that the user will not be satisfied with the repeating queries and the result of those queries, we set the repeating queries' discount factor of MDP to 0. The formula is as follow.

$$\gamma_i' = \begin{cases} 0, \{i \mid i \in [j,k), \exists\, q_j = q_k, j < k\} \\ \gamma_{i,} \quad otherwise \end{cases}$$

d) return the top5 documents to the user and end the IR process.

6. Statistics Calculating and Updating Phase

There are two parts in this phase: statistics calculating and $P_{us}(t|d)$ updating. The statistics calculating process is the first step after data preprocessing. It will calculate the values needed in the following refinement and score calculating steps. The $P_{us}(t|d)$ represents the relevance between word term t and doc d. It updates every time query changes in the DR process, which will adjust the score of different docs.

6.1 Calculating Details

The statistics calculating process mainly calculate the following values:

1. $c(t, d)$. Count of term t in doc d.
2. $c(t, c)$. Count of term t in the entire doc set.

3. $\mathrm{P}_{us}(\mathrm{t}|d) = \frac{c(t,d)}{|d|}$. Relevance between term t and doc d.

4. $\mathrm{P}_{s}(\mathrm{t}|\mathrm{d}) = \frac{c(t,d)+\mu \mathrm{c}(\mathrm{t},\mathrm{c})}{|d|+\mu}$. Relevance between term t and doc d after dirichlet smoothing. μ in the formula is the parameter of dirichlet smoothing, fixed to 5000 in this case.

5. $idf(t) = \ln\frac{D}{D_W}$. The inverse document frequency for term t. D is the total count of all docs and D_w is the count of docs containing term t.

In order to calculate the values, all the term in each doc are count and stored into database. By summing up the term counts in each doc, the $c(t,c)$ can also be calculated. All the values mentioned above can be calculated now. They will be stored in the database to avoid repeated calculation.

6.2 $P_{us}(t|d)$ Updating Details

The $\mathrm{P}_{us}(\mathrm{t}|d)$ only updates when the query changes. By updating the $\mathrm{P}_{us}(\mathrm{t}|d)$ of terms in query, the score of docs for query will be adjusted. In a certain query step, $\mathrm{P}_{us}(\mathrm{t}|d)$ may be increased, decreased or not changed. Refer the current query as q_i, the previous query as q_{i-1} and the last query result doc set as $\mathrm{D}_{\mathrm{i}-1}$. Let q_{theme} be the longest common substring of q_i and q_{i-1}. $+\Delta\mathrm{q}$ and $-\Delta\mathrm{q}$ are defined as follows:

$$+\Delta\mathrm{q} = q_i - q_{theme}$$

$$-\Delta\mathrm{q} = q_{i-1} - q_{theme}$$

$+\Delta\mathrm{q}$ contains the new terms in q_i while $-\Delta\mathrm{q}$ contains the removed terms in q_{i-1}.

In the first query step, $\mathrm{P}_{us}(\mathrm{t}|d)$ is already initialized and will not change. When $\mathrm{t} \notin \mathrm{D}_{\mathrm{i}-1}$ and $\mathrm{t} \in -\Delta\mathrm{q}$, $\mathrm{P}_{us}(\mathrm{t}|d)$ will not change, where d is the doc containing term t.

One possible case is $\mathrm{t} \in \mathrm{D}_{\mathrm{i}-1}$. $\mathrm{t} \in -\Delta\mathrm{q}$ means the term is not useful, so $\mathrm{P}_{us}(\mathrm{t}|d)$ should be reduced. Meanwhile, $\mathrm{t} \in +\Delta\mathrm{q}$ denotes that term t is already concerned, in order to lead the new terms contribute more value for relevance between query and doc, $\mathrm{P}_{us}(\mathrm{t}|d)$ of old terms should relatively decrease. Additionally, the more frequent term t is in $\mathrm{D}_{\mathrm{i}-1}$, the more should be cut back from $\mathrm{P}_{us}(\mathrm{t}|d)$. Refer d^*_{i-1} as the doc which has the biggest relevance with q_{i-1}. Using logarithmic function to avoid float calculation error, the new $\mathrm{P}_{us}(\mathrm{t}|d)$ can be calculated as follow:

$$\log P_{us}(t|d)_{new} = \left(1 - P_{us}(t|d^*_{i-1})\right) \log P_{us}(t|d)$$

$$d^*_{i-1} = \underset{d_k \in D_{i-1}}{\operatorname{argmax}} P(q_{i-1}|d_k)$$

$$\mathrm{P}(q_{i-1}|d_{i-1}) = 1 - \prod_{t \in q_{i-1}} (1 - \mathrm{P}_{us}(\mathrm{t}|d_{i-1}))$$

When $\mathrm{t} \notin \mathrm{D}_{\mathrm{i}-1}$ and $\mathrm{t} \in +\Delta\mathrm{q}$, term t is not mentioned before. $\mathrm{P}_{us}(\mathrm{t}|d)$ should be increased to denote the prefer over the new term. In order to avoid a violently increase over $\mathrm{P}_{us}(\mathrm{t}|d)$, $idf(t)$ is better in the update formula. Then the new $\mathrm{P}_{us}(\mathrm{t}|d)$ can be calculated as follow this time:

$$\log P_{us}(t|d)_{new} = (1 + idf(t)) \log P_{us}(t|d)$$

Another case for increasing $\mathrm{P}_{us}(\mathrm{t}|d)$ is $\mathrm{t} \in q_{theme}$. In this case, term t usually be a topic or frequent word, whereas not frequent in the whole doc set. Therefore, use $1 - \mathrm{P}(\mathrm{t}|d^*_{i-1})$ here to substitute idf(t):

$$\log \mathrm{P}_{\mathrm{us}}(\mathrm{t}|\mathrm{d})_{\mathrm{new}} = (1 + (1 - \mathrm{P}_{\mathrm{us}}(\mathrm{t}|\mathrm{d}^*_{\mathrm{i}-1}))) \log \mathrm{P}_{\mathrm{us}}(\mathrm{t}|\mathrm{d})$$

After the update process finish, the information refinement process will use the new $\mathrm{P}_{\mathrm{us}}(\mathrm{t}|\mathrm{d})$ to calculate the score of each doc for q_i and get the best search results.

7. Termination Condition

Obviously, the search process must finish in finite iterations. There are two different terminations. If the process has iterated for more than 20 times, then the result is considered to be convergent. In the search process, only the top 30 documents are considered to be related with query. If the DR process fails to generate new query from all the top 30 documents, even if iteration time is less than 20, the whole search process is still considered to be fail. That is, 20 times of iteration or fail to generate new query from top 30 documents are the termination condition of search process.