BIT and MSRA at TREC KBA CCR Track 2013[*]

Jingang Wang
School of Computer Science
Beijing Institute of Technology
Beijing, China
bitwjg@bit.edu.cn

Dandan Song[†]
School of Computer Science
Beijing Institute of Technology
Beijing, China
sdd@bit.edu.cn

Chin-Yew Lin
Knowledge Mining Group
Microsoft Research Asia
Beijing, China
cyl@microsoft.com

Lejian Liao
School of Computer Science
Beijing Institute of Technology
Beijing, China
liaolj@bit.edu.cn

ABSTRACT

Our strategy for TREC KBA CCR track is to first retrieve as many vital or documents as possible and then apply more sophisticated classification and ranking methods to differentiate vital from useful documents. We submitted 10 runs generated by 3 approaches: question expansion, classification and learning to rank. Query expansion is an unsupervised baseline, in which we combine entities' names and their related entities' names as phrase queries to retrieve relevant documents. This baseline outperforms the overall median and mean submissions. The system performance is further improved by supervised classification and learning to rank methods. We mainly exploit three kinds of external resources to construct the features in supervised learning: (i) entry pages of Wikipedia entities or profile pages of Twitter entities, (ii) existing citations in the Wikipedia page of an entity, and (iii) burst of Wikipedia page views of an entity. In *vital + useful* task, one of our ranking-based methods achieves the best result among all participants. In *vital only* task, one of our classification-based methods achieve the overall best result.

1. INTRODUCTION

Knowledge Bases (KBs), such as Wikipedia, have shown great power in many applications including query answering, entity retrieval and entity linking. With the explosion of information on the web, it becomes critical to detect relevant documents and assimilate new information to entities in KBs in a timely manner. However, most KBs are maintained manually by volunteer editors, which are hard to keep up-to-date because of the limit number of editors and the huge volume of entities in KBs. [6] indicates that the median time lag between the publication date of the cited articles and the date of the citations created in Wikipedia is nearly one year. Moreover, some esoteric entities in KBs do not attract enough attentions from the editors. It makes the maintenance more challenging. This gap could be reduced if relevant documents could be automatically found as soon as they are published and then recommended to the editors.

Cumulative Citation Recommendation(CCR) was introduced by Text REtrieval Conference(TREC) Knowledge Base Acceleration (KBA) track in 2012 to address this problem. A CCR system should filter candidate documents for a given set of entities from a time-ordered stream corpus. CCR track continues this year, including diversified entities and larger stream corpus. The target entity set is composed of 121 entities from Wikipedia and 20 entities from Twitter.

KBA 2013 has augmented the stream corpus of KBA 2012, covering the time period from Oct. 2011 to Feb. 2013. Each document in the stream corpus contains several fields. Table 1 lists fields used in our work.

Table 1: Document fields used in our CCR system

Field	Description
stream_id	an unique identifier of the document
clean_visible	plain text content of the document
source	source of the document
timestamp	a timestamp measured in seconds since the 1970 epoch

A CCR system is fed with the stream corpus in chronological order and outputs a confidence score in the range of (0, 1000] for each document-entity pair. The confidence score represents the relevance level between the document and the target entity. A cutoff value is varied from 0 to 1000 (stepsize = 10 in this paper) and the documents with scores above the cutoff are treated as positive instances by the system. Correspondingly, the documents with scores below the cutoff are negative instances. There are two measures defined by TREC KBA 2013 to evaluate the system performance: (i) $max(F(avg(P), avg(R)))$ and (ii) $max(SU)$. SU(Scaled utility) is a metric introduce in [9] to evaluate the ability of

[*] This work was done when the first two authors were visiting the Knowledge Mining Group of Microsoft Research Asia
[†] Corresponding Author

a information filtering system to separate relevant and irrelevant document. Given a cutoff, we could calculate P, R, F and SU respectively for each entity and obtain the macro-average values of all entities.

There are two sub tasks of CCR in KBA 2013: (i) *vital only:* treating only vital documents as positive instances and non-vital as negative instances, and (ii) *vital + useful:* accepting both vital and useful documents as positive instances.

We submitted 10 runs to KBA CCR Track 2013, including 2 query expansion runs, 2 classification-based runs and 6 ranking-based runs. Query expansion runs, as our baselines, outperform the median and mean of all 140 submissions. In *vital + useful* task, our ranking-based run with burst feature achieves the best result. While in *vital only* task, classification runs are better than ranking-based runs in general, and one of them is the overall best run.

The rest of this paper is organized as follows: Section 2 introduces a pre-processing step to reduce the size of stream corpus. Next, we present our approaches in Section 3. Section 4 lists the features used in our supervised approaches in detail. Finally, we summarize the results of our submissions and conclude some insights in Section 5.

2. PRE-PROCESSING

Before relevance analysis for document-entity pairs, our CCR system contains a pre-processing step, including indexing and filtering.

2.1 Indexing

In order to process the huge stream corpus efficiently, we utilize *ElasticSearch* to index the whole stream corpus. *ElasticSearch* is an open-source, Lucene-based text search engine[1]. We only care about 4 fields of each document: *stream_id*, *clean_visible*, *timestamp* and *source*. Table 1 describes the meanings of these fields.

2.2 Filtering

It's too time-consuming and laborious to process all the documents in the stream corpus for each entity. According to the annotation analysis of KBA 2012, none of the document with zero mention of the target entity is annotated as central, and there are only 0.4% of the documents with zero mention of the target entity have been labeled as relevant [6]. So we filter the index through retaining as many relevant documents as possible. We construct a high-recall phrase query assuring that the retrieved documents should mention the target entity at least once, either exactly entity name or surface forms. Therefore, The prerequisite of filtering is expanding enough surface forms for each target entity.

For each target entity from Wikipedia, we extract the redirect[2] names as its surface forms. For example, *Geoffrey E. Hinton*, who is a computer scientist in machine learning, owns the following redirect names in Wikipedia: *Geoffrey Hinton*, *Geoff Hinton*, *Geoffrey E. Hinton*, *Geoffrey Everest Hinton*.

For each target entity from Twitter, we add its display name into its surface form set. For the entity *@AlexJoHamilton*, we could acquire its display name *Alexandra Hamilton* via Twitter's APIs.

We construct a *matchPhraseQuery*[3] with the target entity name and all its surface forms together and search against the index. Only the hit documents are processed in the following steps.

After the pre-processing step, the number of documents retained in the stream corpus decreases from 442,325,966 to 77,589. This indeed makes our CCR system more efficient.

3. APPROACHES

We have tried 3 families of methods in our submissions to KBA 2013, including query expansion, classification and ranking-based methods. The query expansion method is an unsupervised baseline method, and the other two are supervised methods.

3.1 Query Expansion

Query expansion is an unsupervised baseline approach. For each entity, we construct a basic phrase query with its name and surface forms (see subsection 2.2). Listing 1 (Line. 1-8) shows the basic phrase query construction using java API to *ElasticSearch*. Although the basic query can hit the documents mentioning the entity names from the index, it neither can disambiguate ambiguous entities with a same name, such as *basic_element_(company)* and *basic_element_(music_group)*, nor can differentiate the relevance levels of the hit documents.

The most pervasive and effective approach to resolve name entity disambiguation is leveraging contextual information [3]. In this work, we expand the basic phrase query with contextual related entities extracted from three sources: target entities' Wikipedia or profile pages, annotation documents and existing citations in Wikipedia. For Wikipedia entities, we use JWPL APIs [10] to extract the anchor texts of inlinks in their Wikipedia pages as related entities. For Twitter entities, Stanford Named Entity Recognizer [4] is employed to recognize entities from their profile pages. Besides, the relevant (vital or useful) documents in the ground truth data is of great help to differentiate documents with different relevance levels. The related entities appear in a vital (useful) document can help us find more vital (useful) documents. So we also extracted related entities for target entities from annotation data. It is worth noting that we only extract related entities from documents annotated as vital or useful. In addition, existing citations in Wikipedia entities' entry pages also contribute some related entities. For Twitter entities with few citations in their profile pages, we use the entities' displaying name to query in Google and crawl the top 5 hit documents as their pseudo citations.

After extracting related entities, we incorporate them into basic query and then search against the built index. The hit documents are treated as relevant documents and the rank-

[1] http://www.elasticsearch.org/

[2] http://en.wikipedia.org/wiki/Wikipedia:Redirect

[3] http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/query-dsl-match-query.html

```
BoolQueryBuilder basicQuery = QueryBuilders
                .boolQuery()
                .should(matchPhraseQuery("clean_visible", entity_name));
//surfaceform_set is redirect names set
for(String surfaceform: surfaceform_set)
        basicQuery.should(matchPhraseQuery("clean_visible", surfaceform));
//make sure the hit document match a surface form at least
basicQuery.minimumNumberShouldMatch(1);
//rel_set is related entity set of target entity
for(String rel_entity: rel_set)
        basicQuery.should(matchPhraseQuery("clean_visible", rel_entity))
```

Listing 1: Query Expansion in ElasticSearch

ing scores returned by *ElasticSearch* are scaled to (0,1000] as the final confidence scores.

We submitted 2 query expansion runs: ECQ (EntityCentricQuery) and ECQUpdate. The difference between them is that we incorporate related entities extracted from pseudo citations into the query in ECQUpdate.

3.2 Classification

CCR could be formulated as a binary classification task to differentiate relevant/irrelevant documents or vital/useful documents.

We submitted two classification runs: RFClassStrict and RFClassLoose. The former classifies the candidate documents into vital or useful, while the latter classifies the candidate documents into relevant (vital + useful) or irrelevant (neutral + garbage). We employ Random Forest classifier implementation in Weka toolkit [7] with default parameter settings.

In KBA CCR track 2012, most of the teams train a unique classifier for each target entity to exploit training data adequately. However, training data of KBA 2013 is not so sufficient. For some entities, there is no vital instance in training data. Therefore, it's unfeasible to train a unique classifier for each entity. Instead, we train a general classifier for the whole entity set with all the training instances. The features used are introduced in Section 4.

3.3 Learning to Rank

CCR could also be deemed as a ranking problem because of the ordering of the relevance levels, i.e., vital $> useful >$ $neutral > garbage$. As demonstrated in [1], ranking-based approaches have more potential than classification approaches on all evaluation measures. Therefore, we concentrated more on ranking-based approaches.

We have submitted 6 ranking-based runs. All the random forest ranking runs are implemented with RankLib[4]. The features used in ranking-based methods are mostly consistent with those in classification methods.

RFUniModel. Train a general Random Forest (RF) ranking model for all the entities with all the features except temporal features.

RFMultiModel. Train a general RF ranking model for all entities with all the features except temporal features. If there exist enough training instances for an entity (more than a pre-defined threshold), we also train a specific ranking model for it. Therefore, for the entities with few training data, the general model is selected to make predictions. While for entities with enough training data, two prediction results by the general model and the specific model are combined as the final result. RFMultiModel_1 is a parameter-tuned version of RFMultiModel.

RFDiffModel. Train two separate general RF ranking models for Wikipedia and Twitter entities respectively with all the features except temporal features, i.e. a Wikipedia ranking model and a Twitter ranking model.

RFBurst. Train a general RF model for all the entities, including burst features. We also submitted a run, named as RFBurst_1, which incorporates the annotation data of KBA 2012 into our training data through treating central as vital and relevant as useful.

4. FEATURES

In this section, we introduce the features used in our supervised approaches. [2] has summarized 4 types of useful features for CCR, including document features, entity features, document-entity features and temporal features. We adopt and enrich these features. Furthermore, we explore the citation features to improve the performance further. All the features are listed in Table 2.

Document Features. For each document, we use some features to represent its basic characteristics, such as its length, publishing date and source.

Entity Features. There is only one entity feature, i.e. the number of related entities of the target entity. For each Wikipedia entity, its entry page is useful in profiling the target entity and filter relevant documents from stream corpus. Similarly, each entity from Twitter owns a profile page. All

[4] http://sourceforge.net/p/lemur/wiki/RankLib/

entities found in the entry or profile page are considered as related entities of a target entity, as introduced in subsection 3.1.

Document-Entity Features. All the document-entity features are listed in the third block of Table 2. Except the last four similarity features, all the other features are normalized by the document's length. Given a document-entity pair, if the entity owns an entry page in Wikipedia, we calculate cosine and jaccard similarities between its Wikipedia article and the document. If the entity is from Twitter, we calculate the two similarities between its profile page and the document instead.

Temporal Features. Because CCR is a time-dependent task, some kinds of temporal features have been investigated. [8] has tried statistics gathered on a sliding window over the past week as temporal features, such as the number of the entity is mentioned in previous documents. In our approaches, we zoomed the sliding window into past 10 hours instead. Besides, [2] presents that Wikipedia page view statistics is a useful signal to capture if something are happening around the target entity at a given time point. Based on our observation, when an entity's page views present a sudden ascending, which is named as burst, the number of vital and useful documents in stream corpus show a similar trend. one reason of this phenomenon may be that most of the vital edits of entity's page would trigger lots of views from the web. The magnitudes of Wikipedia page views of different entities varies depending on their popularity. To normalize the gap between different entities, we define a *burst_value* for each document-entity pair as follows.

$$burst\_value = \frac{N * wpv(d_n)}{\sum_{i=1}^{N} wpv(d_i)} \qquad (1)$$

N is the total days the stream corpus covers. d_n means the document is published on the n_{th} day of the stream corpus. $wpv(d_i)$ is the views of the target entity's Wikipedia page during i_{th} day of the stream corpus.

Citation Features. For Wikipedia entities, there usually exist some citations in their entry pages. In our opinion, these existing citations are extremely valuable in identifying relevant documents. For each document, we calculate similarities (cosine and jaccard) between it and each cited article if the cited date is prior to the document's publishing date. For Twitter entities, we create pseudo citations as described in subsection 3.1 for each target entity. Not all entities have the same number of citations, but we need to set a fixed number of features for different entities to train a general model. Therefore, we use 6 measures to represent all the similarity features: max, mean, min, top1, top2, and top3.

5. RESULTS AND DISCUSSIONS

As reported in [5], ranking-based approach RFBurst_1 is the overall best run on *vital+useful*, and classification-based method RFClassStrict is the overall best run on *vital only.* All the results of our runs are listed in Table 3. We not only demonstrate the overall measures for all entities, the most primary measure of the performance, but also calculate these measures for Wikipedia and Twitter entities separately to evaluate these runs in a fine-grained level. Please note that the cutoffs to reach maximum of F (SU) for overall entity set and for separate entity set may be different, so the value of an overall measure is not always between the values of two separate measures.

Our two baselines (ECQ and ECQUpdate) outperform the overall mean and median of all submissions. It is not so hard to filter relevant documents from stream corpus when we expand basic phrase query with related entities. All the measures of ECQUpdate for Twitter entities are better than those of ECQ, which illustrates that pseudo citations for Twitter entities do work, although the overall performance is not improved explicitly. This may result from the few amounts of twitter entities in the whole entity set.

Almost all the classification and ranking-based approaches perform better than the two baselines in both tasks. The performance of RFDiffModel is better than that of RFUniModel, revealing that Wikipedia and Twitter entities vary from each other in the CCR task. We should tackle them respectively to improve the performance further. we find that RFMultiModel do not improve the performance very much. This may be caused by the uncertainty of "enough" training data. We manually set a threshold, while different entities require different sizes of training data to train robust models. We could utilize data-dependent mixture techniques to select a more reasonable threshold for each entity in future. All the ranking-based approaches perform very similarly if the temporal features are not included in the feature set. Temporal features (burst_value) could improve the ranking results as we speculate in Section 4.

To differentiate vital from useful, classification methods perform better than ranking methods. This reverses the conclusions on KBA 2012 data in [1], in which the authors proves that ranking-based methods are better than classification methods. We do not prepare specialized features for vital/useful classification, which shares the same feature set with relevant/irrelevant classification.

[5] has pointed out that all submissions perform approximately on $max(SU)$ and none of them can achieve a $max(SU)$ over 0.333, which is corresponding to a run with no output. This finding suggests that separating vital and useful documents is the hardest part in the CCR task. Future work needs to be done to investigate better algorithms to recognize vital evidences in stream.

6. ACKNOWLEDGMENTS

The authors would express their great appreciations to the members of Knowledge Mining Group at MSRA, especially Jing Liu(Harbin Institute of Technology), Jyunyu Jiang (National Taiwan University) and Chungyi Li (National Taiwan University) for their valuable suggestions and great help. This work is funded by MSRA Visiting Young Faculty Program, the National Program on Key Basic Research Project (973 Program, Grant No. 2013CB329605), National Science Foundation of China (NSFC, Grant Nos.60873237

Table 2: Features

Feature	Description
Document Features	
$log(length)$	log of document length
Source	source of the document, i.e. news, social, linking, etc.
Weekday	post date of the document
Entity Features	
$N(E_{rel})$	# of related entities of the target entity E in Wikipedia/profile page
Document-Entity Features	
$N(D, E)$	# of occurrences of the target entity E in document D
$N(D, E_p)$	# of occurrences of the partial name of target entity E in document D
$N(D, E_{rel})$	# of occurrence of the related entities in document D
$FPOS(D, E)$	position of first occurrence of entity E in document D
$FPOS_n(D, E)$	$FPOS(D, E)$ normalized by document length
$FPOS(D, E_p)$	position of first occurrence of partial name of entity E in document D
$FPOS_n(D, E_p)$	$FPOS(D, E_p)$normalized by document length
$LPOS(D, E)$	position of last occurrence of entity E in document D
$LPOS_n(D, E)$	$LPOS(D, E)$ normalized by document length
$LPOS(D, E_p)$	position of last occurrence of partial name of entity E in document D
$LPOS_n(D, E_p)$	$LPOS(D, E_p)$ normalized by document length
$Spread(D, E)$	$LPOS(D, E) - FPOS(D, E)$
$Spread_n(D, E)$	$Spread(D, E)$ normalized by document length
$Spread(D, E_p)$	$LPOS(D, E[p]) - FPOS(D, E_p)$
$Spread_n(D, E_p)$	$Spread(D, E_p)$ normalized by document length
$Sim_{cos}(D, W_E)$	cosine similarity between document and entity's Wikipedia/Profile page
$Sim_{jac}(D, W_E)$	jaccard similarity between document and entity's Wikipedia/Profile page
Temporal Feature	
$PreMention(E, h)$	# of the entity E is mentioned in previous h hours before the timestamp of document
Burst_Value	see Equation 1, only used in `RFBurst` and `RFBurst_1`
Citation Features	
$Sim_{cos}(D, C_i)$	cosine similarity between document and existing citation C_i
$Sim_{jac}(D, C_i)$	jaccard similarity between document and existing citation C_i

Table 3: Results of official runs. All the measures are reported by official scorer with cutoff-step-size=10. Median and Mean are the median and mean of results aggregated from all the submissions in this year's KBA CCR track

Run	Vital Only						Vital + Useful					
	$max(F(avg(P), avg(R)))$			$max(SU)$			$max(F(avg(P), avg(R)))$			$max(SU)$		
	Overall	Wiki	Twitter	Overall	Wiki	Twitter	Overall	Wiki	Twitter	Overall	Wiki	Twitter
ECQ	.281	.288	.257	.170	.178	.174	.645	.658	.567	.544	.557	.466
ECQUpdate	.281	.288	.274	.173	.178	.194	.645	.658	.600	.544	.557	.536
RFClassStrict	**.303**	**.300**	**.330**	.249	.247	**.292**	.614	.614	.604	.535	.534	.537
RFClassLoose	.300	.298	.330	.204	.203	.230	.649	.661	.604	.557	.562	.553
RFUniModel	.285	.293	.312	.230	.233	.224	.644	.657	.646	.546	.557	.574
RFDiffModel	.291	.293	.290	.216	.217	.217	.655	.659	.653	.562	.565	**.604**
RFMultiModel	.286	.293	.311	.234	.239	.226	.644	.657	.649	.547	.557	.578
RfMultiModel_1	.285	.293	.240	.234	.239	.217	.644	.657	.567	.544	.557	.477
RFBurst	.294	.294	.306	.228	.227	.255	.653	.657	.657	.563	.562	.586
RFBurst_1	.296	.296	.312	.243	.247	.273	**.659**	**.660**	**.665**	**.570**	**.566**	.599
Max	.303	.300	.330	.277	.280	.292	.659	.660	.665	.570	.566	.604
Median	.174	.179	.164	.255	.259	.233	.406	.382	.333	.423	.433	.389
mean	.166	.172	.136	.137	.240	.224	.376	.433	.360	.425	.438	.364

and 61003168), and Beijing Higher Education Young Elite Teacher Project (Grant No. YETP1198).

7. REFERENCES

[1] K. Balog and H. Ramampiaro. Cumulative citation recommendation: classification vs. ranking. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '13, pages 941–944, New York, NY, USA, 2013. ACM.

[2] K. Balog, H. Ramampiaro, N. Takhirov, and K. Nørvåg. Multi-step classification approaches to cumulative citation recommendation. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, OAIR '13, pages 121–128, Paris, France, France, 2013.

[3] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[4] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[5] J. R. Frank, S. J. Bauer, M. Kleiman-Weine, D. A. Roberts, N. Tripuraneni, C. Zhang, C. Re, E. M. Voorhees, and I. Soboroff. Evaluating stream filtering for entity profile updates for trec 2013. In *TEXT RETRIEVAL CONFERENCE*, 2013.

[6] J. R. Frank, M. Kleiman-Weiner, D. A. Roberts, F. Niu, C. Zhang, C. Re, and I. Soboroff. Building an Entity-Centric Stream Filtering Test Collection for TREC 2012. In *Proceedings of the Text REtrieval Conference (TREC)*, 2012.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[8] V. B. Ludovic Bonnefoy and P. Bellot. Lsis/lia at trec 2012 knowledge base acceleration. In *Proceedings of the Text REtrieval Conference (TREC)*, 2013.

[9] S. Robertson and I. Soboroff. The trec 2002 filtering track report. In *TEXT RETRIEVAL CONFERENCE*, 2002.

[10] T. Zesch, C. Müller, and I. Gurevych. Extracting lexical semantic knowledge from wikipedia and wiktionary. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Marrakech, Morocco, May 2008. electronic proceedings.