Quote Recommendation for Dialogs and Writings

Yeonchan Ahn*, Hanbit Lee*, Heesik Jeon**, Seungdo Ha* and Sang-goo Lee*
*School of Computer Science and Engineering, Seoul National University, Seoul, Korea
**Artificial Intelligence Team, Samsung Electronics Co., Ltd., Seoul, Korea
*{skcheon, acha21, seungtto, sglee}@europa.snu.ac.kr, **heesik.jeon@samsung.com

ABSTRACT
Citing proverbs and (famous) statements of other people can provide support, shed new perspective, and/or add humor to one's arguments in writings or dialogs. Recommending quote for dialog or writing can be done by considering the various features of the current text called context. We present five new approaches to quote recommendation: 1) methods to adjust the matching granularity for better context matching, 2) random forest based approach that utilizes word discrimination, 3) convolutional neural network based approach that captures important local semantic features, 4) recurrent neural network based approach that reflects the ordering of sentences and words in the context, and 5) rank aggregation of these algorithms for maximum performance. We adopt as baseline state-of-the-arts in citation recommendation and quote recommendation. Experiments show that our rank aggregation method outperforms the best baseline by up to 46.7%. As candidate quotes, we use famous proverbs and famous statement of other person in dialogs and writings. The quotes and their contexts were extracted from Twitter, Project Gutenberg, and Web blog corpus.

CCS Concepts
• Information systems ~ Recommender systems • Natural language processing.

Keywords
Quote recommendation; Context matching; Random forest; Convolutional Neural Network; Recurrent Neural Network; Rank aggregation

1. INTRODUCTION
Citing proverbs and (famous) statements of other people is an important part in conversation and writing. Such quotes or quotations can provide support, shed new perspective, and/or add humor to one's arguments. However, it is not easy for a person to find from a large number of quotes an appropriate one for a given context since the words in quote are usually metaphorical.

Quote recommendation in writing has been introduced in Tan, et al. [7]. Quote recommendation is a task of recommending a ranked list of quotes which are relevant to the current body of text which we call *context*. We separate *context* into *pre-context* and *post-context*, which refer to texts that appear before and after a quote within certain fixed length respectively. For dialogs, unlike for writings, we only use pre-contexts because post-contexts are usually unavailable for on-the-fly recommendation of quotes during a conversation in real world applications. We define *query* as a context for which the user desires a list of recommended quotes. Figure 1 shows an example of quote usage in our Twitter dataset. In this example, the block of text that appears before the quote '*Strike while the iron is hot*' is the pre-context.

CBRecSys 2016, September 16, 2016, Boston, MA, USA.

Figure 1 An example of quote usage in Twitter thread

On investigating our collected datasets, we found that various features of context, such as keywords, topic, n-grams, latent semantics, etc., can be exploited in the recommendation. For example, word matching-based algorithm such as ranking with cosine similarity between query and context of quote was able to find the correct quote in Figure 1, since many contexts of the same quote in training dataset mention the keywords such as *casino* and *luck* but the others do not. Also, some of the quotes are closely related to specific situations, topic or semantics behind query not to only keywords.

In this paper, we present five new approaches for quote recommendation based on observations in our datasets: 1) methods to adjust matching granularity for better context matching, 2) Random Forest (RF) based approach that utilizes word discrimination, 3) convolutional neural network (CNN) based approach that captures important local semantic features, 4) recurrent neural network (RNN) based approach that reflects the ordering of sentences and words in the context, and 5) rank aggregation of these algorithms for maximum performance. As baseline, we adopt previous works on citation recommendation [1, 3] and quote recommendation [7]. Experiments show that the proposed approaches significantly outperform baseline methods in real world datasets.

2. RELATED WORKS
Quote recommendation can be viewed as task of searching or recommending short texts which are appropriate to given current writing or dialog context. Most related works are citation recommendation for academic articles [1, 3], which recommends relevant reference articles for academic writing. For citation recommendation, rich information on paper such as title, abstract, full text and venue can be exploited. In contrast, in quote recommendation such rich information is not available. This makes quote recommendation more challenging. Tan, et al. [7] present a method for recommending quote for the first time. They apply learning-to-rank approach with several features which is quote-context, context-context (or context-query), and quote feature. In their experiments, they show that the algorithm heavily depends on context-context feature. However, we argue that enough exploration on the context-context features is not

conducted. For this, we focus on how to mine the semantics on contexts of quote for recommending quote.

3. APPROCHES

In this section, we describe four approaches and our rank aggregation method which combines the four approaches for quote recommendation.

3.1 Matching Granularity Adjustment

In this section we discuss methods to deal with the contexts of quotes when measuring relevance between query and a set of contexts of quotes, which we call matching granularity adjustment. As usage of words or words themselves in the quote are different from that in context, the state-of-the-arts in quote/citation recommendation [1, 3, 7] measures the relevance between query and contexts of a quote. More specifically, all of them attempt to examine individual context of a quote to the query. A drawback of this approach is that it suffers from sparsity problem that words in query do not match the individual context of the correct quote. In order to alleviate this sparsity problem, we propose methods to adjust the matching unit of contexts to the given query. We believe that more semantics can be exploited if the contexts of a quote are treated collectively.

Firstly, we propose a method called context clustering, which group the context by context cluster which represent (latent) topic. In the collected dataset, we observed that there exist a number of quotes that can be used in different topic. For example, the quote '*All work and no play makes jack a dull boy*' can be used in very different situations such as '*overworking in workplace*' or '*educating children*'. Thus when dealing with query about specific topic, we need to consider the contexts related to it among different topics of quote. In the context clustering, we first clusters contexts of each quote. And we exploit the context clusters to measure the relevance of a quote. For context clustering, we adopt affinity propagation clustering algorithm, which is known to perform better than others in short text clustering [6]. Based on context clustering, we propose a scoring function given query q:

$$sim_{\text{cmax}}(q,t) = \max(sim(q,\ CC_t^{(j)}))$$

where $CC_t^{(j)}$ is concatenated text in jth context cluster of quote t and sim is cosine similarity with their TF-IDF vector representation.

In order to solve the sparsity problem, we present another method called context lumping to adjust the matching granularity. In context lumping, we simply concatenate all the context of each quote and make it a matching unit to the query. Then the lumped context of quote is compared to query with cosine similarity with TF-IDF vector representation. In both of context clustering and lumping, quotes are sorted by the proposed similarities in descending order respectively.

3.2 Random Forest

In the collected dataset, we observe that some simple rules such as checking whether the given context contains certain words are reliable cursors to its correct label. For example, in Twitter dataset, given that a context contains the keywords *invite*, *join*, *come over* or any of the morphemes, there is 40.2% probability that the context is labeled with the proverb '*the more the merrier*'. From this observation, we explore the possibility of adopting tree based classification algorithm into the quote recommendation task.

Among various decision tree algorithms, RF [5] is an ensemble learning method that had notable success in various fields due to its resilience to over-fitting and tendency to exhibit low variance and bias. RF constructs n_{tree} decision trees by training each tree with samples of random subset of features. The method is able to populate each decision tree with the most discriminating quotes at each state and aggregate the results by voting. In the case of our dataset, we view contexts as 'documents' and use bag-of-words TF-IDF as features for each context. Then, we train the Random Forest classifier using the vectors of TF-IDFs and their correct labels i.e. quote. To the best of our knowledge, this is the first time RF classification has been used for quote recommendation.

3.3 Convolutional Neural Network

Word matching-based methods such as context-aware relevance model [1] and citation translation model [3] have difficulty in exploiting n-gram features because of sparsity problem, so they only use unigram-based features. But n-gram features are important because there are many phrases which are meaningful only when the terms in phrase stay together. For example, a phrasal verb *give up* loses the meaning when it is tokenized into *give* and *up*. Unlike matching-based methods, CNN based approach can exploit important n-gram features in the context by learning the parameters of fixed size filters for each n-grams. Generally, CNN is composed of several pairs of convolution layer and max-pooling layer which capture the local patterns from the training example and down-sample extracted features in order to prevent overfitting. When CNN is applied to natural language sentences, it captures the significant local semantics, i.e., n-gram.

We adopted a single-layer CNN, mainly inspired by [4] which reports that simple CNN model shows similar performance to complex one with several convolutions-pooling layers in order to capture distinguished n-gram features in contexts of quotes. Our CNN model takes a context in the form of a list of word embedding vectors of the words in the context. Then the input matrix, a list of context vectors, are fed to the layer which is composed of single convolution layer and max-pooling layer. After that the output vector is fed to fully connected softmax layer in order to compute the probability of candidate quotes and rank the quotes. We use filter size of 3 and 500 hidden nodes in the hidden layer. We also exploit dropout method to prevent overfitting.

3.4 Recurrent Neural Network

We use RNN to tackle our quote recommendation problem in perspective of language modeling, which means that we treat each quote as a special token or word and compute the probability of it given context. While none of above approaches uses order information of words in the context, RNN based approach can model such sequence of words recursively. We use long short-term memory unit (LSTM) [2] which is a recurrent neural network consists of three gates (forget, input, output) those control the networks to learn long-term dependencies without loss of information. The input vector of each time step passes through the three gates and updates latent vectors which LSTM is retaining. In our model we recurrently feed LSTM with a sequence of words in the context in the form of list of word embedding vectors. We use pre-trained word embedding for mapping each word to word vector. The output vector of LSTM layer is passed to fully connected layer and softmax layer in order to compute the probability of target quotes to be recommended. We also use 500 dimension hidden vector in LSTM and also use dropout method.

3.5 Rank Aggregation

We observed that previously proposed algorithms show different recommendation results according to queries (we will

discuss this in the experiment section). This suggests that instead of relying on the single best ranking algorithm, it is better to aggregate rank values of all of the single algorithms to produce accurate and robust ranking, called rank aggregation (RA).

We propose two methods which can aggregate the individual ranking results of previously proposed algorithms. Traditional rank aggregation method Borda [8] assigns a score to candidate quote inversely proportional to its position in a ranked list of individual algorithm, and the scores of each quotes are added up to the final score. We observed that Borda cannot handle the case where one or two inaccurate rank of individual algorithms lowers accuracy of final aggregated rank. In order to cope with this issue, we propose a rank aggregation method called Rank Multiplication (RM) to multiply the ranks of each quotes submitted by individual algorithm. By using this method, we can get the effect that maintaining case that all of the individual ranker rank consistently high, it can give less weight to result of inaccurate ranking algorithm. Thus final score by using RM can be defined as follows:

$$s_{RM}(q,t) = \frac{1}{\prod_i^{|A|} r_i(q,t)}$$

where $r_i(q,t)$ is position in ranked list of ith individual ranking algorithm given query q and candidate quote t. And A is a set of each algorithm. The quotes are ordered by this score in descending order.

We assume that high ranks of individual ranking algorithms are more dependable than lower ranks. From this assumption we propose second rank aggregation method called top-k Rank multiplication (top-k RM) that multiplies only k rank values of a quote from each of k single algorithms for a query. Thus the final score of the top-k RM is defined as follows:

$$s_{TopK\_RM}(q,t) = \frac{1}{\prod_{i \in TopK}^{|A|} r_i(q,t)}$$

where $TopK$ is a set of k algorithms that yield the k highest rank positions given query q and quote t.

4. EXPERIMENTS

4.1 Data Construction

We have collected 439,655 quotes from three sources: *Wikiquote*[1]*, Oxford Concise Dictionary of Proverbs*[2], and *Library of Quotes*[3]. For the context data, we searched blocks of texts that contain these quotes from three different sets of corpus: 2 million tweet threads from Twitter (~2015.11.15), 20GB of electronic book from the *Project Gutenberg Database*[4], and 190GB of ICWSM spinn3r 2009 blog dataset[5]. In the tweet corpus, in order to extract dialogs only, we selected threads where only two users are involved. Next, we chose the top 400 quote set from each corpus according to the number of contexts, in order to reflect the characteristics of the quotes that appeared frequently in different corpus. Finally, we generate three datasets: Twitter dataset, Gutenberg dataset, and Blog dataset.

[1] https://en.wikiquote.org/

[2] Oxford University Press, 1998

[3] http://www.libraryofquotes.com/

[4] http://www.gutenberg.org/

[5] http://icwsm.cs.umbc.edu/data/icwsm2009/

Table 1 number of contexts for each quote in datasets

Datasets	Avg	Std dev	Max	Min
Twitter	556	971	10764	15
Gutenberg	89	122	1366	14
Blog	230	543	5923	24

Table 1 shows the number of context for each quote in each datasets, which describes average, maximum, and minimum number of context for each quote and standard deviation of them. From Table 1, we see that the most frequently appeared quotes from each corpus cover large range of quotes of varying frequencies, helping us deal with the situation recommending quotes by using small number of contexts as well as large number of contexts. We divide dataset to the proportion of 8:1:1, as training set, validation set, and test set. We create test sets by hiding the quotes which the contexts are paired with.

4.2 Evaluation Metric

We consider a plausible application that recommends only a few number of quotes. In such application since the position of correct quote is not important, we use Recall@k as our evaluation metric.

Recall@*k*: Since there is only one correct or hidden quote for each query in the original test set, Recall@k is the number of cases that the gold quote is recommended in the top-k result divided by number of total test cases. We set k as five.

4.3 Baselines and Parameter Settings

We compare our approaches with three state-of-the-art approaches in quote or citation recommendation domain. Learning-to-recommend quote (LRQ) [7] is an algorithm for recommending quote for wring. Context-aware relevance model (CRM) [1], citation translation model (CTM) [3] are algorithms for recommending citation for scientific paper. Also popularity-based method (Popularity), and cosine similarity-based method (Cosine similarity) is adopted as baselines. The methods, Popularity and Cosine similarity methods are used in order to reveal the different levels of difficulties of the datasets. These methods are described in detail below.

LRQ exploits an existing learning-to-rank framework for quote recommendation with quote-based features, quote-query similarity features, and context-query similarity features.

CRM recommends quotes according to average of the squared cosine similarities between contexts of each quote and the query.

CTM recommends quotes according the probability that the query context would be translated into the quote.

Popularity ranks the quotes according to their frequency in contexts of training set.

Cosine similarity ranks the quote by examining individual context of the quote with the given query using bag-of-words representation.

We implement these methods and set the parameters to optimum as specified in the respective papers of the methods. Specifically, we truncate each half-context (pre-context or post-context) of length longer than 150 characters for LRQ, 50 words for CRM and one sentence for CTM respectively as the respective authors suggested in the papers. For our approaches, we set length of half-context to its optimal value which shows best result in

validation dataset: 1) 150 characters of pre-context and post-context with word truncation for context clustering and context lumping, 2) 50 words for RF, and 3) 30 words of pre-context for CNN and RNN. As stated in the introduction, we used pre-context and post-context as query for Gutenberg and Blog dataset and pre-context as query for Twitter dataset. Hyper parameters of single algorithms are set by using validation set. For rank aggregation, we used the proposed five algorithms (context clustering, context lumping, RF, CNN and RNN) and, top-k RM showed best results when k=3.

4.4 Results and Discussions

Results of experiments are listed in Table 2. Recall@5 and the improvement ratio of each algorithm over the best baseline in each dataset are denoted. The individual algorithms (context lumping and CNN), even without rank aggregation, outperform baselines in all of the datasets. Surprisingly, the simple method context lumping is the best performer in Gutenberg and Blog dataset, which beats LRQ up to 35%. Context clustering outperforms CRM and Cosine similarity which does not treat the context of quote collectively. These better results of context lumping and context clustering show the effectiveness of adjusting context matching granularity. One can observe that performance of the baseline Cosine similarity in Twitter dataset is worse than ones in Gutenberg and Blog dataset. This means that sparsity problem is more serious in Twitter where the tweet contains more infrequent words than others. In Twitter dataset, deep learning algorithms (CNN and RNN) outperform CTM by up to 43%. From this result, we can see that deep learning algorithms are able to mitigate such serious sparsity problem because it is not based on word matching. Results of RF show that it is competitive to CTM algorithm. In fact, in our preliminary experiments on top 100 Twitter dataset, RF outperforms CNN. However, in large dataset, generalization of the algorithm is not made as expected; an area for future investigation.

Although some of our single algorithm outperform others in specific datasets, there is no single algorithm that outperforms all the others. Also even in a dataset, there exists a portion of queries where each of single recommendation algorithms is exclusively correct. See Table 3. These justify our motivation of adopting rank aggregation, and as expected, improvement attained through rank aggregations (RM RA and top-k RM RA) are better than the best baseline algorithm on average 44.0% and 46.7% respectively.

Table 2 Results of Recall@5 of different methods.

Context source / Approaches	Twitter	Gutenberg	Blog
Context clustering	0.190 (-30%)	0.299 (- 1 %)	0.494 (0 %)
Context lumping	0.286* (+ 5%)	0.409* (+35%)	0.521* (+ 5 %)
RF	0.244 (- 11%)	0.246 (-19 %)	0.470 (- 5 %)
CNN	0.390* (+43%)	0.326* (+ 8 %)	0.506 (+ 2 %)
RNN	0.389* (+42%)	0.294 (- 3 %)	0.473 (- 4 %)
RM RA	0.424* (+55%)	0.445* (+47%)	0.640* (+30 %)
top-k RM RA (k=3)	**0.436*** (+60%)	**0.451*** (+49%)	**0.648*** (+31 %)
LRQ	0.196	0.302	0.494
CRM	0.119	0.237	0.382
CTM	0.273	0.257	0.441
Popularity	0.156	0.111	0.223
Cosine similarity	0.196	0.248	0.469

(* indicates that each of our algorithms outperform the best baseline algorithm with statistically significant increase at $p < 0.01$ in two-tailed t-tests)

Table 3 number of correct cases of single algorithms in Twitter dataset

Approaches	# correct case (A)	#case exclusively correct (B)	(B) / (A)
Context lumping	6,031(0.286)	1,122	0.186
RF	5,186(0.244)	493	0.095
CNN	8,213(0.390)	935	0.114
RNN	8,191(0.389)	1023	0.125

In conclusion, although some of our single algorithms such as context clustering or RF do not outperform the baselines, there are cases where each single algorithm is able to exclusively answer correctly, which we believe we were able to exploit in our proposed rank aggregation method.

5. CONCLUSIONS

In this paper, we tackled quote recommendation by exploring four single recommendation approaches considering different aspects of the context. And we presented new rank aggregation methods for maximizing performance. Over our datasets, we showed that the proposed algorithm (top-k RM RA) outperforms the best baseline by up to 46.7%. In the future, we plan to extend our research to recommend common phrase which has wider applications in the real world.

6. REFERENCES

[1] He, Q., Pei, J., Kifer, D., Mitra P., and Giles, C. L., 2010. Context-aware citation recommendation. In *Proceedings of the 19th international conference on World wide web* (Raleigh, NC, USA, April 26 - 30). WWW '10. ACM, New York, NY, USA, 421-430. DOI=http://dx.doi.org/10.1145/1772690.1772734

[2] Hochreiter, S. and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* (Nov. 1997). 1735-1780

[3] Huang, W., Kataria, S., Caragea, C., Mitra, P., Giles, C. L., and Rokach, L. 2012. Recommending citations: translating papers into references. In *Proceedings of the 21st ACM international conference on Information and knowledge management* (Maui, HI, USA, October 29 - November 02, 2012). CIKM '12. ACM, New York, NY, USA, 1910-1914.DOI=http://dx.doi.org/10.1145/2396761.2398542

[4] Kim, Y. 2014, Convolutional Neural Networks for Sentence Classification, In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (Doha, Qatar, October 25-29, 2014). EMNLP '14. ACL

[5] Liaw, A. and Wiener, M. 2002. Classification and regression by randomForest. *R News* (2002). 2(3):18–22

[6] Rangrej, A., Kulkarni, S., and Tendulkar, A. V. 2011. Comparative study of clustering techniques for short text documents. In *Proceedings of the 20th international conference companion on World wide web* (Hyderabad, India, March 28 - April 01, 2011). WWW '11. ACM, New York, NY, USA, 111-112. DOI=http://dx.doi.org/10.1145/1963192.1963249

[7] Tan, J., Wan, X. and Xiao, J. 2015. Learning to recommend quotes for writing. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (Austin Texas, January 25–30, 2015). AAAI'15. AAAI Press, USA, 2453-2459

[8] Young, H. P. 1974. An axiomatization of Borda's rule. *J. Econ. Theory* 9, 1, 43-52