RMIT at the NTCIR-12 MobileClick-2: iUnit Ranking and Summarization Subtasks

Kevin Ong, Ruey-Cheng Chen, and Falk Scholer
RMIT University
Melbourne, Australia
{kevin.ong, ruey-cheng.chen, falk.scholer}@rmit.edu.au

ABSTRACT

In NTCIR-12 MobileClick-2 challenge, the RMIT Information Storage and Retrieval (ISAR) group participated in both the English iUnit Ranking and Summarization tasks. This paper describes how we applied a learning-to-rank approach to the problem of iUnit Ranking and how the outcome of ranking was adapted to produce the summaries.

Team Name

RISAR

Subtasks

iUnit Ranking (English); iUnit Summarization (English)

Keywords

Learning-to-Rank, Semantic Analysis

1. OVERVIEW

The RMIT University ISAR group participated in the NTCIR-12 MobileClick-2 iUnit Ranking and Summarization subtasks. In this challenge, we took a feature-based approach and set out to explore novel features to enhance iUnit Ranking. This work is based on our previous efforts on sentence selection for non-factoid question answering [1, 7], in which a Learning-to-Rank (LTR) framework from the domain of query-biased summarization is extended to the problem of answer sentence retrieval. Our efforts were to maximise the scores from ranking subtask first and use them to improve the summarization subtask. In the following sections, we describe the ranking and the summarization experiments conducted for the MobileClick-2 challenge.

2. RANKING EXPERIMENT

We approached the problem of iUnit Ranking in a LTR framework. This approach has previously been successfully applied to similar problems such as sentence/answer selection in query-biased summarization and non-factoid question answering. iUnits essentially cover a full spectrum of sub-sentence text units of different lengths and different types of information contents. They can be short phrases representing simple factoids about some celebrity, or long, complex clauses that address users'natural language questions. Whether the LTR framework alone can deal with this level of heterogeneity is of technical interest.

2.1 Features

Five classes of features were explored in this experiment, with the full list of features given in Table 1:

- odds ratio derived from the language model baseline;
- features for query-biased summarization [5];
- "semantic features" for answer finding [7];
- features that addresses entity queries;
- features that exploits document contexts of the iUnit.

Query-Biased Summarization and Semantic Features.

The first class has one feature, which is the score from the baseline language model. While the second class of features covers the lexical/synonymy features derived from Metzler and Kanungo [5], which was originally developed for the sentence selection task in query-biased summarization. We implemented 5 such features: `ExactMatch`, `TermOverlap`, `SynonymOverlap`, `LanguageModel` (not to be confused with the baseline), and `iUnitLength`. The one indicating relative position of the iUnit was left out. Another feature similar to this one, called `AverageSentencePos`, is actually covered in the fifth class about document contexts.

The third class implements an extension to Metzler and Kanungo. This extension was described in Chen et al. [1] and Yang et al. [7], developed to cover the semantic relatedness between the query and the sentence. This class covers three features:

ESA Explicit Semantic Analysis (ESA) [3] can semantically annotate any given text with a set of Wikipedia concepts and assign appropriate weights to such annotation. Using this method, the semantic relatedness between a query and an iUnit can be computed as the cosine similarity between ESA vector representations.

Word2Vec We used the pretrained Word2Vec word embedding model by Mikolov et al. [6] to compute word vectors in the query and in the iUnit.[1] The relatedness between the query and the iUnit is measured by the average pariwise cosine similarity between any query-word and iUnit-word vectors.

Tagme We used an entity linking system Tagme [2] to annotate queries and iUnits. Relatedness is computed by the Jaccard coefficient between the Wikipedia pages linked to the query q and those linked to the iUnit.

[1] `https://code.google.com/p/word2vec/`

Base: *Baseline*	
`OddsRatio`	Likelihood ratio score from the LM baseline
QSum: *Features for query-biased summarization* [5]	
`ExactMatch`	Binary feature indicating whether the query is a substring of the iUnit
`TermOverlap`	Fraction of query terms that are also in the iUnit
`SynonymOverlap`	Fraction of query terms that have a synonym (including the original term) in the iUnit
`LanguageModel`	Log likelihood of the query being generated from the iUnit
`iUnitLength`	Number of terms in the iUnit
Sem: *Features for non-factoid question answering* [7]	
`ESA`	Cosine similarity between the ESA vectors representing the query and the iUnit
`Word2Vec`	Average pairwise cosine similarity between any query-word vector and any iUnit-word vector
`Tagme`	Degree of the set overlap between the entities associated with the query and the iUnit
QT: *Features regarding natural language questions and entity queries*	
`Leading5W1H`	Binary feature indicating whether the query begins with the `who`, `what`, `where`, `when`, `why`, or `how`
`TopRankedWikiPage`	Binary feature indicating whether a wikipedia page is among the top 3 retrieved webpages
`WikipageRR`	Reciprocal rank of the top-ranked wikipage
`WikipagePassage`	Fraction of query terms covered by the best-matching wiki passage divided by the passage rank
Ctx: *Features regarding the context of the iUnit*	
`CollectionFrequency`	Number of occurrences of the iUnit in the retrieved webpages
`AverageSentencePos`	Average position of the sentences (in the retrieved pages) that contain the iUnit
`AverageDocumentRR`	Average reciprocal rank of the webpages that contain the iUnit

Table 1: List of features

Question and Entity Query Features.

The fourth class of features was developed to address query type. We noticed that some of the queries are natural language questions, and some others are "entity queries." The entity queries are queries that have one or more Wikipedia pages listed in the respective top-3 retrieved results. This suggest that the need behind these queries can be addressed with access to a knowledge base and are likely to be informational rather than navigational. We notice that more than half of the topics in the MobileClick data fall within this category. In the training 65 out of 100 topics and in the test set 58 out of 100 topics are actually entity queries.

We believe that our ranking model would fit better to the data by separating the query types, and this can be done simply by incorporating the following query type features:

Leading5W1H The feature is set to 1 if the iUnit starts with interrogative words `who`, `what`, `where`, `when`, `why`, or `how` (commonly referred to as five Ws and one H), and to 0 otherwise.

TopRankedWikiPage The feature is set to 1 if a Wikipedia page is found in the top-3 retrieved pages for the query (i.e., having `wikipedia.org` in the URL), and to 0 otherwise.

WikipageRR The reciprocal rank of the first Wikipedia page found in the top-3 results. If, for instance, a Wikipedia page is retrieved at rank 3 for this query, the feature is set to 1/3. If the top result does not have this page, the feature is set to 0.

WikipagePassage This feature aims at assessing the relevance between the query and the first top-ranked Wikipedia page. It is computed by first locating the "best-matching" 3-sentence passage in the Wikipedia page (with the best query term coverage), and then following the equation:

$$\frac{\text{fraction of query terms covered}}{\text{position of the passage in the Wiki page}}. \quad (1)$$

Note that the position of the passasge is based on the position of its first constituent sentence. As an example, a passage that spans sentences 10 to 12 and covers half of the query terms would give a value 0.5 / 10 = 0.05 for this feature. This equation sort of has the flavor of the fraction of term coverage discounted by the passage rank. The reasoning is that the best-matching passage may be less informative as it becomes more far down in the wikipage.

Document Context Features.

The fifth class of features are based on the document contexts of the iUnits. An iUnit may have multiple occurrences in the HTML pages, and the documents where they appear in are called the document context. Exploiting this resource can reveal properties such as the repetition of an iUnit, whether the iUnit are in the early part of documents, whether the documents that contains the iUnit are high in the ranking, signals that indicate iUnit importance.

We therefore come up with the following three features:

CollectionFrequency The number of occurrences of the iUnits in the retrieved webpages.

AverageSentencePos The average position of the sentences (in the respective document) in which the iUnit appears. For instance, if an iUnit appears in one document at sentences 1 and 2 and in another at sentence 9, the average would be 4.

AverageDocumentRR The average reciprocal rank of the documents in which the iUnit appears. For instance,

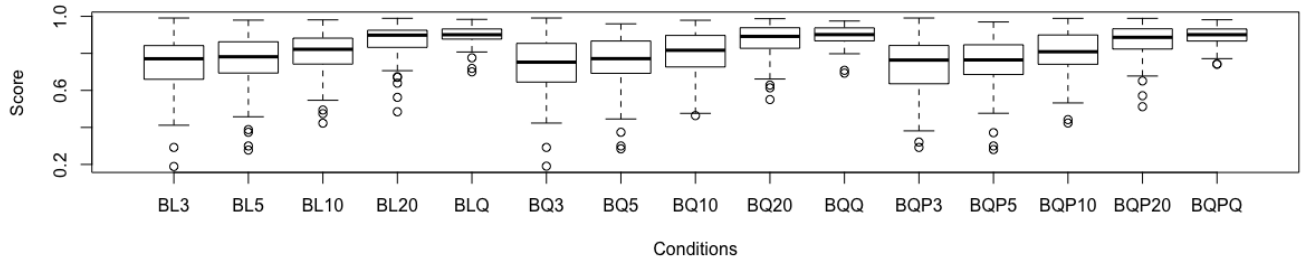
Method	Algorithm	nDCG@3	nDCG@5	nDCG@10	nDCG@20	Q
Base (**BL**)		0.7438	0.7643	0.7961	0.8325	0.8772
Base+QSum (**BQ**)	Coordinate Ascent	0.7669*	**0.7821***	**0.8137***	**0.8504****	**0.8882****
Base+QSum+Sem	LambdaMART	**0.7686***	0.7759	0.8085*	0.8422*	0.8848**
Base+QSum+Sem+QT	LambdaMART	0.7674*	0.7788	0.8070*	0.8453**	0.8875**
Base+QSum+Sem+QT+Ctx (**BQP**)	LambdaMART	0.7668*	0.7801*	0.8067	0.8435*	0.8819

Table 2: Performance results on the training set for English iUnit Ranking. Significant improvements are indicated by * and ** (Two-tailed t-test; $p < 0.05$ and $p < 0.01$ respectively).

Method	Algorithm	nDCG@3	nDCG@5	nDCG@10	nDCG@20	Q
Base (**BL**)		**0.7460**	**0.7596**	**0.8033**	0.8689	**0.8975**
Base+QSum (**BQ**)	Coordinate Ascent	0.7354	0.7557	0.8015	**0.8690**	0.8972
Base+QSum+Sem+QT+Ctx (**BQP**)	LambdaMART	0.7352	0.7532	0.8002	0.8666	0.8962

Table 3: Performance results on the test set for English iUnit Ranking.

Figure 1: Shows the distribution of score across all the tasks. BL/BQ/BQP represents $Baseline$**,** $Baseline + QSum$ **and** $Baseline + QSum + Sem + QT + Ctx$ **respectively.**



if the iUnit appears in documents at rank 3 and 6, the average reciprocal rank would be $1/3 + 1/6 = 1/2$.

The occurrences of iUnits are identified by running case-sensitive string matching. To further pinpoint the iUnits at the sentence level, we used the sentence delimiter in Apache OpenNLP to split sentences.

2.2 Evaluation

We followed the setting in Yang et al. [7] to set up the features of the second and the third classes. We set $\mu = 10$ for the feature `LanguageModel`, and used WordNet synsets to implement `SynonymOverlap`. We experimented with two ranking algorithms, Coordinate Ascent and LambdaMART and using the one with the best performance for each feature set. The former was implemented using RankLib[2] and the later using jforests[3].

Model training was done entirely the training data in two stages: The hyperparameters of the ranking model were first optimized by using a cross-validated grid search, and then a new model was fitted to the full training set using the found hyperparameters. The query topics were shuffled in advance to ensure that each fold covers diverse types of topics. Feature values were normalized to the range [0, 1] using max-min normalization. For LambdaMART, we ran a thorough search tweaking the following parameters: number of trees, number of leaves in a tree, minimum instance percentage per leaf, learning rate, sub sampling rate, and feature sampling rate. For Coordinate Ascent, we tweaked the number of random starts and the number of search iterations.

Both ranking models were optimized based on nDCG. The training weights were converted to graded relevance by simply scaling the original values. We experimented with several mappings and eventually settled on the following:

Original weight	0–2	3–4	5–6	7–8	9–10
Graded relevance	0	1	2	3	4

The training result is given in Table 2. In this experiment, we tried building up a full ranking model by incorporating one feature group at a time. The first combination we tested is the baseline score (Base) coupled with the query-biased summarization features (QSum). This combination is later extended by incorporating semantic features (Sem) to address the query-iUnit term mismatch problem, question and entity features (QT) to focus on separate query types, document context features (Ctx) to incorporate the signals from the retrieved results. Each feature combination was tested on both Coordinate Ascent and LambdaMART, but only the one delivered the best performance is reported.

From Table 2, it is shown that adding more features does not lead to better performance. Training performance has not seen improvements by extending the Base+QSum combination. The results shows that using all the results (Q) for $Base + QSum$ method gives the best performance (BQQ).

Test Results.

The test result is given in Table 3. We observed that

[2] `http://www.lemurproject.org/ranklib.php`
[3] `https://github.com/yasserg/jforests`

Figure 2: Distribution of score for summarization subtask between Baseline and BQP. Baseline is observed to perform better than BQP features

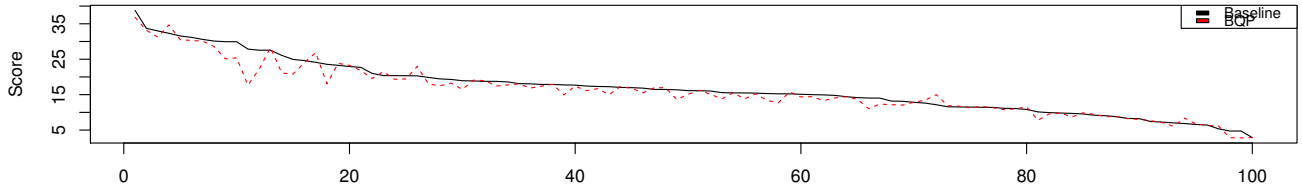
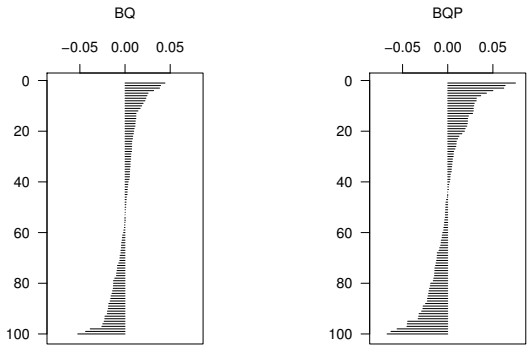


Figure 3: Difference of scores (Q) across all the query topics against Baseline



we managed to beat Baseline scores for all values of nDCG except for nDCG@3. Figure 1 gives a breakdown of the performance at various levels of nDCG scores (3, 5, 10, 20 and all results (Q)). The figure shows values for baseline (BL), baseline+QSum (BQ) and Baseline+QSum+Sem+QT+Ctx (BQP). Within each method, the scores improves as more results are added for evaluation. The observed variance for Q is smaller for all methods.

Figure 3 shows the differences of the distribution of score across all query topics against Baseline for BQ and BQP. The variance of the score for BQ against baseline is relatively smaller compared to BPQ against Baseline.

While we found that our results were better for the training set of data, we did not get the same performance for test data. As we were not able to do further testing, it may be possible that our LTR algorithm may have been tuned to the training data, resulting in a slightly less than ideal performance on the test dataset.

3. SUMMARIZATION EXPERIMENT

In the iUnit Summarization task, we modified the two-tier Baseline algorithm to work with custom ranking input. To experiment with the new features, we fed the algorithm with the ranking run BQP, which made use of all features, rather than the run BQ that yielded a better test result for the iUnit Ranking subtask. We did not experiment with BQ feature set because we want to see if $Sem{+}QT{+}Ctx$ features would improve performance over Baseline. The results are summarized in Table 4 and shows that the Baseline performs better than our BQP features.

We originally planned to use the ranking algorithm to influence the summarization method, but we found that did not give use the results we were after. Interestingly, we found that BQP ranking scores is negatively related to BQP summarization scores (Pearson correlation, $r = -0.14$; $p > 0.05$). Although the effect is not significant, it seems to suggest that our efforts on improving ranking failed to translate into better summaries for individual query topics. A further investigation into this may be worthwhile.

Figure 2 shows the distribution of score between Baseline and BQP features for the summarization subtask. The distribution shows that in most instances, we did not manage to beat Baseline across all 100 query topics.

Method	M
Base (**BL**), 2-layer	**16.8975**
Base+QSum+Sem+QT+Ctx (**BQP**), 2-layer	16.047

Table 4: Results for English iUnit Summarization.

4. CONCLUSION

In terms of ranking subtask, we observed that Q improved performance for all methods and Base+QSum (**BQ**) with Q gave the best performance for ranking training data. However, we did not observe the same performance increase in the test dataset. This suggests that overfitting might have occurred for the training dataset. In terms of summarization subtask, we found that Base +QSum+Sem+QT+Ctx (**BQP**) did not manage to beat Baseline (**BL**).

5. REFERENCES

[1] R.-C. Chen, D. Spina, W. B. Croft, M. Sanderson, and F. Scholer. Harnessing semantics for answer sentence retrieval. In *Proceedings of ESAIR'15*, 2015.

[2] P. Ferragina and U. Scaiella. Fast and accurate annotation of short texts with wikipedia pages. *Software, IEEE*, 29(1):70–75, 2012.

[3] E. Gabrilovich and S. Markovitch. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of IJCAI'07*, 2007.

[4] M. P. Kato, M. Ekstrand-Abueg, V. Pavlu, T. Sakai, T. Yamamoto, and M. Iwata. Overview of the NTCIR-11 MobileClick Task. In *NTCIR*, 2014.

[5] D. Metzler and T. Kanungo. Machine Learned Sentence Selection Strategies for Query-Biased Summarization. In *Proceedings of SIGIR 2008 Learning to Rank Workshop*, 2008.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[7] L. Yang, Q. Ai, D. Spina, R.-C. Chen, L. Pang, W. B. Croft, J. Guo, and F. Scholer. Beyond factoid QA: Effective methods for non-factoid answer sentence retrieval. In *Proceedings of ECIR'16*, 2016.