Ranking and Summarization using word-embedding at NTCIR-12 MobileClick Task

Shinsuke Yokoyama
Rakuten, Inc.
shinsuke.yokoyama@rakuten.com

Sho Nakamura
Rakuten, Inc.
sho.nakamura@rakuten.com

Risa Kitajima
Rakuten, Inc.
risa.kitajima@rakuten.com

Yu Hirate
Rakuten Institute of Technology
yu.hirate@rakuten.com

ABSTRACT
Our team's approach is based on word-embedding. We converted queries and iUnits to vectors using word2vec. To these generated vectors, ranking-generation methods and summarization methods are applied

Keywords
Word2vec, Learning to Rank, Information Retrieval, MobileClick

Team Name
R-Panda

Subtasks
Japanese iUnit Ranking

Japanese iUnit Summarization

1. INTRODUCTION
In real-world web search, the most of queries are short and has insufficient information. We need to complement additional information. That makes generation of search-ranking difficult. In mobileclick2 task, not only query but also iUnit is a-bit-longer short-text. This makes this task more difficult.

In short-text analysis, it's common to compensate its context by adding from external knowledge data. This time, we do it through word2vec.

Since Mikolve published word2vec model [1], word-embedding has been one of very popular topic in text mining field. Varieties of its applications are reported in text-mining tasks. As Omer Levy pointed out [2], word2vec has robust performance in many benchmarks. This property is very useful to utilize data in web.

2. MODELS OF RANKING SUBTASK (JAPANESE)

2.1 Query classification
In the training and test dataset, there are several mixed types of queries, not they tend to be just randomized but categorized. We began to divide the queries into the following seven classes in training data

- Actors and musicians
- Politicians
- Athletes
- Particular places or stores
- Geological locations with purposes
- Common noun terms
- Natural sentences

As for test data, it doesn't have politicians or particular places or stores. We decided not to use these two classes to build a model. Then by merging actors, musicians and athletes, we ended up with keeping the following four classes both for training and test data: famous people, places, common noun terms and natural sentences.

Based on the classification, we used them with two approaches. One is simply dividing the training and test set by the query class. The other is to add these classes as dummy variables to features by *1-of-K* representation.

2.2 Word2vec Feature Generation
To make word2vec model, Bing's search result text are used as external knowledge data. These search results include variety of topics related to the queries; also the qualities of documents are checked to some degree. Therefore, they have good quality for external knowledge data. To build word2vec model, we use gensim [4]. The following is the procedure.

1. Delete contents of "Script" and "Style" tag from HTMLs
2. Apply morphological analyzer (Mecab) and split it to the Part-of-speech.
3. Filter out all the Part-of-speech other than noun.
4. Delete new-line among a document; Then feed them to gensim's word2vec API.

Most of queries and iUnits are split into multiple terms. When calculating a vector of iUnits and queries, we simply use an average vector of each term's vector.

2.3 Ranking

2.3.1 Approach 1: Predicting importance by Smoothing Method
It is widely known word2vec's property that similar terms are converted closer coordinates than non-similar terms. If we assume that similar iUnits have close importance values, an importance of iUnit is predicted by its neighbor's importance. Based on this idea, we use weighted average of k-nearest iUnits' importance as predicted value.

The Algorithm is

1. Classify target iUnit's query (by method 2.1)
2. Select k-nearest iUnits (this time k=4) from iUnits whose query is the same class as the target query.

3. Calculate weighted average of those iUnits importance by following expression.

$$\widehat{GG}_0 = \frac{1}{n}\sum_{i=1}^{n} W_i \cdot GG_i$$

With

$$W_i = \frac{1}{\|x_0 - x_i\|^2}$$

$$GG_i = (iUnit's\ importance)$$

$$x_i = (i_{th}\ nearest\ iUnit's\ vector)$$

$$x_0 = (target\ iUnit's\ vector)$$

2.3.2 Approach 2: Learning To Rank

Although iUnit is a short sentence, typical learning to rank approaches should be able to be applied with generated features. For iUnit Ranking Subtask, intent and intent probability are not given. Thus, regardless of intent for each query, we directly learn and predict global importance. We chose Random Forest [5] as a rank generator from RankLib [6]. This depends on the ranker part of MART [7].

2.4 Noise Cancellation

Looking into test data, compared to training data, there are more noisy iUnits, which should not meet any of intents for the query. We introduced two heuristic ways to displace them. Both of them focus on the trailing part of iUnits. One is that the iUnits end with some special symbols such as exclamation and question mark supposing that these iUnits don't give objective answers for the user query. The other is that the last token of iUnit is honorific term. Almost all of iUnits end with indeclinable words such as noun, which mean they are not like sentences but bullet point descriptions, especially honorific forms tend to be spam among them on this particular dataset.

3. RESULTS OF RANKING SUBTASK (JAPANESE)

In 2.2 process, we created word2vec model. With this model, we converted each iUnit in training dataset to vector and use them as input features. Figure. 1 shows the result.

Since we can't know the importance of test dataset, all evaluation here is based on test dataset. For smoothing approaches, we applied *leave-one-out* cross validation per query: leave one query and predicting its iUnits' importance from the other queries' iUnits. On the other hand for learning-to-rank approaches, 5-fold cross validation were applied instead because *leave-one-out* takes time but it performed almost the same well. This means out of 5,000 queries 1,000 are used for test and the remaining are for training set. For comparison, we generated LDA features and applied the same learning to rank algorithm.

3.1 Discussion

Table 1. Q-measure of learning word2vec features by smoothing model

	Q-measure
Random baseline	0.7728
No query class	0.8385
Seven query class separately predicted	0.8250

As for the smoothing model, we began with predicting iUnits' importance from the same query's iUnits. The Q-measure value is 0.888 in seven query-classes model. But when predicting another query's importance from the queries in the same class, the result is much lower q-measure value, 0.838. This means that selecting similar text by word2vec model works to a certain degree. But when a query is changed, having similar word2vec vector is not enough evidence to have similar importance. Although two queries are in a same class, the tendencies of high-importance iUnit are very different. In general text search, features that calculated using both query and iUnit could compensate this problem, but our model doesn't have this type of features.

At the same time, this smoothing method highly depends on the density of data. If there are some similar queries in training data, the prediction would be more accurate. But if not, the advantages of our model are lost. This time, training data set have only 100 queries, and most of queries are combination of multiple keywords. This time, the number of queries in training dataset is too small to make this model work fine. The direct dependence to number of training data is one of the bottlenecks of the smoothing based prediction model.

Table 2. Q-measure of learning word2vec features by learning-to-rank

	Q-measure
Word2vec, 100 features	0.8527
Word2vec, 100 features separately learned inside the query class	0.8380
Word2vec 100 + class representative features	0.8586
LDA features	0.8228
LDA features learn inside the query class	0.8326

Table 2 shows the results of the learning-to-rank approaches. The q-measure scores show the effectiveness of this approach. In word2vec, the meaning of each vector is not clear, but learning to rank algorithms learned the relations between them and importance. Compared to the traditional features like TF-IDF, the each vector is not strong, but those vectors can help task the sematic search.

When compared word2vec based learning to rank with LDA based one, word2vec features results in higher scores than LDA. This is partially because LDA is not robust to noises. A web-page contains not only an article, but also ads and UI parts like footer, header. Those elements prevent LDA from working as it designed. On the other hand, word2vec learn the vector by keywords around it. So those elements don't affect result if the same keywords don't appear in the UI components.

As for using query's class or not, the number of training data is a problem If we divided 100 queries into 7 classes, the average number of queries and iUnits in each group is less than 15 and

770. It's not enough number for learn something from 100 features data.

4. MODELS OF SUMMARIZATION SUBTASK (JAPANESE)

Summarization task is divided into three parts; assigning each iUnit to intent, calculation of penalized importance of iUnits and iUnits layering.

4.1 Intent Labeling

In iUnit ranking subtask, iUnits are converted to vectors. In the same way, intents are converted to vectors. For each iUnit, one intent is assigned based on cosine similarity between the vectors.

4.2 Calculation of importance

In iUnit ranking sub task, we calculated each iUnit's predicted importance. Since each iUnit's utility has decay according to their position, using shorter iUnit results in better M-measure score than using longer one, if their importance is the same. Thus, to take decay factor into consideration, we calculate each iUnit's importance using its length of iUnit and ranking sub task's result.

Given a constant parameter α, the weight w_i calculated by iUnit ranking sub task for iUnit i, and length of iUnit l_i, the re-calculated weight w_i' of iUnit i is given by:

$$w_i' = (1.0 - \alpha) * w_i + \alpha * \frac{280}{l_i}.$$

We set α = 0.3 heuristically. This calculation makes the weight of longer sentence smaller while importance of shorter sentence larger by dividing total length of summarization by the length of iUnit. 280 is the length limitation of characters for this Japanese task.

4.3 IUnit Layering

Next we determine layer of each iUnit: which iUnit should be listed on the first layer and the second layer. Here we assume that iUnits that have large standard deviation between similarity score to and each intent should be listed on the second layer, because those sentences have large similarity to certain intent and we can put them under the link from the intent. Here we set 0.05 as threshold heuristically and if the standard deviation is higher than 0.05 then the iUnit is put on second layer, while if the standard deviation is lower than 0.05 then the iUnit is put on first layer. Then we calculate each weight for the intent itself by calculating average of length-normalized weight of all intents, which are assigned to the intent. On both of first and second layers, iUnits are sorted by its length-normalized weight.

5. RESULTS OF SUMMARIZATION SUBTASK (JAPANESE)

5.1 Results

Table.3 shows that M-measure for our proposed method and other baseline methods. Our proposed method gives higher score on M-measure rather than baseline methods in average of all queries.

Table 3. Mean of M-measure

	Score
Proposed method	18.4203
LM-based two-layer baseline	17.4376
Random baseline	15.0373
LM-based baseline	12.799

Table 4 and Table 5 show five queries which show highest and lowest M-measure, respectively. As for MC2-J-0083, iUnit and its intent is missing in dataset. Other four low M-measure score on queries are classified to "geological locations with purposes".

Table 4. Five highest M-measure queries

Qid	Query	M-measure
MC2-J-0042	ジャガー	52.6187
MC2-J-0004	シャロン･ストーン	41.54
MC2-J-0022	博物館 豊橋市	40.7866
MC2-J-0061	Ups	40.2946
MC2-J-0073	バゲット	39.9209

Table 5. Five lowest M-measure queries

Qid	Query	M-measure
MC2-J-0083	コールスローを作る方法	0.0000
MC2-J-0034	ホテル 勝浦駅	3.3758
MC2-J-0035	カフェ 川之江駅	4.6622
MC2-J-0038	皮膚科 京都市北区	4.7315
MC2-J-0037	銀行 いすみ市	4.8742

5.2 Discussion

As for MC2-J-0034, all iUnits are listed on the second layer. It means our threshold, which defines if each iUnit should be listed on the first layer or the second layer, sometimes too small to divide iUnits into two layers. If we loosen restriction for iUnit to be put on the first layer by setting larger value as threshold, more iUnits are listed on the second layer in this case. On the other hand, as for MC2-J-0042, 11 iUnits are chosen to be listed on the first layer with same threshold.

Here, we focus on the number of intents for each query. The average number of intents for queries listed on Table. 4 is 13.25, while the one for queries listed on Table. 5 is 2.6. From this, we suppose that our threshold 0.05 is fit if the query has few intent, and not adequate for the case that the query has many intents. Although we applied the same value as threshold for all queries here, it is assumed that we can improve our proposed method by

introducing logic to decide it depending on dispersion of similarity between intent and iUnits.

6. CONCLUSION

We described our approach in this paper. In the ranking subtasks, the features are generated from query classification and word2vec. The 1st model are based on smoothing based prediction, the 2nd model is based on MLR. The both achieved higher score than the baseline.

In the summarization subtasks, the features are based on the ranking subtask's result and word2vec, also result in higher score than the baseline.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado,Greg, and Dean, Jeffrey. 2013. Distributed representations of phrases and their compositionality. In Advances on Neural Information Processing Systems

[2] Omer Levy, Yoav Goldberg, and Ido Dagan. 2014. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *TACL 2015.*

[3] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= http://doi.acm.org/10.1145/161468.16147.

[4] https://radimrehurek.com/gensim/

[5] L. Breiman. 2001. Random Forests. Machine Learning 45 (1): 5-32

[6] https://sourceforge.net/p/lemur/wiki/RankLib/

[7] J.H. Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, 29(5): 1189-1232

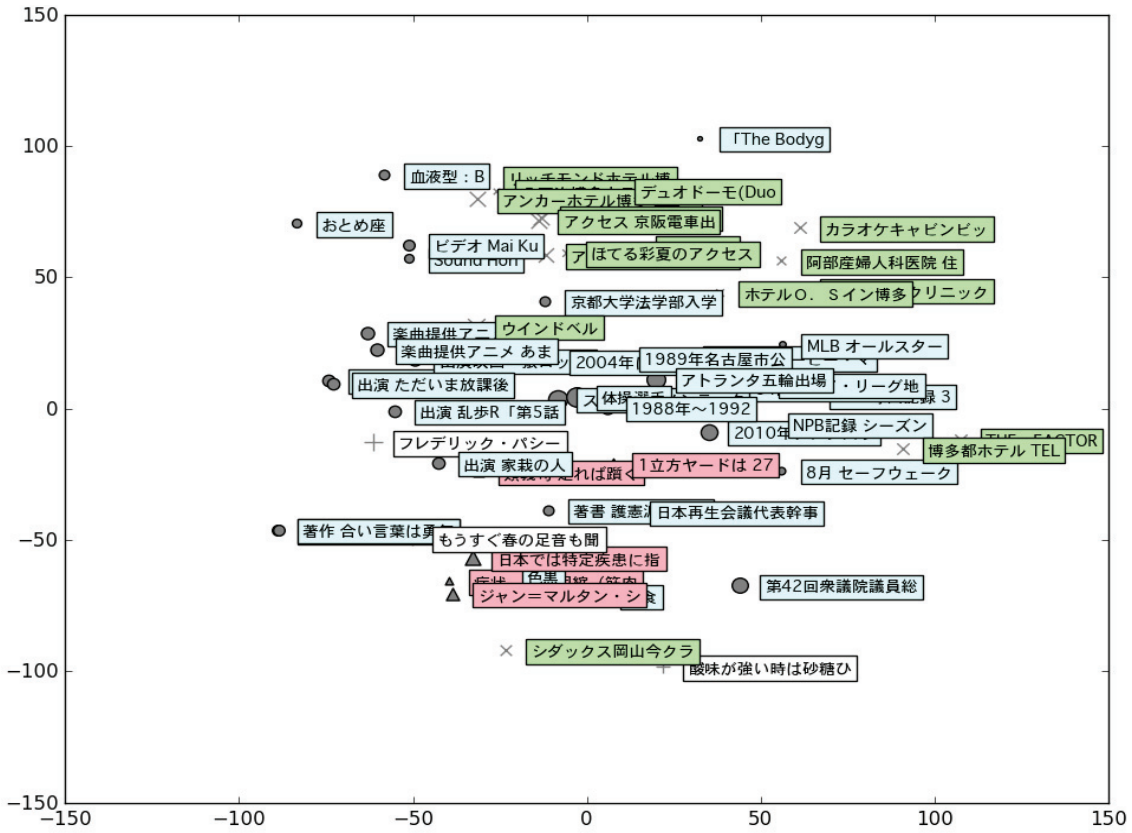

Figure 1. iUnits in word2vec space

IUnits vectors are reduced to two dimensional space by applying t-SNE, commonly used dimension reduction technique.

The sizes of marks show its importance, the colors of text-boxes corresponds to its query class.