OKSAT at NTCIR-11 Temporalia - Plural Sets of Search Terms for a Topic -

Takashi SATO, Shingo AOKI {sato,aoki}@ss.osaka-kyoiku.ac.jp (Osaka Kyoiku University)

[1] Introduction

- The need of information retrieval including temporal information is increasing nowadays.
- NTCIR-11 Temporal Information Access (Temporalia) focuses on this problem.
- OKSAT participated in Temporal Information Retrieval (TIR) subtask of NTCIR-11 Temporalia.
- We describe our system and techniques used.
- Following experimental results, we show an example to discuss the effectiveness of our methods.

[2] Our Approach - Searching and Scoring

- We used only <text> tag among tags.
- Because titles in <title> tag are short and we observed that search words were not often used in the title even if a document related to the topic.
- About <T> tags in <text>, we could not use these values in order to distinguish time factor of subtopics.
- Using index made by text in <text> tag, we retrieved the search terms (1), (2) and (3) prepared above.
- Then we scored and ranked output document id by using probabilistic model.

[2] Our Approach - Removal of Tags from Corpus

- From temporalia corpus, we extracted the text surrounded by title tag (<tag name="title">...</tag>) in <meta-info> tag.
- Using 'temporalia_solrify.pl' prepared by task organizer, the text surrounded by text tag (<text>...</text>) in which all tags were removed was extracted.
- In addition, the val of T tags (<T val="...">) in the text tag were extracted also.

[2] Our Approach - How to Make Search Terms

- We prepared the following three types of search terms.
  - (1)From topic file, we extracted words from title and each subtopic of the topic, and then they were filtered by stop word list.
  - (2)We searched the internet (Wikipedia and Google) by words from (1), and then we simply added most common words in the search results.
  - (3)We classified words from internet search of (2) if possible. Then we added each classified group to (1). As a result, we got plural sets of search terms for a subtopic.

[3]Structure of Gram Base Index

- Gram base indices are known as index structures, which enable arbitrary string search.
- Grams consist of strings (character sequences), which start every character in a text.
- The length of strings is less than 3 (such as 1-gram or 2-gram) in common cases.
- Our grams are longer than 4-grams in average by encoding grams in w_g byte. w_g is set 6 in this task.
- At first, characters are coded in varying length bit in accordance with their frequency of appearance.
- Then they are stuffed within w_g. Figure 1 shows the stuffing of coded characters.

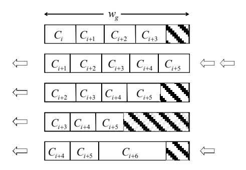
w_g
C_i C_{i+1} C_{i+2} C_{i+3}
C_{i+1} C_{i+2} C_{i+3} C_{i+4} C_{i+5}
C_{i+2} C_{i+3} C_{i+4} C_{i+5}
C_{i+3} C_{i+4} C_{i+5}
C_{i+4} C_{i+5} C_{i+6}

Figure 1. Stuffing of Coded Characters in a gram.

Table 1. Code Table for Gram Coding

C	L	CODE	C	L	CODE	C	L	CODE	C	L	CODE
_	11	0	9	8	40	J	9	1	T	3	5
0	8	1	A	4	1	K	7	21	U	5	19
1	8	41	B	6	30	L	5	d	V	7	1
2	8	81	C	5	9	M	5	11	W	6	31
3	8	82	D	5	C	N	4	2	X	8	80
4	8	83	E	3	7	O	4	7	Y	6	11
5	8	84	F	6	2	P	6	3	Z	10	1
6	8	85	G	6	1	Q	11	1			
7	8	86	H	4	9	R	4	d			
8	8	87	I	4	3	S	4	5			

```
43: KFWB_NE
42:  FWB_NEW
40:   WB_NEWS
34:    B_NEWS
39:     _NEWS_
47:      NEWS_TALK
43:       EWS_TALK
40:        WS_TALK
45:         S_TALK_
41:          _TALK_
46:           TALK_98
43:            ALK_98
47:             LK_980
42:              K_980
46:               _980_
35:                980_
27:                 80_
19:                  0_
11:                   _
```

An example of gram coding (each row is gram length in bit and characters in a gram) when text is 'KFWB_NEWS_980_' a part of the tile of the first document of the corpus.

Figure 2. Example of Grams.

[3]Structure of Gram Base Index – Cnt'd

- Figure 3 shows a data structure of gram index. An index has three parts called root, leaf and locator.
- Grams are sorted and stored in secondary storage as leaf.
- There is a pointer from a gram in leaf to a bucket, which stores the document numbers where the string corresponding to the gram is found and the count of the gram appeared in the document.
- Locator, which is stored in secondary storage, is collection of buckets.
- Root, which is put in main memory when searching, is a wide range map of grams.

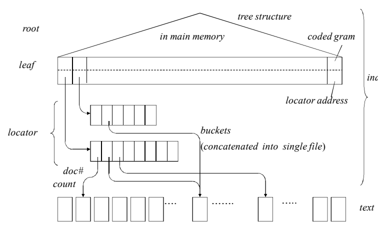

Figure 3. Structure of Gram Index.

[3]Structure of Gram Base Index – Cnt'd

- The search algorithm is explained in terms of three cases according to the relation between the length of search key word (l_k) and gram length (l_g).
- Figure 4 (a), (b) and (c) show how to follow the pointers in leaves and locators when $l_k = l_g$, $l_k < l_g$ and $l_k > l_g$ respectively.
- Since the buckets of the locator are stored sequentially, they are drawn in one box and separated by double lines.

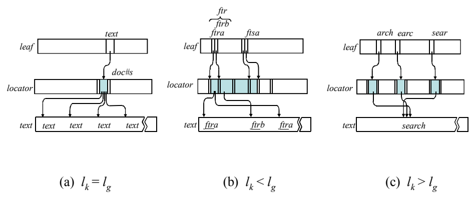

Figure 4. Index Search.

[4]Experimental Results - Indexing

- We made gram based indices from text surrounded by <text> and <title> tag of corpus.
- Table 2 shows specifications of computer we used. And Table 3 shows statistics of indices.

Table 2. Specifications of computer

CPU	Intel Core i5-4430@3.0GHz 4C/4T
MEM	8GB, DDR3-1600
O S	FreeBSD 8.4, 64bit
HDD	1TB, SATA 6GB/s, 64MB Cache

Table 3. Statistics of Indices

	title	**text**
data size (MB)	0.170	8.90
index size (MB)	0.489	18.9
time (min.)	2.08	153

[4]Experimental Results - Runs

- Using three types of search terms we made the following runs.
  - OKSAT-TF01 : type (1) of slide[2]
  - OKSAT-TF02 : type (2) of slide[2]
  - OKSAT-TF03 : type (3) of slide[2]
- Table 4 shows time (searching and scoring in minutes) and AP (mean average precision) of our submitted runs.
- Because task organizers have a shallow pooling at document 20, P@20 and nDCG@20 are shown in this table also.

Table 4. Time and MAP of Submitted Runs

RUN ID	Time	AP	P@20	nDCG@20
OKSAT-T-TF01	54	0.2138	0.5828	0.4566
OKSAT-T-TF02	83	0.2063	0.5843	0.4551
OKSAT-T-TF03	25	0.1971	0.5723	0.4391

- The CPU of our computer has 4 cores and can process 4 threads simultaneously.
- OKSAT-TF01 and OKSAT-TF02 were executed simultaneously by 2 threads each.
- On the other hand OKSAT-TF03 was executed alone by 4 threads.
- So, OKSAT-TF03 was executed twice as fast as OKSAT-TF01 and OKSAT-TF02.

[4]Experimental Results - Temporal Class Analysis

- Table 5 shows P@20 of each temporal query classes.
- The atemporal query class was better than other query classes.
- The past query class was difficult for our group.

Table 5. P@20 of Each Temporal Query Classes

RUN ID	atemporal	future	past	recency
OKSAT-T-TF01	0.6460	0.5740	0.5050	0.6060
OKSAT-T-TF02	0.6260	0.6030	0.5070	0.6010
OKSAT-T-TF03	0.5830	0.6190	0.4870	0.6000

[4]Experimental Results - Topic-based Analysis

- We show poorly and better performing topic examples about expansion of search terms (i.e. from OKSAT-FT-01 to OKSAT-FT-02).
  - 026a: We expanded search term from 'passive smoke' to 'smoke' and 'smoking'. This made P@20 fall down from 0.9500 to 0.5500.
  - 045f: We expanded search term from 'Papacy' to 'Pope'. This raised P@20 from 0.0500 to 0.4000.
- However, the effect of search term expansion was not similar about other temporal classes.
- So the effect of word expansion was sensitive to temporal classes.

[4]Experimental Results - Examples of Plural Sets of Search Terms

- In temporal information retrieval, there are cases when plural events should be searched.
- Searching by only one set of search terms is not enough, in those cases.
- So, we made our system to handle plural sets of search terms for a subtopic.
- Concretely, we explain an effect of plural sets of search terms about subtopic id 001p (subtopic of type past of topic id 001).
- We added the words extracted from title of topic id 001 and its subtopic 001p to the words after '+' below.
  - 001p-1 (+ 'Tokyo subway',' Tokyo', 'subway', 'sarin')
  - 001p-2 (+ 'Hiroshima nuclear')
  - 001p-3 (+ 'Tohoku earthquake')
- Then three sets of search terms were made.
- Table 6 shows relations of the term sets above and the AP (average precision), P@20 and nDCG@20 of their runs.

[4]Experimental Results - Examples of Plural Sets of Search Terms - Cnt'd

Table 6. Term Set and AP etc.

Term Set	AP	P@20	nDCG@20
001p-1	0.0807	0.2355	0.2328
001p-2	0.0911	0.3009	0.3716
001p-3	0.01849	0.5319	0.4958

- We merged these three runs into one by two ways.
- One is merging by score order of document and the other is merging by rotation (first we gather top of three runs, next we gather second of three runs and so on.)

[4]Experimental Results - Examples of Plural Sets of Search Terms - Cnt'd

- Table 7 shows merge type and AP, P@20 and nDCG@20.
- In this table, 'base words' stands for no words from the internet added i.e. words extracted from title of topic id 001 and its subtopic 001p only.
- From Table 7, we observe that the AP, P@20 and nDCG@20 of merging by rotation is higher than those of base words only and merging by rotation.

Table 7. Merge Type and AP etc.

Merge Type	AP	P@20	nDCG@20
base words	0.1849	0.5139	0.4958
by score	0.1368	0.4205	0.4465
by rotation	0.2669	0.6362	0.5589

[5] CONCLUSIONS

- OKSAT submitted three runs for Temporal Information Retrieval (TIR) subtask of NTCIR-11 Temporalia).
- In third run, we prepared plural sets of search terms for a subtopic using words added by the internet search.
- Analyzing experimental results, we observe the effectiveness of using plural sets of search terms for a subtopic.