Analysis of Similarity Measures between Short Text for the NTCIR-12 Short Text Conversation Task

Kozo Chikai
Graduate school of information science and technology, Osaka University
chikai.kozo@ist.osaka-u.ac.jp

Yuki Arase
Graduate school of information science and technology, Osaka University
arase@ist.osaka-u.ac.jp

ABSTRACT

According to rise of social networking services, short text like micro-blogs has become a valuable resource for practical applications. When using text data in applications, similarity estimation between text is an important process. Conventional methods have assumed that an input text is sufficiently long such that we can rely on statistical approaches, *e.g.,* counting word occurrences. However, micro-blogs are much shorter; for example, tweets posted to Twitter are restricted to have only 140 character long. This is critical for the conventional methods since they suffer from lack of reliable statistics from the text.

In this study, we compare the state-of-the-art methods for estimating text similarities to investigate their performance in handling short text, specially, under the scenario of short text conversation. We implement a conversation system using a million tweets crawled from Twitter. Our system also employs supervised learning approach to decide if a tweet can be a reply to an input, which has been revealed effective as a result of the NTCIR-12 Short Text Conversation Task.

Team Name

Oni

Subtasks

Short Text Conversation (Japanese)

Keywords

Twitter, short text, similarity, micro-blog

1. INTRODUCTION

Micro-blogging services (*e.g.,* Twitter[1], Google+[2], Weibo[3], and Tumblr[4]) have been popular, so that they produce the enormous amount of text every second. They are a kind of blogging services but each post should be short. A characteristic of such microblogging services is that users actively communicate with each other. Therefore, they provide us a huge amount of conversational text pairs. Short Text Conversation (STC) Japanese Task[5] in NII Testbeds and Community for Information access Research (NTCIR[6]) make use of this characteristic to develop a conversation system between a computer and human. Even with latest technologies in natural language processing, it is still challenging to generate natural replies to human's input from scratch. As the first step of the conversation system, STC task turns the reply-generation process into an information retrieval task. STC task gives a pool of tweet conversations; post tweets and their replies, which can be crawled from Twitter, and asks participants to search appropriate replies from the pool for an input post .

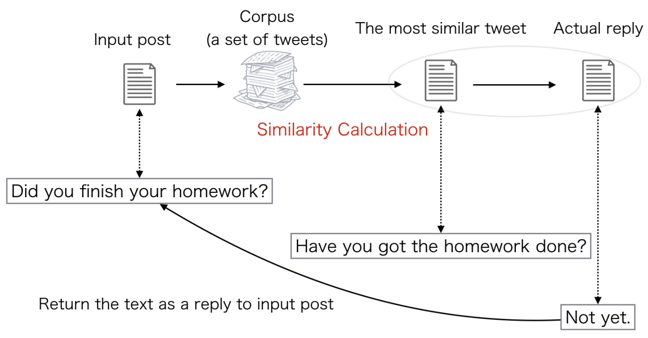
Input post
Corpus
(a set of tweets)
The most similar tweet
Actual reply
Similarity Calculation
Did you finish your homework?
Have you got the homework done?
Return the text as a reply to input post
Not yet.

Figure 1: System design

We participate in the STC task. Figure 1 shows our system design. The principle of our system is that replies to tweets similar to an input are also effective as the input's replies. Our system first searches for tweets that are similar to the input, and then returns their replies. Thus the key is how we can precisely estimate similarity between tweets, which are extremely short.

The standard way to estimate similarity between texts is; 1) represent text by any vector space models, and 2) compute similarity between the vectors. If text is sufficiently long, a simple approach works well. For example, we may use bag-of-words vectors. However, we inevitably suffer from sparsity problem when handling short text. As we can easily imagine, only several words appear in short text. Due to this characteristic, vectors representing tweets become sparse, which results in degenerated similarity estimates.

In this study, we compare conversational vector space models and similarity measures to handle short text. We implement the conversation system and evaluate their effec-

[1]`https://twitter.com/`
[2]`https://plus.google.com/about?hl=ja`
[3]`http://www.weibo.com/login.php`
[4]`https://www.tumblr.com/`
[5]`http://ntcir12.noahlab.com.hk/japanese/stc-jpn.htm`
[6]`http://research.nii.ac.jp/ntcir/index-ja.html`

tiveness. In addition, our system uses a supervised learning method to learn if a pair of tweets can be a post-reply pair. Analysis of the formal run results shows that the supervised method is effective especially for typical conversations in Twitter.

2. RELATED WORK

Previous studies focusing on Twitter have used heterogeneous data available in it, such as follower-followee relationships, location information, timestamps, and hashtags. Yan *et al.* [1] propose a graph ranking algorithm using two kind of graphs; an undirected graph representing similarities between tweets and a directed graph representing user's follower-followee relationships. Using the ranking algorithm, they developed a recommendation system that suggests tweets to take a look. Zhao *et al.* [2] propose the Twitter-LDA model in which LDA [3] has been adapted to handle tweets. They show that restricting a tweet to have only one topic leads to higher accuracy for identifying topics in tweets.

These studies use the heterogeneous data in Twitter. Such data are definitely effective to improve performance of a system designated to a specific application. However, at the same time, systems depend on these heterogeneous data are hard to extend to use different kinds of data source other than Twitter. In order to make our method as flexible to different kinds of applications as possible, we use only text (*i.e.,* tweets) and post-reply relationships in this study.

The standard approach to estimating similarity between text represents texts as a vector and computes similarity between the vectors. Typical vector space models are bag-of-words and its variations using TF-IDF for weighting, and topic models (*e.g.,* pLSI [4], LDA [3]). However, Mihalcea *et al.* [5] and O'Shea *et al.* [6] show that topic models fail to extract correct topics if input text is too short due to lack of word co-occurrence statistics. Guo *et al.* [7, 8] proposed Weight Text Matrix Factorization (WTMF) model that aims to complement the sparseness in vectors representing tweets by assuming that unobserved words in a tweet should not be relevant to the tweet. On the other hand, Gabrilovich *et al.* [9] complements the sparseness using Wikipedia[7] corpus. As an orthogonal approach, we can use word or document embeddings generated, using deep neural network with an enormous text corpus [10, 11, 12, 13, 14].

3. MAPPING TWEETS TO VECTORS

In this section, we briefly summarize conventional vector space models to convert a tweet to a vector. We compare these methods in our system.

3.1 WTMF model

WTMF model aims to vectorize sparse text like tweets. Its principle is that let unobserved words in a tweet be irrelevant to the tweet. As Figure 2 shows, WTMF approximates a tweet-word matrix $X \in \mathbb{R}^{M\times N}$ by the product of a matrix $P \in \mathbb{R}^{K\times M}$ and a matrix $Q \in \mathbb{R}^{K\times N}$. Accordingly, each tweet s_j is represented by a K–dimensional latent vector $Q_{\cdot,j}$. In a similar fashion, vector $P_{\cdot,i}$ refers to a word w_i.

The matrices P and Q are derived by minimizing the objective function of Equation (1).

[7]https://www.wikipedia.org/

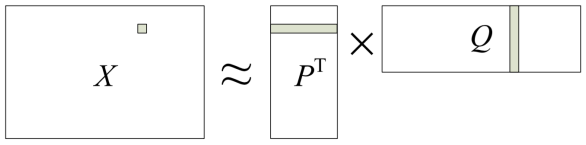

Figure 2: Weighted Textual Matrix Factorization [7]

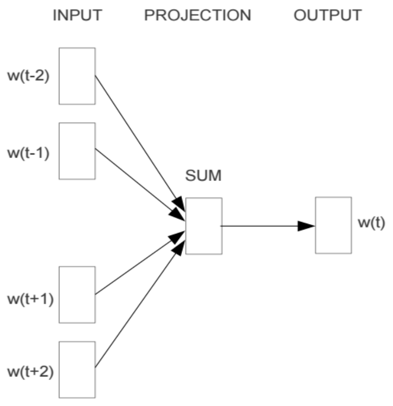

Figure 3: CBOW Model [11]

$$\sum_{i,j} W_{i,j}(P_{\cdot,i}\cdot Q_{\cdot,j} - X_{i,j}) + \lambda||P||_2^2 + \lambda||Q||_2^2 \ , \qquad (1)$$

$$W_{i,j} = \begin{cases} 1 & (\text{if } X_{i,j} \neq 0) \\ w_m & (\text{if } X_{i,j} = 0) \ . \end{cases} \qquad (2)$$

Last two terms in Equation (1) are regularizers to avoid over-training. $W_{i,j}$ is defined by Equation (2), standing for the weight of unobserved words. Since most of cells in X are unobserved words (0 weight), the impact of observed words is significantly diminished. Therefore a small weight w_m is assigned for each unobserved words in X in order to preserve the influence of observed words.

3.2 Word Embedding

Instead of directly generating vectors of tweets, we may use word embedding to represent a tweet. Recent studies [10, 11, 12, 13, 14, 15] have used deep neural network for word embedding that generates low-dimensional vectors representing words.

We use word2vec [10, 11, 12, 13] with CBOW model. As in Figure 3, CBOW is modeled to predict a word from its surrounding context.

4. IDENTIFY REPLY TO A TWEET

In this section, we describe methods to identify a tweet that can be a reply to an input.

4.1 Cosine similarity

Cosine Similarity is one of the standard methods to calculate the similarity between vectors:

$$\cos(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|} \ , \qquad (3)$$

where $\boldsymbol{x}$, $\boldsymbol{y}$ are vectors. The closer to 1 the output figure is, the more similar $\boldsymbol{x}$ and $\boldsymbol{y}$ are.

We rank tweets using the cosine similarity, and then extract replies to the similar tweets as replies to the input.

4.2 Classification-based method

Since our corpus consists of tweets and their replies, we can use a supervised learning approach to directly learn if a pair of tweet can be a post-reply pair.

Specifically, we use Random Forest [16] for its superior accuracy and lighter computational cost. We regard post-reply pairs in the corpus as positive examples. Since there is no explicit negative examples, we synthesize negative examples by pairing a tweet with a randomly picked one. For these examples, we concatenate vectors of each tweet, and train a Random Forest. The trained model describe if an input tweet pair is positive (*i.e.,* can be post-reply pair) of negative.

5. DESIGN OF CONVERSATION SYSTEM

STC task provides participants a list of 500K pairs of tweet IDs posted in 2014, *i.e.,* 500K tweets and their replies, in total one million tweet IDs. We crawled the tweet text using Twitter API[8]. Due to the lag between the time when the task organizers prepared the list and the actual crawling, some tweets had become unavailable. We have collected $428,124$ pairs (post and comment tweets) and $64,395$ tweets (residue tweet, which either post or comment tweet is unavailable), in total $920,643$ tweets.

Figure 4 shows the overview of our conversation system. The system takes an input tweet that needs replies. It searches and ranks tweets as described below. Then, using the top–50 tweets, the system generates the final ranking of potential replies. Specifically, starting from the top-1 tweet, its pair is inserted into the list, and then the tweet itself is inserted to the list as the second rank. This means if the tweet is a post tweet in the corpus, we rank its reply higher, and similarly, if the tweet is a reply tweet, corresponding post tweet is ranked higher. This is because we observe that direction of post–reply can be reversed in some context as following example shows.

post : 今日の北海道めっちゃ涼しい
(It is pretty cool in Hokkaido today.)

reply : 夏の北海道の気候って最高だよな
(Summer is the best season in Hokkaido.)

We include the tweet itself since they are useful as a reply agreeing with the input tweet. When the similar tweet is a residue tweet, we include only the residue tweet into the reply list.

We implement the following five methods for ranking tweets in the system and evaluate their performance.

① TF-IDF
Tweets are represented by a bag-of-words matrix with TF-IDF weighting. Similarity between tweets is calculated using cosine similarity (Equation(3)). Finally, top-50 similar tweets are used to generate the reply list.

② WTMF
Vectors representing tweets are generated by WTMF model described in Section 3.1 . Similarity between tweets is calculated by cosine similarity (Equation(3)).

③ Word2vec → TF-IDF
We first find tweets relevant to an input tweet using word embedding. For this purpose, word vectors are generated using word2vec with Wikipedia dump data[9]. Then 20 most similar words for each noun and verb in the input tweet are extracted using the word vectors. Tweets containing these words are searched and set as candidates. Finally, these candidate tweets are ranked based on their similarities to the input tweet in the same manner with ① TF-IDF.

④ Random Forest → TF-IDF
Tweets are represented by vectors in the same manner with ① TF-IDF. Using these vector, a classifier is trained as described in Section 4.2. Specifically, we sample 300K post-reply pairs from the corpus as positive examples and another 300K randomly paired tweets as negative examples. Random Forest is trained as a binary classifier; the number of trees are set to 100. For speeding up the training process, the dimension of a vector representing tweet is reduced to be 100 using singular value decomposition [17].

The vector of the input tweet is concatenated with that of tweet in the corpus, and then fed to the trained classifier. As a result, we collect all tweets decided as positive, *i.e.,* regarded as post–reply pairs. Finally, these tweets are ranked in the same manner with ① TF-IDF method.

⑤ Random Forest + TF-IDF
This method is a variation of ④. It ranks the positive tweets by summing the likelihood of being positive, which is output of the Random Forest classifier, and cosine similarity computed in the same manner with ①.

We note that other conventional approaches for generating vectors of tweets, *i.e.,* LDA [3], HDP [18], and doc2vec [15], as well as similarity measures proposed in [19] have been also evaluated. However, due to their poor performance compared to the methods described in this section, we omit reporting their results.

6. PRELIMINARY EXPERIMENT

In this section, we describe a preliminary experiment to understand characteristics of our system.

6.1 Data set

STC task provides participants a development that consists of 225 tweets. Each tweet has 10 tweets that are assigned scores by humans as 0, 1, 2, or NA:

0 : The tweet does not make sense as reply.

[8]`https://dev.twitter.com/`

[9]`http://dumps.wikimedia.org/jawiki/latest/jawiki-latest-article.xml`

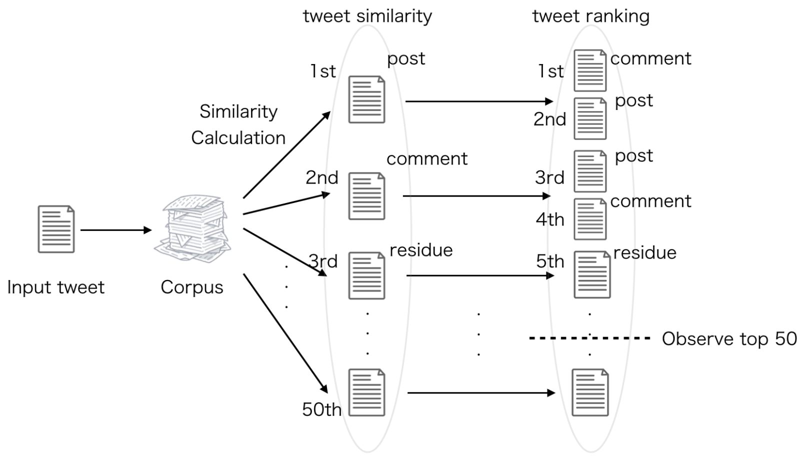

Figure 4: Ranking Method

1 : The tweet is acceptable as a reply in some context.

2 : The tweet is appropriate as reply.

NA : The input tweet does not make sense.

Annotation has been performed via crowd sourcing, where at most 10 annotators assigned a score to each tweet. We conduct a preliminary experiment using the development set to investigate effectiveness of each method. We average scores of tweets and regard ones that have more than or equal to 0.7 as correct replies, excluding tweets and their pairs that have assigned NA. Consequently, we have 179 input posts (tweets) with potential replies.

6.2 Evaluation Criteria

The STC task requires to generate a ranked list of potential replies to an input tweet, and thus we employ five evaluation criteria; P@k, TOP@k, MRR, MAP, nDCG@k.

P@k (Precision at k) evaluates the precision of top–k replies, while TOP@k is a simplified measure of P@k, which evaluates if a correct reply is contained in the top–k ranking.

MRR (Mean Reciprocal Rank) evaluates if correct tweets is ranked higher. Specifically, it only observes and evaluates top most correct tweet in a ranking. MRR checks the ranked tweets from top, then calculates reciprocal rank:

$$MRR = \frac{1}{N}\sum_{i}^{N}\frac{1}{r} \quad , \tag{4}$$

N is the number of rankings, and r is the rank of the correct tweet.

MAP (Mean Average Precision) computes the mean of the average precision scores:

$$AP_i = \frac{1}{R_i}\sum_{j} I(j)\frac{count(j)}{j} \quad , \tag{5}$$

$$MAP = \frac{1}{N}\sum_{i}^{N} AP_i \quad , \tag{6}$$

where R_i is the number of correct tweets contained in top-i ranking, $I(j)$ is a binary indicator whether ranked tweet is correct or not, and $count(j)$ is the number of correct tweets contained in the top-j rank.

nDCG (Normalized Discounted Cumulative Gain) can evaluate ranking quality when the ground-truth provides scores representing appropriateness of each item. DCG appreciates that higher scored tweets are ranked higher. DCG accumulated at a particular rank k is defined as:

$$DCG_k = rel_1 + \sum_{i=2}^{k}\frac{rel_i}{\log_2 i} \quad , \tag{7}$$

where rel_i represents the score that the i–th ranked item has. Since the number of correct replies may be diverse for each input tweet, nDCG normalizes DCG against an ideal DCG. It can be computed by sorting the second items in the gold-standard by descending order and computes its DCG. Finally, nDCG is defined by Equation(8).

$$nDCG_k = \frac{DCG_k}{IDCG_k} \quad . \tag{8}$$

All the evaluation criteria range from 0 to 1, where higher the figure is, better the ranking quality is.

6.3 Results

Figure 5 and 6 show performance of each method. Overall, ① TF-IDF and ③ Word2vec → TF-IDF outperformed others in all evaluation metrics. As we look into details, ① outperforms ③ for TOP@k and MAP, while ③ outperforms ① for MRR and P@k. This means that ① has a larger number of correct replies in its ranking, but the ranking quality itself is degraded than ③. This is the positive effect of tweet filtering perfomed using word vectors in ③.

② WTMF performs worse than ① TF-IDF, where only the difference between ① and ② is the vector space model. This result suggests that the principle of WTMF, *i.e.,* unobserved words should not be relevant to a tweet, is too strict when handling extremely short text like tweets.

The performance of ④ Random Forest → TF-IDF and ⑤ Random Forest + TF-IDF show similar trend with ③

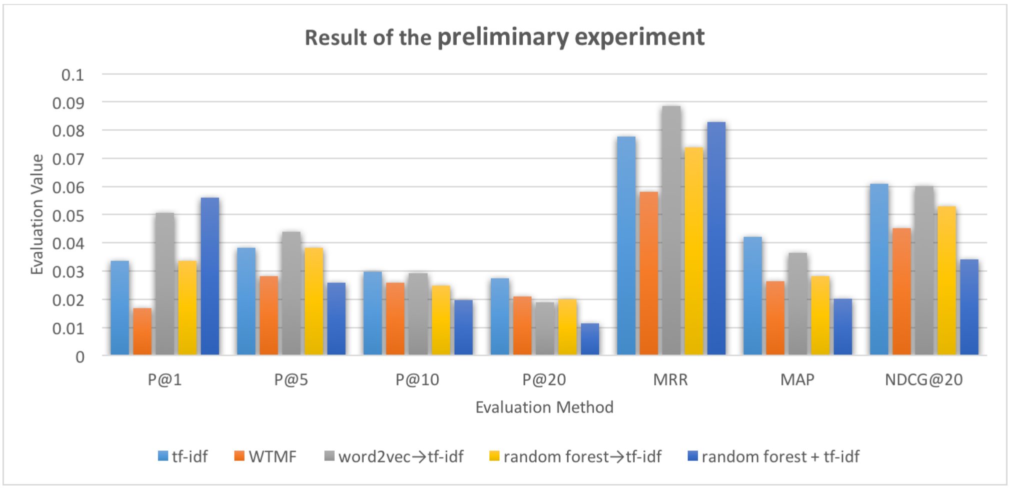

Figure 5: Results of P@k, MRR, MAP, and nDCG@20

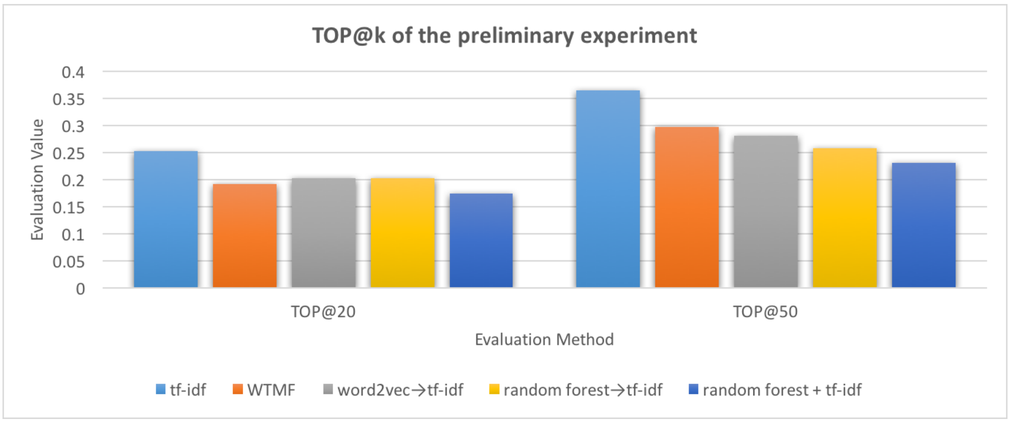

Figure 6: Result of TOP@k

word2vec → TF-IDF. When we observe replies output by ④ and ⑤, we find that these methods work better on shorter input tweets. For example, given an input tweet of "Thank you for following me," these methods find correct reply of "Welcome, thank you too for following." This is because such a short conversation is typical in Twitter, thus there should be a larger number of examples included in the training set. On the other hand, post–reply tweets with long text are rare in the corpus, thus they are hard to generalize while training.

7. FORMAL RUN

In this section, we discuss results of the formal run in STC Task.

7.1 Our submission

We submitted ranked lists of tweets that were generated by methods described in Section 5, which are reported in the task overview [20]. Table 1 shows correspondences between submission names and method names in Section 5.

STC task evaluated submissions by three criteria: nDCG@1, nERR@5, and Accuracy. Details of these evaluation measures are described in the overview.

Table 1: The Methods we submitted

Name of method	Technique
Oni-J-R1	④ Random Forest → TF-IDF
Oni-J-R2	⑤ Random Forest + TF-IDF
Oni-J-R3	① TF-IDF
Oni-J-R4	③ Word2vec → TF-IDF
Oni-J-R5	② WTMF model

7.2 Results

Figure 7 summarizes the evaluation results. There are 3 to 5% gaps between the best-performed method and the worst one. Overall, Oni-J-R1 (④ Random Forest → TF-IDF) performed best in all criteria. This results shows that the supervised learning based approach is effective to distinguish post-reply pairs and random pairs.

The task overview reports that the five methods we developed achieved higher ranks in $nERR$@5, A_{ccL2}@5, and $A_{ccL1,L2}$@5 compared to $nDCG$@1, A_{ccL2}@1, and $A_{ccL1,L2}$@1. This reveals that our methods failed to rank a good reply at the first position, but have relatively good ranking overall. Figure 8 shows the mean of scores (*i.e.*, labels 0, 1, and 2) given each reply at different ranks. It shows that the second

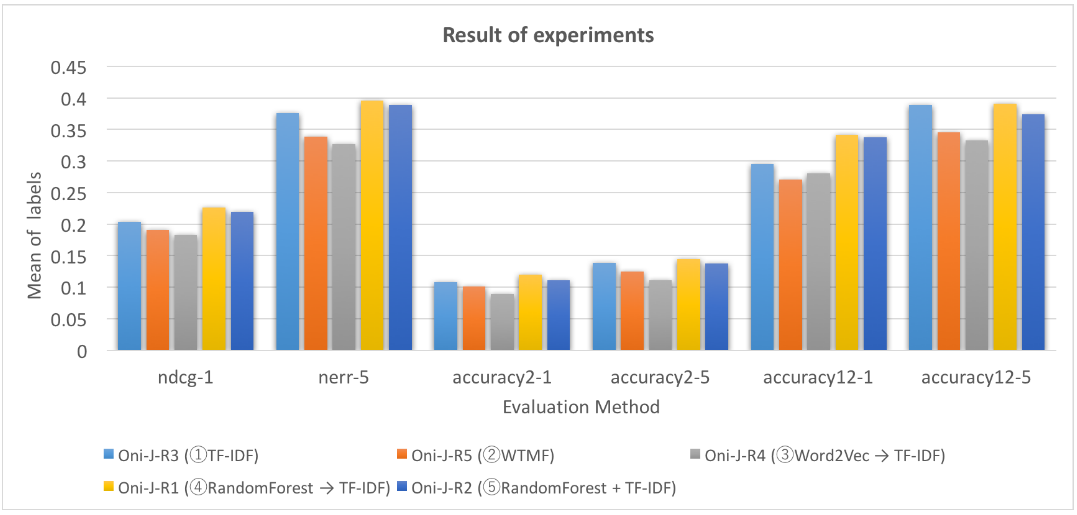

Figure 7: Result of nDCG@1, nERR@5, and Accuracy

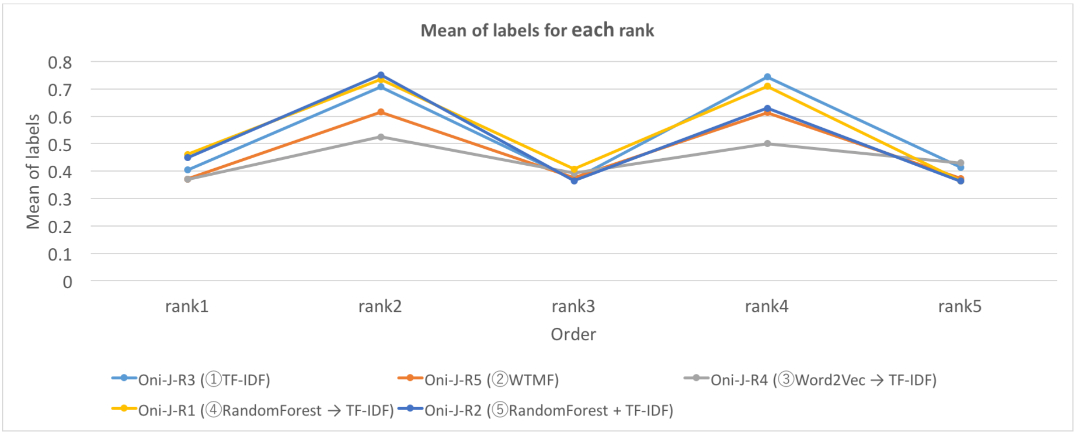

Figure 8: Mean of labels for each rank

and forth ranked replies have much higher scores than the first and third ranked replies in all methods. We can see that our assumption that replies to the similar tweet with an input should be also effective as replies to the input did not hold. The reason is twofold. First, that assumption requires quite precise similarity computation to hold, and the methods we used are too far to reach that bar. If similarity computation is not satisfactory, replies to not-that-similar tweets are hopeless to be acceptable replies. Second, judging if a reply is acceptable or not is tough even for human. It is easy to imagine that tweets talking about the similar topic tend to be assigned label 1 or 2, since they are at least relevant.

As we discussed in Section 6.3, methods using Random Forest (Oni-J-R1 and Oni-J-R2) perform better on shorter tweets. Table 2 shows means of scores for different character lengths of inputs. We compare Oni-J-R1(④ Random forest → TF-IDF) to Oni-J-R3 (① TF-IDF). These two methods are different whether using Random forest as filtering at first or not. Therefore, if Oni-J-R1 has higher score (colored red in Table 2), it is effect of using Random forest to decide if a pair of tweet can be a post-reply pair. As you can see, Oni-J-R1 achieves higher scores when an input post has less than 40 characters. This trend is more remarkable when the input post has less than 20 characters. This is because short text like less than 20 characters is typical conversation in Twitter, and thus there are the larger number of training examples. For example, Table 3 shows the actual input post and replies produced by Oni-J-R1. In addition, in the case that input post is a short text, the number of the topic included in a tweet is few, thus it has lesser ambiguity to handle while training.

When we observe the replies generated by Oni-J-R4 (④ Word2vec → TF-IDF), they have different characteristics from others. Specifically, compared to replies generated by R3 (① TF-IDF), Oni-J-R4 produces replies that have relevant topics with the input but with wider scopes. For example, Table 4 shows the differences of replies between R3 and R4. This table shows that R3's replies pinpoint to the topic of "LINE [10]." On the other hand, R4's replies broadly covers topics in IT: "PC", "smartphone", and "online games." R4 did not outperform other methods, however this feature is interesting and has potential to produce replies to extend the conversation topics like humans do.

[10]http://line.me/ja/

Table 2: Effect of R1 and R3 due to the number of characters

input	$0 \leq$ # of characters ≤ 20		$20 <$ # of characters ≤ 40		$40 <$ # of characters ≤ 60		$60 <$ # of characters	
system	R1	R3	R1	R3	R1	R3	R1	R3
rank1	0.810	0.587	0.424	0.408	0.376	0.279	0.350	0.350
rank2	0.860	0.793	0.722	0.709	0.703	0.685	0.691	0.644
rank3	0.503	0.550	0.401	0.328	0.371	0.365	0.375	0.341
rank4	0.583	0.670	0.731	0.749	0.694	0.732	0.775	0.806
rank5	0.227	0.600	0.359	0.417	0.471	0.356	0.397	0.294

Table 3: Example of replies generated by Oni-J-R1

Input post	フォローありがとうございます！ Thank you for following!
rank1	こちらこそ、懐かしい画像をありがとうございました♪ よろしくお願いします！ Thank you too for sharing the nostalgic photos! Keep in touch.
rank2	RT ありがとです♪ フォローさせていただきました！ Thank you for retweeting. I had followed you!
rank3	お休みなさい♪ 今年も宜しくお願いします！ Good night. Wish we'll have fun together!
rank4	寝ます。おやすみなさい。 さっそく今日から仕事だ！ Going to bed. Good night. Gotta work from today!
rank5	んぅ いくぅ hmmm...

Table 4: Example of output from Oni-J-R3 and Oni-J-R4

Input post	LINE 使えんので用のある人はこっちでお願いします！ Cannot use LINE now. Please contact me via Twitter!	
Method	R3	R4
rank1	新しいアカウント作ったから教えてー！ Let me know your account! I've just created one.	PC で扱うようなでかい解像度で色々したりするんでなければいよいよ十分な感じなんです？ It should be satisfactory unless you process high-resolution images that require PC-like performance.
rank2	LINE 使えるようになった？ Has LINE recovered?	んー、パソコンで使えた機能の大半はもう使えてるよね。 単純に使える、だけだけど。　次の 8 コアのスマートフォン向けの cpu が出るからそれで単純に使える機能から卒業して、もうスマートフォンで十分になる機能は出るだろう Most functions in PCs are already covered by smart phones, although they are just functioning... They should be actually usable with the next 8-core CPU for smart phones.
rank3	パソコンの LINE ツンデレすぎるよ！ もう！ 早く携帯よとどけ！ LINE app for PC is hard to use... Get me a cell phone!	単発でやる環境に長くいたせいで、最近キャンペ参加できるようになったのですがなかなか上手い感じにキャンペ用キャラが組めないんですよね…… I cannot figure out how to organize my party for online campaign as I've been playing alone...

In either the preliminary experiment or formal run, Oni-J-R5(② WTMF model) did not outperform TF-IDF based models. As we discussed in Section 6.3, assumption in WTMF model may be too strict for extremely short text like tweets. Also failures in morphological analysis may have been affected the quality of its vector space model. Since tweets are informal and colloquial, the morphological analyzer trained by conventional documents produce lost of errors.

As a result of observation, we also found that when the input post was long (more than 40 characters), it was difficult to find appropriate replies. Let us take the following input post as an example.

Input post : んで今から金沢！　今更テラハ見てきまーす笑 [個人名] カーブと駐車できんくせに 直線くそ速い笑 生きて帰ってこれますように…笑
(We are going to Kanazawa! And see the movie "terrace house". [personal name] can't go round a curve nor park, but he drives toooo fast on a straight road. Wish we could come home safely...)

We expect a reply should answer the phrase of "生きて帰ってこれますように (Wish we could come home safely...)." However, our methods output replies talking about Kanazawa city by being too much influenced by the word "金沢." In order to solve this problem, we should identify the topic that input post mostly requires answers. This is our future work.

Finally, we found that the performance of the examined methods are diverse between the preliminary experiment and formal run. One reason is that annotated replies in the development set were extracted from the pool using Lucene [11]. Its search function is based on TF-IDF, and thus, ① TF-IDF should have been compatible in the preliminary experiment where we used the development set.

8. CONCLUSION

We take part in the NTCIR-12 STC Japanese Task and compare the state-of-the-art methods for a conversation system.

Results revel their characteristics to handle short text in the conversation system. Supervised learning is effective to decide if a reply is acceptable for an input when the input is short and contains only a few topics. As future work, we plan to work on identifying the topic in longer tweets that require answers in order to produce appropriate replies.

9. REFERENCES

[1] R. Yan, M. Lapata, and X. Li, "Tweet recommendation with graph co-ranking," In Proceedings of ACL, pp. 516–525, July 2012.

[2] W. X. Zhao, J. Jiang, J. Weng, J. He, E. Lim, H. Yan, and X. Li, "Comparing Twitter and traditional media using topic models," In Proceedings of ECIR, pp. 338–349, April 2011.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," Journal of Machine Learning Research, vol. 3, pp. 993–1022, May 2003.

[4] T. Hofmann, "Probabilistic latent semantic indexing," In Proceedings of SGIR, pp. 50–57, August 1999.

[5] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," In Proceedings of AAAI, vol. 1, pp. 775–780, July 2006.

[6] J. O'Shea, Z. Bandar, K. Crockett, and D. McLean, "A comparative study of two short text semantic similarity measures," In Proceedings of KES, pp. 172–181, Springer, March 2008.

[7] W. Guo, and M. Diab, "Modeling sentences in the latent space," In Proceedings of ACL, pp. 864–872, July 2012.

[8] W. Guo, H. Li, H. Ji, and M. Diab, "Linking Tweets to news: A framework to enrich short text data in social media," In Proceedings of ACL, pp. 239–249, August 2013.

[9] E. Gabrilovich, and S. Markovitch, "Computing semantic relatedness using Wikipedia-based explicit semantic analysis," In Proceedings of IJCAI, pp. 1606–1611, January 2007.

[10] T. Mikolov, W. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations.," In Proceedings of HLT-NAACL, pp. 746–751, June 2013.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint, arXiv:1301.3781, 2013.

[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in Neural Information Processing Systems 26, eds. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, pp. 3111–3119, Curran Associates, Inc., October 2013.

[13] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," arXiv preprint arXiv:1309.4168, 2013.

[14] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," In Proceedings of EMNLP, vol. 32, pp. 1532–1543, October 2014.

[15] T. Mikolov, and Q. V. Le, "Distributed representations of sentences and documents," In Proceedings of ICML, vol. 32, pp. 1188–1196, June 2014.

[16] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, October 2001.

[17] M. Udell, C. Horn, R. Zadeh, and S. Boyd, "Generalized low rank models," arXiv preprint arXiv:1410.0342, 2014.

[18] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," Journal of the American Statistical Association, vol. 101, pp. 1566–1581, November 2006.

[19] Y. Song, and D. Roth, "Unsupervised sparse vector densification for short text similarity," In Proceedings of HLT-NAACL, pp. 1275–1280, May 2015.

[20] L. Shang, T. Sakai, Z. Lu, H. Li, R. Higashinaka, and Y. Miyao, "Overview of the NTCIR-12 short text conversation task," In Proceedings of the NTCIR Workshop Meeting on Evaluation of Information Access Technologies (NTCIR-12) (to be published)., 2016.

[11] `https://lucene.apache.org/core/`