K2U at TREC 2014 KBA Track

Shun Kawahara*, Kazuhiro Seki†, Kuniaki Uehara*
*Graduate School of System Informatics
Kobe University
†Faculty of Intelligence and Informatics
Konan University
Email: kawahara@ai.cs.kobe-u.ac.jp

I. Introduction

Kobe University and Konan University (K2U) collaborated on the vital filtering task of the 2014 TREC KBA track. This paper describes our proposed system developed on the distributed and fault-tolerant realtime computation system, *Apache Storm*, and reports the results obtained for our submitted runs.

The remainder of this paper is structured as follows: Section II briefly introduces Apache Storm and its components, Section III describes our proposed system for the vital filtering task, Section IV reports and discusses the results for our submitted runs, and Section V concludes this paper with a brief summary.

II. Apache Storm

Apache Storm[1] is a distributed realtime computation system, processing unbounded streams of data. To use Storm, one needs to define "topologies". A topology is a graph of computation and each node in a topology has processing logic and edges between nodes indicate how data should be passed around between nodes.

There are two types of nodes, called "spouts" and "bolts". A spout is a source of streams (sequences of tuples). In case of the KBA track, a spout would read document data from the provided KBA corpus and emit them as a stream. A bolt receives any number of input streams, does some processing, and may emit new streams. For the KBA track, bolts would determine whether inbound documents from the streams are relevant. Each node in a Storm topology executes in parallel and one can specify how much parallelism he/she wants for each node.

III. Our system

A. Overview

Figure 1 depicts the topology of our proposed system for the vital filtering task, where corpus spout reads documents from the KBA corpus and sends them to surface form name (sfn) filter bolt, followed by relevant filter bolt, vital filter bolt, and so on. These bolts process the input stream of documents in parallel for a given target entity. The following paragraphs provide brief descriptions of the each type of nodes of the topology.

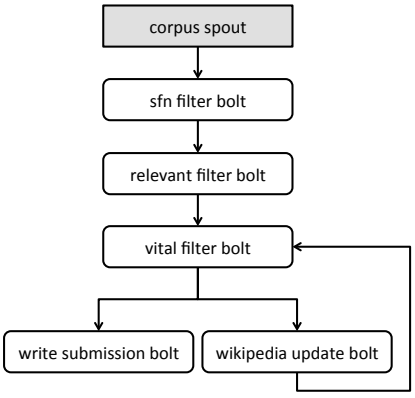

Fig. 1. Topology of our system developed on Storm.

First, corpus spout reads individual documents from the input stream (the StreamCorpus 2014 for this task) and sends them to sfn filter bolts. Note that each document is preprocessed as described in Section III-B.

Sfn filter bolt checks if each input document contains any surface form names of a given target entity. Only the documents containing the surface form name(s) are sent to the succeeding processes. For this study, `target_id` and `canonical_name` fields in the profiles[2] provided to the track participants are used as surface form names. In addition, for those entities which have Wikipedia URLs in the `external_profile` field, redirect information extracted from the Wikipedia dump on 1/4/2012[3] are also utilized.

Relevant filter bolt judges whether an input document is relevant (i.e., vital or useful) or not. If judged to be relevant, the document is sent to vital filter bolt. For this purpose, a language model built from a set of non-relevant documents for the target entity is used. More details are found in Section III-C.

Then, vital filter bolt judges if an input document is vital or useful. The predicted class and the `document_id` are sent to write submission bolt. In addition, the document judged to be vital (only) is sent to wikipedia update bolt (see Section III-E for more details). The classification between vital or useful is based on another language model built from the Wikipedia article for the target entity. The details are described in Section III-D.

[1] https://storm.incubator.apache.org/

[2] trec-kba-2014-07-11-ccr-and-ssf.profiles.yaml

[3] http://s3.amazonaws.com/aws-publicdatasets/trec/kba/enwiki-20120104/index.html

Lastly, write submission bolt collects and outputs the results of the preceding vital filter bolts for TREC run submission. The following sub-sections give more details of the main components of the system as well as document preprocessing.

B. Document Preprocessing

This study uses a filtered subset of the 2014 StreamCorpus (the kba-streamcorpus-2014-v0_3_0-kba-filtered corpus). The filtered corpus was generated and provided by the track organizers.[4]

For efficiency, we further process the filtered corpus as follows:

- Discard documents not containing surface form names of the target entities.
- Remove duplicate documents with the same `document_id`.
- Uncapitalize words.
- Remove stop words and symbols.
- Apply the Porter stemmer [3].
- Build a document language model $p(w|d)$ for each document d with Dirichlet smoothing [5]:

$$p(w|d) = \frac{c(w,d) + \mu p(w|\mathcal{C})}{|d| + \mu} \quad (1)$$

where Google-Ngram[5] (unigrams) is used to calculate the background language model $p(w|\mathcal{C})$.

C. Relevant filter bolt

Relevant filter bolt judges whether an input document is relevant (i.e., vital or useful) or not. For this purpose, we use a language model built from non-relevant documents for the target entity in question adopting the concept of MultiNeg [4], which has been shown effective for difficult queries (topics) where the search results are poor.

MultiNeg is a model to improve ad-hoc retrieval by negative relevance feedback, which takes advantage of (pseudo) feedback of non-relevant documents. More specifically, according to the similarity between the language models built from non-relevant documents and an input document, the relevance score of the document is adjusted. The non-relevant documents here mean those irrelevant to search intention within the initial search result.

For example, consider the case where a user would like to search for information regarding Apple Inc. and uses a query "apple". The search results would contain documents regarding apples (fruit), which are considered non-relevant documents in this case. MultiNeg builds a language model (called a negative model) $\Theta = \{\theta_1, ..., \theta_f\}$ for each of such irrelevant documents $L = \{l_1, ..., l_f\}$ using the standard EM algorithm [1]. The f negative models are used to compute KL-divergence D with respect to document d and the minimum, i.e.,

$$\min(\bigcup_{i=1}^{f}\{D(\theta_i||\theta_d)\}), \quad (2)$$

discounts the relevance score of d computed based on the KL-divergence retrieval model [2]. This study uses the minimum KL-divergence to filter out irrelevant documents.

To estimate the negative models θ_i, our system uses non-relevant documents in the training time range of the KBA corpus ("KBA TTR subset" for short). The training time range is individually defined for each target entity. We consider documents which contain surface form name(s) of a target entity but do not have a "vital" or "useful" label as non-relevant documents. Note that if the number of documents containing a surface form name is too large (>100k for our experiments to be reported), the name is unlikely informative and thus is not utilized. Also, if a target entity has no non-relevant document, relevant filter bolt is disabled for the entity. In applying the filter, if the lowest distance (Eq. (2)) for an input document d is smaller than a predefined threshold t_r, the document is considered irrelevant and discarded. The threshold t_r is set to the smallest value among the thresholds based on which vital or useful documents in the KBA TTR subset would be judged to be relevant.

D. Vital filter bolt

Vital filter bolt judges if an input document is vital or useful using a language model built from a Wikipedia article corresponding to a target entity. Similarly to the extraction of redirect information described above, we first obtain Wikipedia articles in the Wikipedia dump on 1/4/2012 for entities which have Wikipedia URLs in the `external_profile` fields in their profiles. The same preprocessing described in Section III-B is applied to the extracted articles. Also, the vital or useful documents in the KBA TTR subset are extracted for each target entity as they are the past, known information regarding the entity. Using the Wikipedia articles and extracted vital/useful documents, a unigram language model is created for each target entity in advance. Note that only vital/useful documents are used for topics without Wikipedia URLs.

In filtering, the similarity between the language models and input documents are computed based on the KL-divergence retrieval model, from which the document is judged to be vital or not (i.e., useful). We explore two opposite criteria in the judgment. One is to decide that the document is vital if the similarity is lower than a predefined threshold t_v. The rationale behind is that a document more different from a Wikipedia article would contain more, and possibly relevant, information not described in the article. The other is to decide that, conversely, the document is vital if the similarity is greater than the threshold. The assumption here is that the Wikipedia article, and the language model built from it, would capture some features characteristic to vital documents and a document similar to it would be also vital.

The threshold t_v is set for each entity using the vital and useful documents in the KBA TTR subset such that F_1 score is maximized based on their similarity scores with the language model.

[4] http://s3.amazonaws.com/aws-publicdatasets/trec/kba

[5] http://googleresearch.blogspot.jp/2006/08/all-our-n-gram-are-belong-to-you.html

E. Wikipedia update bolt

Wikipedia update bolt receives a document which is judged as vital in the preceding vital filter bolt and updates the Wikipedia language model. The updated model is sent back to vital filter bolt and will be used afterwards. This update simulates the editing of Wikipedia articles in the light of new information related to the target entities.

IV. Evaluation

Table I summarizes the three runs submitted by our group, where "exact_match" simply treated the documents which went through sfn filter bolt as vital. That is, the documents containing surface form name(s) of the target entities were judged as vital. This run is considered as the baseline. The other runs, "ccr_03" and "ccr_08", are results by our developed system. Specifically, "ccr_03" treated the documents which have relevance scores lower than a threshold as vital at vital filter bolt, while "ccr_08" treated those with similarity scores greater than the threshold as vital (see Section III-D). In addition, we limited the vocabulary of ccr_08 for efficiency based on chi-squared tests using vital vs. useful document sets.

TABLE I. Summary of our submitted runs.

run_id	Description
exact_match	Documents which went through sfn filter were judged as vital.
ccr_03	Documents with low similarity scores were judged as vital.
ccr_08	Documents with high similarity scores were judged as vital.

For this experiment, we set the number of bolts in the topology as shown in Table II.

TABLE II. Number of bolts in the topology.

Bolt	Number
sfn filter bolt	1
relevant filter bolt	4
vital filter bolt	1
write submission bolt	1
wikipedia update bolt	1

The official results are shown in Table III and Table IV.

TABLE III. Official results for vital filtering task (vital only).

run_id	Prec.	Recall	F_1	SU
exact_match	0.288	0.953	0.442	0.266
ccr_03	0.287	0.845	0.429	0.266
ccr_08	0.300	0.609	0.402	0.299

TABLE IV. Official results for vital filtering task (vital+useful).

run_id	Prec.	Recall	F_1	SU
exact_match	0.706	0.713	0.709	0.805
ccr_03	0.706	0.708	0.707	0.803
ccr_08	0.706	0.708	0.707	0.803

In most cases, the baseline (exact_match) produced the best performance. This is expected since the official evaluation ignores unannotated documents in computing precision. Our filters intended to reduce false positives so as to improve precision but their effects will not be properly evaluated by the official evaluation if the filtered-out documents are not annotated.

To alleviate the potential problem, we treated unannotated documents as true negatives and re-evaluated the submitted runs. Table V and Table VI show the results, which were obtained by kba-scorer[6] with "`--unannotated-is-true-negative`" option. As shown, both ccr_03 and ccr_08 improved the baseline, suggesting the effects of our language model-based filters.

TABLE V. Performance when unannotated documents are treated as true negatives (vital only).

run_id	Prec.	Recall	F_1	SU
exact_match	0.099	0.953	0.179	0.070
ccr_03	0.112	0.845	0.197	0.085
ccr_08	0.129	0.609	0.213	0.143

TABLE VI. Performance when unannotated documents are treated as true negatives (vital+useful).

run_id	Prec.	Recall	F_1	SU
exact_match	0.191	0.666	0.297	0.283
ccr_03	0.209	0.655	0.317	0.308
ccr_08	0.209	0.655	0.317	0.308

For the "vital only" setting, ccr_08 yielded the best result on average as shown in Table V. However, statistical significance was found only between exact_match and ccr_03 at the 5% significance level by pair-wise t-test. This is due to the larger variance of ccr_08 in the improvement over exact_match as contrasted in Figure 2 for ccr_03 and Figure 3 for ccr_08. While ccr_08's improvement is more noticeable, it also showed strong negative effects for some entities, including "Lizette_Graden" and "Corisa_Bell". On the other hand, ccr_03 improved over the baseline for the majority of the entities, even though around a half of them are marginal.

For the "vital+useful" setting, vital and useful documents are not distinguished and thus the performance of ccr_03 and ccr_08 are exactly the same. In other words, this experiment focuses on the effectiveness of relevant filter bolt only. The improvement from exact_match were found statistically significant at the 1% significance level, showing the effectiveness of the language model built from irrelevant documents.

Then, we examined the effect of the updates of the Wikipedia language models. Table VII compares the cases with/without the updates of the language models. We observed that there were slight improvement in the performance in F_1 for both ccr_03 and ccr_08 with the model updates.

TABLE VII. Effects of Wikipedia language model updates.

run_id	Prec.	Recall	F_1	SU
ccr_03 (w/o updates)	0.109	0.853	0.193	0.084
ccr_03 (with updates)	0.112	0.845	0.197	0.085
ccr_08 (w/o updates)	0.143	0.407	0.211	0.176
ccr_08 (with updates)	0.129	0.609	0.213	0.143

In further investigation, we also noted that the number of irrelevant documents in building the language models was largely different from entity to entity. Figure 4 shows a

[6]https://github.com/trec-kba/kba-scorer

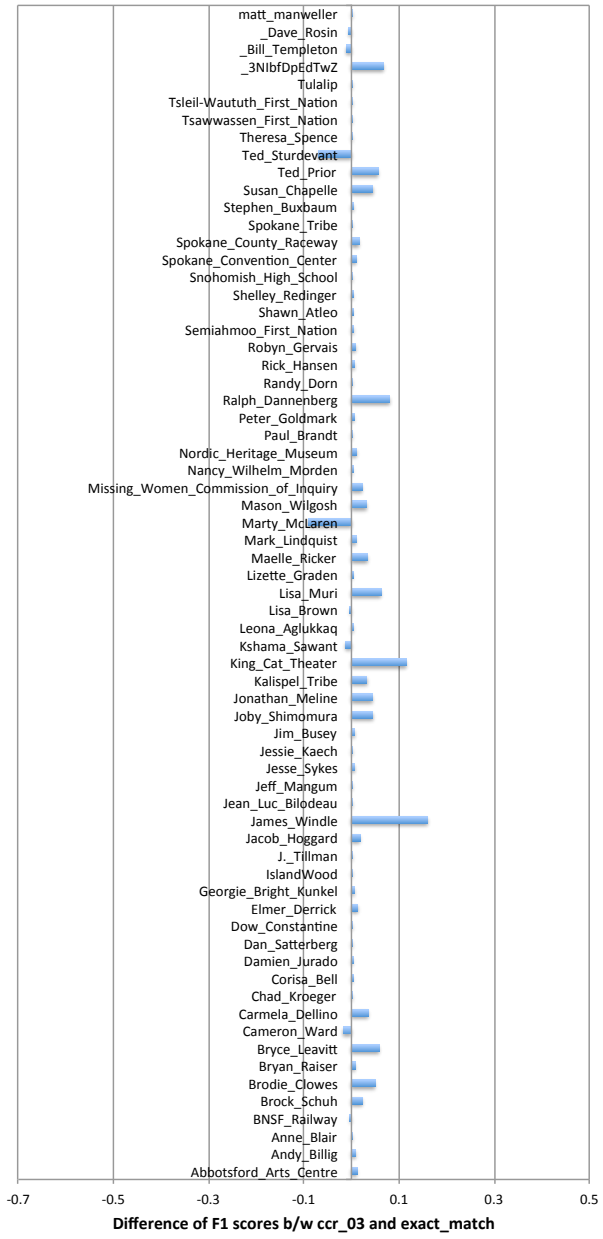

Fig. 2. Per-topic difference of F_1 scores between ccr_03 and exact_match for each entity.

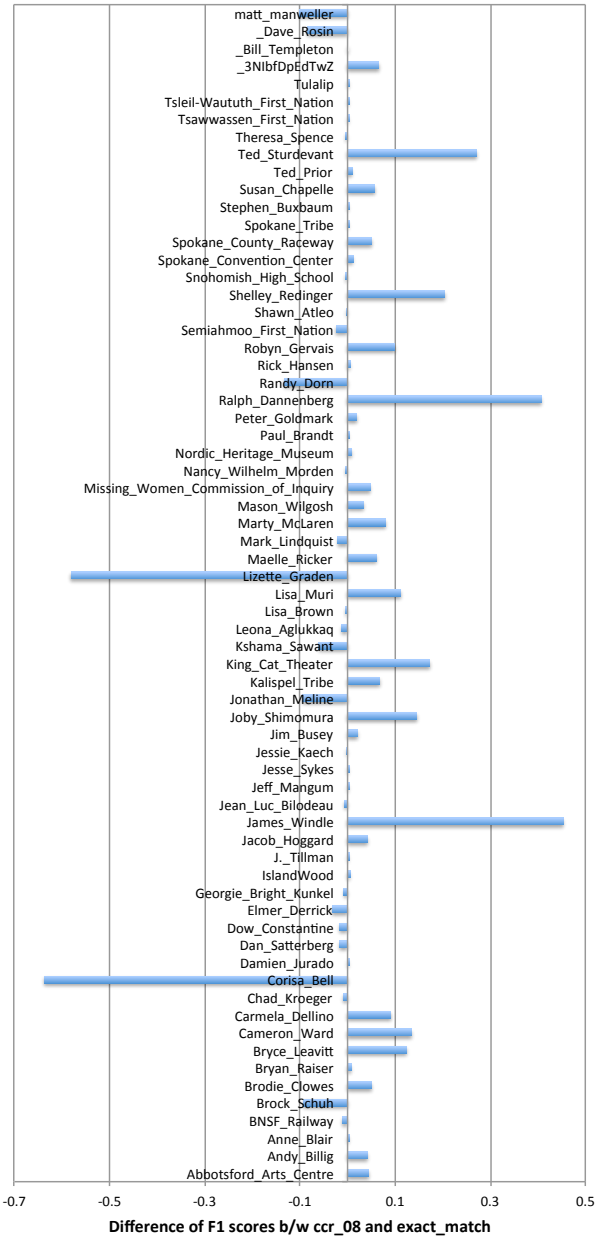

Fig. 3. Per-topic difference of F_1 scores between ccr_08 and exact_match for each entity.

scatter plot between the difference of F_1 scores of ccr_03 and exact_match and the log number of irrelevant documents. The plot suggests that the number of documents may have some influence to the performance improvement. We will plan to investigate and leverage the relation in future work.

V. Conclusion

This paper described our proposed system developed for the TREC 2014 KBA vital filtering task and reported the obtained results. Our system employed language model-based approach and used irrelevant documents for identifying relevant (vital or useful) documents and Wikipedia articles for further identifying vital documents, where the Wikipedia language models were updated given newly identified vital documents. The approach was empirically found effective, improving our baseline using only string matching of surface form names of target entities.

There is a plenty of work needed to consider to improve our system. The current system utilizes only irrelevant documents but the feedback for relevant documents would be also beneficial. Also, we found that ccr_03 and ccr_08 were somewhat complementary and conjecture that they could be integrated. Lastly, we would like to take advantage of the rich information found in the KBA corpus as well as Wikipedia.

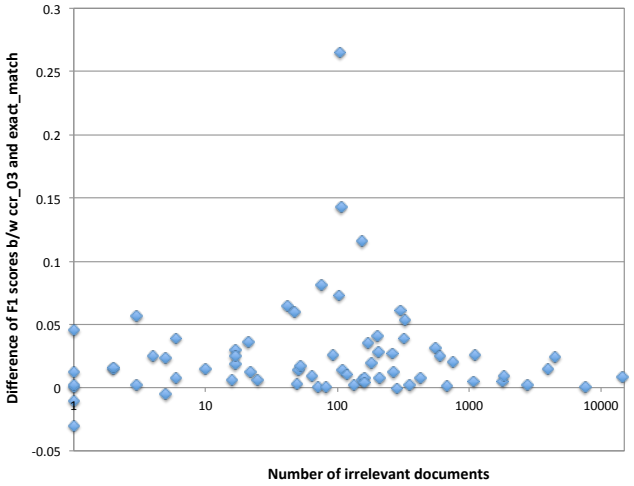

Fig. 4. Relation between the number of irrelevant documents and the difference of F_1 scores between ccr_03 and exact_match.

ACKNOWLEDGMENT

The authors would like to thank Sayaka Kitaguchi at Kobe University for processing the KBA corpus. This work is partially supported by JSPS KAKENHI Grant Numbers 25330363 and MEXT, Japan.

REFERENCES

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society, series B*, vol. 39, no. 1, pp. 1–38, 1977.

[2] J. Lafferty and C. Zhai, "Document language models, query models, and risk minimization for information retrieval," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001, pp. 111–119.

[3] M. F. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980.

[4] X. Wang, H. Fang, and C. Zhai, "A study of methods for negative relevance feedback," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008, pp. 219–226.

[5] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to Ad Hoc information retrieval," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001, pp. 334–342.