An Extractive Chat Summary Generation Method for Ecommerce Chatbots

Daolin Han, Xiao Song, Yong Cui

ABSTRACT

The usage of chatbot for handling the conversations is gradually increasing. By using this technique, companies can provide a real-time and efficient channel to deliver the information to their users. Chatbot can response significant volume of frequent-asked and typical questions, which is bound to reduce the company human resource consumption. However, sometimes a chatbot cannot satisfy users with the provided answers, in such situation, chatbot should be transferred to an agent in order to handle the rest of the conversation. A concise summary of the user utterances helps the agent to handle the rest of the chat without raising annoying delay. Because the dialog between chatbots and user is fragmented and informal when compared to classic summarization dataset, high complexity models often have difficulty in achieving ideal results. In this paper, we present an extractive chat summarization system to provide a concise summary of the discussed topics in chat. We conclude three groups of critical and universally applicable features dedicated to summarization tasks and use the features to extract keywords for ranking sentences’ importance in the chat. We evaluate supervised and unsupervised methods for keyword extraction and generate the summary by combining selected sentences with specific rules. We compare our obtained results with two state-of-art, deep learning based methods over a new chatlog dataset. Experimental results reveal our method to be more effective and efficient than currently popular, high complexity methods.

KEYWORDS
Chatbots, Summarization, Machine Learning.

INTRODUCTION

Chat summarization is a notable utility which is exploited to various domains like study, military, etc. Using of chatbots for handling conversations is rapidly increasing and companies are trying to exploit chatbots in order to provide a timely open channel to the users requirements. Chatbots can response significant volume of user's questions at any time. But sometimes chatbots cannot satisfy users with their answers, in such the case, the conversation must be transferred to a human agent to handle rest of the conversation. Normally, the agent must be aware about what was discussed between the user and chatbot right before conversation be transferred to the agent. One way for agent is to read all previous messages before going to handle rest of the conversation. This may take a considerable time which causes a long delay for the user. An concise and descriptive summary of the logged chat can help agent to understand what happened.
in the chat, very quickly. Furthermore, logging a summary of user and human agent in the next step also could be useful for future topic analyzing of user requirements.

BACKGROUND

Text summarization is a branch of nature language processing, which is mainly used to compress and summary the text in different scenes such as news streamlining, document indexing and information retrieval. Maintaining the Integrity of the Specifications

Chat summarization is an application of text summarization. Nowadays the chatbot are mainly used in the field of intelligent customer service, real-time information platform, question and answer service, etc. For questions raised by users, chatbots can automatically give corresponding answers and imitate the way human communicate. But when the existing chat and dialogue technologies cannot meet the needs of the users and cannot respond well to complex problems, the dialogue might handle by human. Human agent need to confirm the content of the chat between the user and the chatbot, which might take a long time can cause a certain delay. The chat summarization model can be used as a supplement to the dialogue system. It can automatically summarize the dialogue before it is handed over and extract the key information, which will definitely shorten the understanding time of the human agent and improve the efficiency of the system. Similarly, chat summarization can also be applied to meeting reviews, course summaries and other scenarios to help participants better obtain information.

For designing the summarization model, researchers often adopt the machine learning, deep learning and feature engineering methods. Jingjing Liu, Stephanie SE Neff, Victor Zue[1] used frequency, emotion, semantic unilateral features, SVM and decision tree to extract dialogue summary. Experiments prove that decision tree is better than SVM in chat summary task. Abigail see, Peter j Liu, Christopher d manning[2] proposed a Pointer-Generator network to solve the abstract summarization tasks. Ramesh nallapati, Feifei Zhao, Bowen Zhou[3] proposed a SummaRuNNer network model based on RNN to solve the problem of document summarization. Chih-Wen Goo, Yun-Nung Chen proposed a Sentence-Gated model based on dialogue behavior selection in 2018. Didi AI Labs[4] proposes a Leader-Writer network to solve the problem of summary generation based on auxiliary key sequence. All of these researches can make the generation of customer service conversation summary more complete, logical and correct. It can be noted that the research in recent years has mainly focused on the deep learning method and achieved good results. However, compared with machine learning methods and feature engineering methods, the deep learning method is more complex, which means that deep learning methods highly rely on the quality of the raw chat text and the quality of chat text in a real chat scenario might much lower than the dataset for deep learning. In this paper, we use feature engineering and supervised/unsupervised keyword extraction techniques to generate the dialogue summary. First of all, we use the method of feature engineering to extract 3 groups, 24 features to support the key word extraction process; Then the supervised/unsupervised keyword extraction method is used to input and extract the candidate keywords of the dialogue with corresponding features; According to the extracted results, sorting and classifying are carried out to generate summaries. We have also built a dialogue text dataset, which uses the real chat data of the enterprise dialogue system and has certain complexity.
and authenticity. Finally, we analyzed the experimental results and compared this method with the deep learning method.

**APPROACH**

**Algorithm Architecture**

This section is inspired by [5], in which authors generate proper features to extract keywords from the emails. In this section, we construct feature engineering corresponding to the structure of chats with only two participant. Our keyword extraction system consist of four parts.

Chat processing: In the first part, we distinguish chat's messages of two parties (user-agent/chatbot), all contiguous messages in one side before interrupted by another side are considered as a paragraph. We then sentence-segment each chat followed by tokenization. We ignore grating and regarding messages, which usually are first and last message, to only focus on the chat text.

Candidate extraction: For the candidate extraction, we only use noun phrases and named entities since they carry more meaning of the messages.

Pre-processing: In the pre-processing part, stopwords and numbers are removed, and extracted candidates are converted to lowercase.

Ranking/Classification: In the last part, we extract keywords from chats using two approaches - Unsupervised, and Supervised. In the unsupervised (rank) approach, we first rank words using several linguistic and centrality-based features, and then extract top-ranked words. In the supervised (classification) approach, we classify candidate words as keywords vs. non-keywords using the features, and select ones classified as keywords.

**Feature for Keyword Extraction**

To extract keywords from a chat, we use word features in supervised and unsupervised approach, in the unsupervised approach, each feature is used solely, and in the supervised approach we investigate several combinations on features to extract keywords.

Before we extract keywords from a document, we need to know about its structure. Chat's structure is different from other text documents, it is even differ from email. We generate several features according to a chat structure in which there are two sides of conversation, one side is a user querying about a particular requirement, and another side is a human/bot agent trying to satisfy user by corresponding answers.

**TOKEN FEATURE**

a) TF: Term frequency in a chat. It is a main feature to show importance of a word in a document.

b) TFIDF: Term Frequency-inverse Document, which is calculated with the Brown Corpus using NLTK framework.

c) Word length. Number of characters in a word. Longer words likely are more informative than shorter ones.

d) Is capitalized? Whether the word is capitalized. As well as named entity detection, it is a good indicator for specific words, e.g., model of a phone.

e) Entity Length. Length of the entity containing the word, in which entities are detected using Stanford NLP toolkit.
f) Is in Both sides? whether the word appears in both sides of chat or not. Words appearing in both sides of a conversation likely are parts of a discussing topic.

g) Is in Both sides with no gap. Whether the word appears in both sides with no gap e.i., no paragraph in between. Appearing words in contiguous paragraphs of a chat shows possible importance of them to be part of a discussing topic in the chat.

PARAGRAPHS/SENTENCES FEATURES

a) Difference of first and last Positions. Position difference of the first occurrence of a word and its last occurrence. Difference is measured by number of paragraphs appeared between two occurrences of a word. Words with less distance likely are in the center of a discussing topic in a chat.

b) Difference of first positions in both sides. Position difference of the first occurrence of a word in one side and its first occurrence in the other side. Close words discussing in the first paragraphs of a chat might be part of a general view of the main discussed topic.

c) First position. Position of the first occurrence of a word in a chat. Position is measured by counting words appeared between a word and beginning of the chat. Words in the first paragraphs of a chat might be part of an introduction for the main discussed topic.

d) Last position. Position of the last occurrence of a word in a chat. Position is measured by counting words appeared between a word and beginning of the chat. Words in the last paragraphs of a chat might be part of a conclusion for the main discussed topic.

e) Normalized first position. First position feature, normalized by the number of words in the chat.

f) Normalized last position. Last position feature, normalized by the number of words in the chat.

g) Chat length. Number of words in a chat. We only use this feature in supervised approach.

h) Paragraph length. Maximum length of the paragraph which contains the word. Longer paragraphs likely contain more information about discussing topics in the chat.

i) Is in the first Paragraph? It is a binary feature which is used in supervised approach.

j) Paragraph Frequency. Number of paragraphs containing the word.

k) Mean Par-length. Mean length of the paragraphs containing the word.

l) normalized position in First paragraph. Normalized position of the word in the first paragraph, normalized by the number of words in the paragraph.

m) Average position In Paragraph. Average position of the word in the paragraphs. Position is measured by number of words since the beginning of the paragraph divided by length of the paragraph.

GRAPH FEATURE

a) Degree. Number of edges incident to a node in a graph. All candidate words are used to form a graph. A word with higher degree is more important.

b) PageRank. It is generated by using PageRank algorithm[6].

c) Coreness. It is generated by calculating the coreness of the word. We denote the graph generated from the raw conversation as G and H is the subgraph of G. If all the nodes in H have at least k edges connect to the other
nodes. The $H$ is a k-core subgraph. We calculate all the subgraph containing the token and define the max k as the coreness of the token.

d) HITS score. It is generated by applying HITS algorithm to a co-occurrence matrix implemented in[7]. We exploit HITS algorithms in a directed graph to emphasize the words appearing in the beginning of a paragraph.

Summary Generation

The extracted and ranked keywords in both supervised and unsupervised frameworks are used to generate final summary of the chat. We generate the summary by extracting important sentences according to the top ranked keywords appearing in the sentences. Each sentence $s$ is ranked using following formula:

$$S(s) = \frac{1}{l} \sum_{w_i \in s} w_i$$

(1)

where $S(s)$ is score of the sentence, $l$ is its length, and $w_i$ is the keyword occurred in $s$.

EXPERIMENT

DataSet

One of the challenges of chat summarization is to find a suitable conversational dataset that can be used for evaluating a proposed approach. Most available conversational corpora do not contain any human written summaries, or the gold standard human written summaries are generic. Furthermore, most of them are multi-participant chatlogs, thus, for our evaluations, we generate our dataset from chats happened between a user and human agent after a chatbot cannot satisfy the user and a human agent should handle rest of the chat. For each chat log in our dataset, there is a concise summary of the main discussed topics in the chat which is generated by human. We use these summaries in our evaluation process with two purposes. First, to evaluate our keyword extraction systems, assuming most important keywords likely appear in a good summary. We use noun phrases in the summaries as human annotated keywords for evaluation. Moreover, we use these summaries to evaluate final extractive summarization system.

Evaluation Measure

For keyword extraction evaluating we consider Exat match agreement, i.e., when two phrases match exactly (up to lowercasing and spaces). For evaluating quality of summaries produced by our proposed method, we used the ROUGE [8] evaluation toolkit. It is a method based on N-gram statistics, found to be highly correlated with human evaluations. The ROUGE-N is based on n-grams and generates three scores Recall, Precision, and the usual F-measure for each evaluation.
\[ R_n = \frac{\sum_{S \subset \text{ref}} \sum_{n-\text{gram}s} \text{CountMatch}(n-\text{gram})}{\sum_{S \subset \text{ref}} \sum_{n-\text{gram}s} \text{Count}(n-\text{gram})} \]  

(2)

\[ P_n = \frac{\sum_{S \subset \text{cand}} \sum_{n-\text{gram}s} \text{CountClip}(n-\text{gram})}{\sum_{S \subset \text{cand}} \sum_{n-\text{gram}s} \text{Count}(n-\text{gram})} \]  

(3)

\[ F = \frac{2 \times P_n \times R_n}{P_n + R_n} \]  

(4)

\( R_n \) (recall) counts the number of overlapping n-gram pairs between the candidate summary to be evaluated and the reference summary created by humans for more details. \( P_n \) (precision) measures how well a candidate summary overlaps with multiple human summaries using n-gram co-occurrence statistics (See [9] for more details). We used two of the ROUGE metrics in the experimental results, ROUGE-1 (unigram) and ROUGE-2 (bigram).

**EXPERIMENTS RESULTS**

Our experiment is twofold, we first evaluate unsupervised and supervised methods in keyword extraction, and then we evaluate keyword-based extractive summarization system. Unsupervised methods rely on the individual use of the features described in Section 3.2, for supervised methods, we apply several Machine learning methods to different combinations of the features.

**Unsupervised Methods**

We first extract keywords in an unsupervised framework, and then generate an extractive summary for a chat. To extract keywords, we use one feature at a time to rank candidate words, and then top 50% of the ranked list are selected to be used in summary generation part. We also evaluate effectiveness of the binary features, but we do not use them individually to extract a summary. For binary features, we select all words with value 1. Table I shows the performance values for each feature.
**TABLE I. PERFORMANCE OF UNSUPERVISED KEYWORD EXTRACTION. BEST VALUES SHOWN IN BOLD.**

| Feature                                             | Precision | Recall  | F-Score  |
|-----------------------------------------------------|-----------|---------|----------|
| TF                                                  | 0.696     | 0.5050  | 0.292    |
| TF-IDF                                              | 0.455     | 0.5000  | 0.238    |
| Different of first and last position                | 0.455     | 0.5000  | 0.238    |
| Different of first position in both sides            | 0.543     | 0.5007  | 0.260    |
| First position                                      | 0.455     | 0.5000  | 0.238    |
| Last position                                       | 0.455     | 0.5000  | 0.238    |
| Normalized first position                           | 0.455     | 0.5000  | 0.238    |
| Normalized last position                            | 0.455     | 0.5000  | 0.238    |
| Word length                                         | 0.455     | 0.5000  | 0.238    |
| Chat length                                         | 0.455     | 0.5000  | 0.238    |
| Paragraph length                                    | 0.496     | 0.5000  | 0.246    |
| Paragraph frequency                                 | 0.749     | 0.5041  | 0.301    |
| Mean Par-length                                     | 0.504     | 0.5080  | 0.253    |
| Normalized position in first paragraph              | 0.455     | 0.5000  | 0.238    |
| Average position in paragraph                       | 0.455     | 0.5000  | 0.238    |
| Entity length                                       | 0.455     | 0.5000  | 0.238    |
| Degree                                              | 0.658     | 0.5057  | 0.285    |
| Pagerank                                            | 0.652     | 0.5040  | 0.284    |
| Coreness                                            | 0.455     | 0.5000  | 0.238    |
| NG-rank                                             | 0.455     | 0.5000  | 0.238    |

**Supervised Methods**

Similar to the unsupervised methods, keywords are extracted in a supervised framework, and then final summary is generated. We run six different classification algorithms on our dataset, including C4.5 decision tree, SVM, KNN, Naïve Bayes, Xgboost, and Catboost. For each algorithm, we examine different combinations of the features to find the best result. Some of the results are shown in Table II.

**TABLE II. PERFORMANCE OF SUPERVISED KEYWORD EXTRACTION. BEST VALUES SHOWN IN BOLD.**

| Machine learning algorithm | Precision | Recall  | F-Score  |
|----------------------------|-----------|---------|----------|
| C4.5                       | 0.9553    | 0.6279  | 0.7577   |
| SVM                        | 0.7714    | 0.6356  | 0.6969   |
| KNN                        | 0.7490    | 0.5859  | 0.6574   |
| Naïve bayes                | 0.5890    | 0.6325  | 0.6099   |
| Xgboost                    | 0.9553    | 0.5243  | 0.6770   |
| Catboost                   | 0.9553    | 0.5891  | 0.7287   |
Figure 1. Graph of performance of supervised keyword extraction.

- **C4.5**: Continuous features will be discretized according to intervals which results in more decision tree branches divided from the continuous features than discrete features. Therefore, the dataset which is dominated by continuous features is likely to over-fitting and often has a good performance on recall rate.

- **SVM**: Due to the poor spatial separability of continuous features and 0/1 discrete features, the training dataset of SVM is mainly composed of integer features. During the training process, the setting of iteration times and the panel will greatly affect the final training result. Excessive increasing iteration times and decreasing penal will both contribute to the over-fitting. At the same time, non-core SVM can only segment hyperplanes linearly, which means the improvement of recall rate is very difficult. Therefore, we use Gaussian kernel SVM to map the low-dimensional space to high-dimensional space, which will enhance the separability of feature space and improve the recall performance.

- **KNN**: Due to the poor spatial separability of 0/1 discrete features, the training dataset of KNN is mainly composed of integer features and continuous features. The KNN classification is very sensitive to the selection of K. Setting value K too low will introduce noisy sample to correctly classified set and setting value K too high will introduce samples far from the targets to the correctly classified set. Because KNN use Euclidean distance to classify categories, different features should be normalized before training which will greatly affect the spatial distribution of features. At the same time, the combination of multiple features also dilute and distort the original feature space. Therefore, single feature data set often outperform the multiple feature data set in precision rate and recall rate.

- **Naïve Bayes**: Due to the characteristic of Naïve Bayes, A feature with better spatial separability will perform better and the causal relationships between features is ignored, which means that the results of the best single feature can completely and effectively represent the classification quality of the features in training set. When increasing the combining feature number, the samples under the high-quality feature will be further divided and original classification will be diluted by relatively low-quality features, which will inevitably lead to a decrease in precision rate. On the other hand, combining more features will expand the search range of positive examples in the sample space, thus increasing the recall rate.

- **xgboost**: The xgboost algorithm will rank the features with their importance in the classification process. Therefore, we apply correlation analysis and divide the features into several group. As an bagging algorithm, xgboost rely on the number of features. When the number of features increases, both precision rate and recall rate will increase. At the same time, according to the characteristics of the
integration algorithm, disadvantage features will automatically reduce their importance in the learning process. Compared with the xgboost algorithm, catboost improves xgboost from 2 aspects. Catboost randomly combines the features in the training set which will modify the feature space and simplify the training steps. In addition, catboost transfer the 0/1 discrete features into intergrat features which will have a benefit on the branch dividing of Cart tree. Both algorithms acquire good precision, recall and F1 results.

| Machine learning algorithms | Precision   | Recall                        | F-Score   |
|-----------------------------|-------------|-------------------------------|-----------|
| C4.5                        | 1           | 23,21,1,8,9,17,18,16,0,15,20,23,4,5,6,7 | 1         |
| SVM                         | 1,6,16,21,22| 1,6,16 with RBF kernel        | 1,6,16,21,22 |
| KNN                         | 21(K=6)     | 3(K=6)                        | 21(K=6)   |
| Naïve bayes                 | 3           | 1,2,3,5,6,11,15,16,21,22      | 3         |
| xgboost                     | 22,23,24,16 | 1,4,11,12,14,19,22,26,24,10,16,20 | 22,23,1   |
| catboost                    | 1           | 1,4,11,12,14,22,23,24,16      | 1         |

From the Table III, Best we can conclude that two points.
- The improvement of recall rate is closely related to the number of features. Supervision algorithm relies on a large number of feature synthesis to obtain the optimal recall rate. In contrast, the quality of features is more important to the precision improvement. Feature1 Term Frequency, as the feature with the highest quality, usually has a better performance with a single feature in most of the supervised extraction algorithm.
- It is more difficult to improve the recall rate than to improve the precision rate, which is why the best performance combinations of F score are usually as same as the precision combinations.

**Extraction evaluation of unsupervised and unsupervised methods**

We use two of Rouge metrics in our comparison, Rouge-1(uni-gram) and Rouge-2(bi-gram). The obtained results are shown in Table 4 and Table 5. The results show that the supervised methods with Proper finetune will have an edge above the baseline which is the summarization generated by Pointer network[2] based on deep learning. Among them, C4.5 and KNN are the most outstanding algorithms, which further illustrates that for dialogues and conversation scene, algorithms with lower complexity can often achieve better results.

| Machine learning algorithms | Avg-R1 | Avg-P1 | Avg-F1   |
|-----------------------------|--------|--------|----------|
| C4.5                        | 0.4050 | 0.2373 | 0.2804   |
| SVM                         | 0.2991 | 0.1153 | 0.1550   |
| KNN                         | 0.3459 | 0.3183 | 0.3036   |
| Naïve bayes                 | 0.2848 | 0.1469 | 0.1801   |
| xgboost                     | 0.3329 | 0.2233 | 0.2468   |
| catboost                    | 0.3379 | 0.2266 | 0.2488   |
TABLE V. SUPERVISED METHODS SUMMARY VS. DEEP LEARNING SUMMARY IN ROUGE-2.

| Machine learning algorithms | Avg-R2 | Avg-P2 | Avg-F2 |
|----------------------------|--------|--------|--------|
| C4.5                       | 0.2472 | 0.1568 | 0.1818 |
| SVM                        | 0.1214 | 0.0578 | 0.0745 |
| KNN                        | 0.2426 | 0.2425 | 0.2250 |
| Naïve bayes                | 0.1391 | 0.0802 | 0.0963 |
| xgboost                    | 0.1997 | 0.1496 | 0.1606 |
| catboost                   | 0.2019 | 0.1515 | 0.1615 |

**CONCLUSION**

In this paper, we have extracted and generated artificial features about the dialogue and conversation with the data set. By analyzing each features' characteristic, we apply different feature combinations with fine tunes to improve the performance of supervised keyword extraction methods. We use Rouge metrics to compare the summary generated by our feature engineering with the reference summary and achieve a significant improvement in summary generation. Our experiment indicate that different algorithms have different optimization methods and the quality of features will have a very critical impact on the effect of learning process. In a complex dialogue scenario, for the algorithm which is sensitive to the distribution of feature space, integer features
and 0/1 discrete features have an advantage over continuous features. For algorithms which rely on the regression and bagging methods, the continuous features usually perform better. When optimizing a single algorithm based on the dialogue dataset, the precision rate requires higher quality of features, while the recall rate requires higher number of effective features. In addition, for the generation of summaries, algorithms with lower complexity often have better results the complex summary generation methods.

REFERENCES

1. Liu J, Seneff S, Zue V. Dialogue-oriented review summary generation for spoken dialogue recommendation systems[C]//Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2010: 64-72.
2. See A, Liu P J, Manning C D. Get to the point: Summarization with pointer-generator networks[J]. arXiv preprint arXiv:1704.04368, 2017.
3. Nallapati R, Zhai F, Zhou B. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents[C]//Thirty-First AAAI Conference on Artificial Intelligence. 2017.
4. Goo C W, Chen Y N. Abstractive dialogue summarization with sentence-gated modeling optimized by dialogue acts[C]//2018 IEEE Spoken Language Technology Workshop (SLT). IEEE, 2018: 735-742.
5. S. Lahiri, R. Mihalcea, and P.-H. Lai. Keyword extraction from emails. Natural Language Engineering, 23(2):295–317, 2017.
6. Mihalcea, Rada and Tarau, Paul. Textrank: Bringing order into text. Proceedings of the 2004 conference on empirical methods in natural language processing. Page: 404—411. 2004.
7. Pourvali, Mohsen and Orlando, Salvatore and Gharagozloo, Mehrad. Improving clustering quality by automatic text summarization. In AIRS, page 292–303. Springer, 2015.
8. C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out: Proceedings of the ACL-04 workshop, volume 8, 2004.
9. K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics, pages 311–318. Association for Computational Linguistics, 2002.