Sentiment Polarity Analysis for Generating Search Result Snippets based on Paragraph Vector

Yujiro Terazawa, Shun Shiramatsu, Tadachika Ozono and Toramatsu Shintani
Department of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology
Nagoya, Aichi, 466-8555, Japan
Email: {tyujiro, siramatsu, ozono, tora}@toralab.org

***Abstract*—Although sentiment polarity is important for understanding public reputations, conventional web search snippets do not include sentiment polarity. Therefore, we propose a system to generate search result snippets that considers sentiment polarity. We also propose a method for extracting reputation sentences from search result snippets based on a given search query. To extract such sentences, we use a Paragraph Vector (PV) to calculate the similarity of words and sentences. Overall, our method is based on this PV and logistic regression for analyzing sentiment polarities. We evaluated the accuracy of our method via the Rakuten dataset. In our evaluation, we focused on methods pertaining to what supports sentiment polarity analysis in our method. We found that considering both a search query and reputation is important in generating our modified snippets. Furthermore, we identified the need for a method to remove meaningless sentences, which will be part of our future studies.**

I. Introduction

A search engine is a fundamental part of the World Wide Web, providing web documents from queries requested by users. In general, a search engine outputs search result snippets; each snippet is a summary of the corresponding web document outputted by a search engine. A snippet comprises parts of a document that include a search query.

Reputation tends to include beneficial information for users; therefore, search engines should utilize reputations as part of search. In this study, we describe our implemented system to support users browsing based on reputations of relevant search results. Our system can generate snippets that consider sentiment polarity. Such snippets support users in browsing reputations written in documents in search results.

Sentiment polarity analysis is a technique to classify sentiment polarities (i.e., positive or negative) of words, sentences, or documents. Hence, sentiment analysis is a social demand of opinion analysis. In other words, sentiment analysis applies to aggregating and summarizing opinions. In Japan, Truetext[1], a text mining API, can analyze sentiment polarities, which enable companies to aggregate opinions of items or services by analyzing the reputations of companies. As noted above, sentiment polarity analysis can provide beneficial information to users.

II. Related Works

In this study, we use natural language processing via Deep Learning. In particular, we use the paragraph vector (PV) introduced by Le. Q. V. et al.[2]. PV is a result of an unsupervised learning algorithm and comprises two methods, i.e., “Distributed Memory Model of Paragraph Vector (PV-DM)” and “Distributed Bag of Words version of Paragraph Vector (PV-DBOW).” Both PV-DM and PV-DBOW represent words and sentences as vectors, which are called word vectors and sentence vectors, respectively. In particular, a word vector is learned by the conjunction of words and a PV is learned by the order of words in a sentence. Both methods represent a variable-length sentence as a fixed length vector. This vector can therefore be used in sentiment polarity analysis. More specifically, sentiment polarity analysis using PV has the advantage of using bag-of-words and n-gram models.

We can calculate similarities between two words or two sentences using vectors learned by PV; as alluded to above, PV is an extension model of the continuous bag-of-words and continuous skip-gram models introduced by Mikolov[3][4], each of which can learn word vectors. The skip-gram model can learn word vectors that capture a large number of precise syntactic and semantic word relationships. Therefore, we can calculate similarities between words using vectors that represent those words. In particular, we can calculate the cosine distance between two vectors. Because PV is an extension model of the continuous bag-of-words and continuous skip-gram models, we can calculate similarities between two words or two sentences using the cosine distance between two word vectors or paragraph vectors.

Sentiment polarity analysis has been a long-studied topic, with numerous applied methods proposed. For example, Pang et al.[5] show machine learning techniques for sentiment polarity analysis. Pang et al. analyzed movie reviews, with positive or negative ratings. Reviews were analyzed using naive bayes classification, maximum entropy classification, and support vector machine (SVM) approaches. They also described that sentiment polarity analysis can apply to recommendation systems and the aggregation of public reputations for companies.

Dave et al.[6] studied unique properties of this problem and developed a method to automatically distinguish between positive and negative reviews. They identified features that consider word occurrence rates, and then, scored these features. Their proposal included the application of naive bayes for scoring; they also proposed RIDF, an information retrieval technology, to extract features. They worked out by scoring they proposed.

Ku et al.[7] developed an algorithm that summarizes opinions of news articles and blogs by considering sentiment

[1]http://www.textmining.jp/

978-1-4799-9958-3/15 $31.00 © 2015 IEEE
DOI 10.1109/IIAI-AAI.2015.239

polarity. In particular, their algorithm scores words, sentences, or documents, and uses scores as part of a decision tree or SVM.

There are also microblog studies in sentiment polarity analysis. For example, O'Connor et al. [8] analyzed sentiment polarities in public reputations posted on the web service Twitter[2]. In particular, they scored a day t when user posts included "obama", "mccain", etc. They counted such posts, which included positive or negative words, and scored particular day t based on these counts. In addition, they scored each month to consider scores of days t, $t-1$, $\cdots$, $t-j$. They classified public reputations for topics such as "obama" and "mccain".

Studies have also focused on the use of metadata in web services. For example, Wang et al.[9] used metadata in the form of Twitter hashtags for sentiment polarity analysis. A hashtag is a word or phrase prefixed with "#" included in Twitter (and other social media sites). Hashtags are used to denote that a post is part of a particular topic. Wang et al. proposed an algorithm to analyze sentiment polarities in hashtags and generate a graphical model by the occurrences of hashtags in posts.

Kucuktunc et al.[10] analyzed sentiment polarities in the web service Yahoo Answers[3], which is a thread flow bulletin board that users post questions and answers to. Yahoo Answers includes metadata, such as personal data (e.g., location, sex, etc.), topics of questions posted, and so on. Kucuktunc et al. scored each set of questions and answers to analyze sentiment polarities in a post. In particular, they scored posts using a classifier for sentiment polarity analysis and calculated the average of scores of question posts, average of scores of particular topics, average of scores of what users who had particular features posted, and so on.

In this study, we generate snippets that are generated by considering sentiment polarity by analyzing sentiment polarities in web documents of search results. Generated snippets comprise positive and negative sentences. Therefore, we must first analyze these sentences. In sentiment polarity analysis for sentences, we need to consider the order of words, but previous work that does not use the PV tends to use only words and does not consider the order of such words. In addition, we generate snippets that coincide with the given search query in terms of reputation, which is primarily based on similarities of words or sentences. The PV approach can be set up to identify paragraph vectors that consider the order of words; we can then calculate similarities of words or sentences using the vectors learned via PV. Therefore, we generate snippets that consider sentiment polarity using PV.

III. Sentiment Polarity Analysis by Using Paragraph Vector

In this study, we propose the use of PV for sentiment polarity analysis. We use paragraph vectors by applying multiple logistic regression analysis (MLRA). In this method, we analyze the sentiment polarity of sentences in web documents.

[2]https://twitter.com/

[3]https://answers.yahoo.com/

A. Dataset

For this study, we used the Rakuten dataset[4], which is a Japanese corpus offered by Rakuten Co., Ltd. The dataset comprises data regarding web services offered by Rakuten Co., Ltd. In particular, this dataset contains goods data, as well as review data of goods in Rakuten Ichiba, which is an online shopping website, data of travel in Rakuten travel, which offers travel information, reports of recipes in the Rakuten recipe, which offers recipes posted by users of Rakuten recipe, information regarding trade and review in Rakuten auction, which is an online auction website, and the Tsukuba corpus, which is a review of annotated polarity labels in Rakuten travel. Polarity labels in the Tsukuba corpus are further detailed in Section III-D below. In this study, we extract sentences in Rakuten corpus to split by point.

B. Splitting Sentences in Sequences of Words

To identify paragraph vectors in Japanese, we need to split sentences into words. To do so, we used morphological analysis and named entity recognition before learning paragraph vectors, because sequences of words are also used in PV-DM and PV-DBOW and Japanese words aren't separated in a sentence. We also need named entities to use these words. In this study, we used CaboCha[5], which is a Japanese dependency parser for morphological analysis that can also perform named entity recognition. Note that CaboCha uses MeCab[6] for morphological analysis. In CaboCha, named entities are classified by definitions given by IREX. Given input sentence s_i to CaboCha, we split s_i in sequence of words $w_{i,1}$, $w_{i,2}$, $\cdots$, $w_{i,m}$ (from CaboCha's output). Here, $w_{i,j}$ is a morpheme or named entity. If CaboCha finds a named entity, we can extract named entity classified as organization, person, or location. Otherwise, we split this sentence into morphemes. For example, the Japanese sentence "田中角栄は政治家である．", which means "Kakuei Tanaka is politician", we split this sentence into the following sequence of words: "田中角栄", "は", "政治", "家", "で", "ある", and "。". In this sequence, "田中角栄" is the named entity, because CaboCha recognized the named entity and classified it as a person; the others are morphemes because CaboCha did not recognize them as named entities.

C. Learning Paragraph Vectors

We use PV-DM and PV-DBOW to learn paragraph vectors. PV-DM and PV-DBOW can learn by sentences s_1, s_2, $\cdots$, s_n and sequences of words w_{11}, w_{12}, $\cdots$, w_{1n}, w_{21}, w_{22}, $\cdots$, w_{nm}. In PV-DM and PV-DBOW learning, we need sentences of words w_{11}, w_{12}, $\cdots$, w_{1m}, by method presented in Section III-B. PV-DM and PV-DBOW can learn paragraph vectors by sentences s_1, s_2, $\cdots$, s_n and words w_{11}, w_{12}, $\cdots$, w_{1n}, w_{21}, w_{22}, $\cdots$, w_{nm}. We use paragraph vectors in sentiment polarity analysis. More specifically, in learning, , we learn paragraph vectors $v^D_{s_1}$, $v^D_{s_2}$, $\cdots$, $v^D_{s_n}$ via PV-DM, and $v^B_{s_1}$, $v^B_{s_2}$, $\cdots$, $v^B_{s_n}$ via PV-DBOW. These vectors are concatenated as $v_i = [v^D_i v^B_i] (i = 1, 2, \cdots, n)$ and are used in sentiment

[4]http://rit.rakuten.co.jp/opendata.html

[5]http://taku910.github.io/cabocha/

[6]http://mecab.googlecode.com/svn/trunk/mecab/doc/index.html

polarity analysis as input. In our study, we generated the PV-DM and PV-DBOW models learned based on the Rakuten dataset introduced in Section III-A above.

D. Analyzing Sentiment Polarities of Sentences

We used MLRA and paragraph vectors to analyze sentiment polarity of sentences. In MLRA model learning, we learn paragraph vectors using the Tsukuba corpus noted in Section **??**, and therefore, we used these vectors for MLRA model learning. We also used polarity labels from the Tsukuba corpus.

We learned the PV-DM and PV-DBOW models using the Rakuten dataset; furthermore, we learned the MLRA model using vectors that represent sentences in the Tsukuba corpus. In learning, we learned the PV-DM and PV-DBOW models using the Tsukuba corpus and another corpus from the Rakuten dataset, because we needed many sentences to learn paragraph vectors. In other words, using only the Tsukuba corpus was inadequate.

After we learn vectors v_{s_1}, v_{s_2}, $\cdots$, v_{s_n} by the methods described in Section III-C, we learned the MLRA model. In this study, we define polarity labels in MLRA as -1, 0, and 1. By definition, polarity label 1 means positive, polarity label -1 means negative, and polarity label 0 means neutral. Sentences in the Tsukuba corpus have polarity labels, also known as evaluation labels. We used sentences with polarity labels "praise", "complaint", "request", and "neutral". Replacing evaluation labels with polarity labels, we define "praise" as 1, "request" and "complaint" as -1, and "neutral" as 0.

IV. Generating Search Result Snippet Taken Account of Sentiment Polarity

In this study, we generated snippets by considering sentiment polarity. We used sentiment polarity analysis to generate snippets. In addition, we used a scoring function based on the search query and determined whether sentences in documents support users browsing opinions when we generate snippets that take sentiment polarity into consideration. In short, we used sentiment polarity analysis and word vectors.

First, we analyzed sentiment polarities of sentences in documents retrieved by the search engine. In analyzing sentiment polarities, we use the MLRA model. After analysis, we scored positive and negative sentiment polarities via Eq.1.

$$score(s) = \lambda \max_{u \in U} sim(v_s, v_u) + (1-\lambda) \max_{w,q} sim(v_w, v_q) \quad (1)$$

Here, λ is a hyperparameter with the range of $0 \leq \lambda \leq 1$, v_p is a vector that represents word or sentence p, s is the sentence score, u is a sentence, which is marked down opinion, and U is a set of u, which needs to be equipped in a system. we use Tsukuba corpus as U. In Eq.1, w is a word in a sentence s and q is a search query input by a user. Search query q needs to be split into sentences via the method described in Section III-B. The function sim is the cosine distance function. The more similar any sentence u in U is with sentence s, the higher the $score(s)$ value. Furthermore, the more similar any word w, which is in s, is with search query q, the higher the $score(s)$ value. We generated snippets from sentences with high scores; hence, such generated snippets consider both the web search query and whether sentences in documents can support users to browse opinions.

1: **Input:** $s_1, s_2, \cdots, s_n$ as $\mathbf{s}$
2: Learn $v_{s_1}, v_{s_2}, \cdots, v_{s_n}$ from $\mathbf{s}$
3: Predict $l_{s_1}, l_{s_2}, \cdots, l_{s_n}$ from $v_{s_1}, v_{s_2}, \cdots, v_{s_n}$
4: $\mathbf{s} \leftarrow DescendingSort(\mathbf{s}, Eq.1)$
5: $\mathbf{s}^{pos}, \mathbf{s}^{neg} \leftarrow filter(\mathbf{s}, l_s = 1), filter(\mathbf{s}, l_s = -1)$
6: **if** $\mathbf{s}^{neg} == \emptyset$ **then**
7: $\quad snippets \leftarrow s_1^{pos}, s_2^{pos}, s_3^{pos}$
8: **else if** $\mathbf{s}^{pos} == \emptyset$ **then**
9: $\quad snippets \leftarrow s_1^{neg}, s_2^{neg}, s_3^{neg}$
10: **else**
11: $\quad snippets \leftarrow s_1^{pos}, s_1^{neg}, MaxByScore(s_2^{pos}, s_2^{neg})$
12: **end if**
13: **Output:** $snippets$

Fig. 1. Snippet Generation Algorithm.

Fig.1 shows how snippets are generated, considering sentiment polarity. We can generate snippets this way by inputting sentences s_1, s_2, $\cdots$, s_n from a document. In Fig.1 $\mathbf{s}$ represents s_1, s_2, $\cdots$, s_n. In lines 2 and 3, paragraph vectors $v_{s_1}, v_{s_2}, \cdots, v_{s_n}$ are learned by $\mathbf{s}$ and labels $l_{s_1}, l_{s_2}, \cdots, l_{s_n}$ are predicted by paragraph vectors $v_{s_1}, v_{s_2}, \cdots, v_{s_n}$. In line 4, $\mathbf{s}$ is sorted in descending order by $score(s)$. In line 5, we extract sentences whose label l is 1 (i.e., positive sentences) and sentences whose label l is -1 (i.e., negative sentences). We define $\mathbf{s}^{pos}$ as set of positive sentences, and $\mathbf{s}^{neg}$ as set of negative sentences. In line 6-12, if $\mathbf{s}^{pos}$ or $\mathbf{s}^{neg}$ don't exist, we generate snippets using the top three sentences in the other set. If both $\mathbf{s}^{pos}$ and $\mathbf{s}^{neg}$ exist, we generate snippets using the top sentences of $\mathbf{s}^{pos}$ and $\mathbf{s}^{neg}$, plus either the second sentence $\mathbf{s}^{pos}$ and $\mathbf{s}^{neg}$, which is chosen based on which score is higher. In line 13, the $MaxByScore$ function outputs a sentence for which the score is the higher of the two input sentences.

In this study, we also generated sentiment polarities of web documents. We used sentences of a web document to generate such snippets. In particular, we used positive and negative sentences based on line 5 of the algorithm shown in Fig.1. Sentiment polarities of web documents are generated by these sentences. We define sentiment polarities of web documents as the proportion of a number of positive sentences to a number of positive and negative sentences, and a proportion of a number of negative sentences to a number of positive and negative sentences. These proportions can abstractly show sentiment polarities in web documents.

V. System Architecture

Fig.2 shows our system architecture. The input to our system is a search query, and the output is sequences of snippets that consider sentiment polarity. More specifically, our system retrieves search results from the Google search engine based on a search query given by the user, and then, outputs search result snippets. Our system generates snippets with sentiment polarity taken into account via the following five steps: (1) retrieve search results from Google based on the given search query; (2) scrape web documents in the search results; (3) analyze sentiment polarities of sentences in these

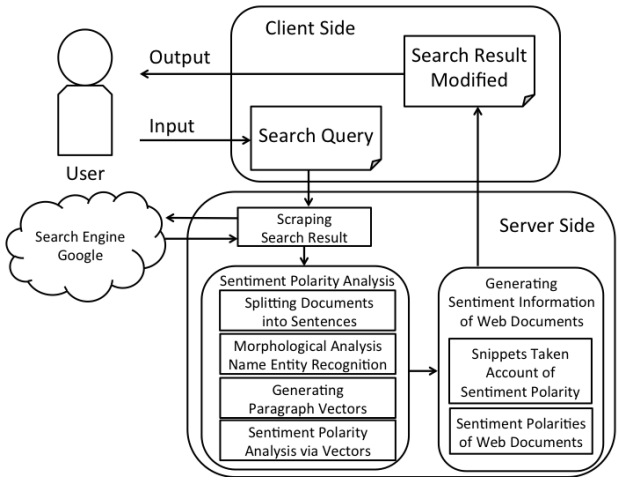

Fig. 2. System architecture.

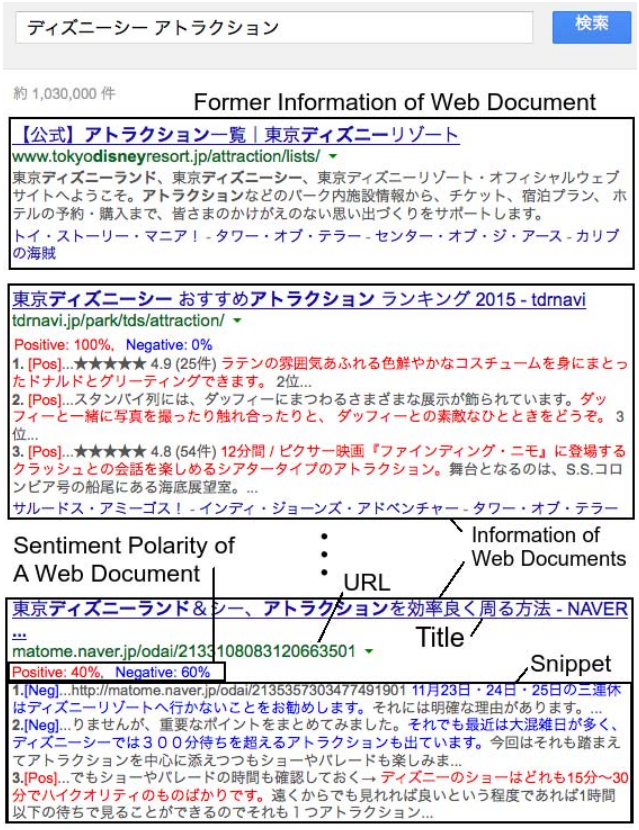

Fig. 3. Example of output snippets taken account of sentiment polarity.

web documents; (4) modify the search results by applying snippets that consider sentiment polarity; and (5) show the modified search results.

Our system receives a search query that a user inputs from the client side. In the "Scraping Search Result" process shown in Fig.2, our system submits a search query to Google, and after it receives search results, it scrapes web documents in the results to analyze HTML source code. After this analysis, our system scrapes web documents to access URLs of other web documents.

In the "Sentiment Polarity Analysis" process, our system analyzes sentiment polarities in the web documents scraped by our system. Our system performs such analysis via the following four processes:(1) Splitting documents into sentences, (2) perform morphological analysis and name entity recognition; (3) Generating paragraph vectors in web documents scraped by our system, and (4) perform sentiment polarity analysis via paragraph vectors. More specifically, our system learns paragraph vectors from web documents and analyzes sentiment polarities of sentences via such paragraph vectors. In morphological analysis and name entity recognition, our system implements the method describe in Section III-B; similarly, in learning paragraph vectors our system performs the method described in Section III-C. In learning paragraph vectors, our system also uses the PV-DM and PV-DBOW models. In sentiment polarity analysis, our system uses a MLRA model, which learns via the method quoted in Section III-D.

In the "Generating Sentiment Information of Web Documents" process in the figure, our system generates snippets that consider sentiment polarity and sentiment polarities of web documents using the results of sentiment polarity analysis and sentences in web documents. In particular, our system implements the method described in Section IV. Next, our system modifies the search results by modifying the snippets of the given search results with the snippets that our system generates.

Finally, our system then shows the modified search results in the client interface. In Fig.2, the server side processes are "Scraping Search Result", "Sentiment Polarity Analyze", and "Generating Sentiment Information of Web Documents"; the client inputs a search query into our server and receives the modified search results.

As noted above, we show the example output of our system in Fig.3. In the figure, when a user inputs search query "ディズニーシー　アトラクション", which is a Japanese sentence meaning "Disneysea Attraction", the output is the modified search results. The dots in the figure indicate that part of the search results has been omitted. As shown in the figure, our system presents an edit box and a "検索" button, which is Japanese for "Search, " in the upper portion of the screen, while search results are presented in the lower portion.

Users can enter a search query into the edit box and click the "Search" button. In the modified information pertaining to a web document in the search results, our system appends a sentiment polarity under the URL and modifies the original snippets by replacing such snippets with those generated by our system. If our system does not modify the web document information, it shows an icon that indicates that no modifications have been made to the snippets, whereas our system shows an icon indicating those snippets that have been modified by considering sentiment polarity and corresponding sentiment polarities of web documents.

In the figure, the last search result shown has been modified by our system. As indicated in the figure, the "Sentiment Polarity of A Web Document" is indicated; more specifically,

we show the respective proportions of sentiment polarities of the web document. In the example given, the sentiment polarities of the web document are "Positive: 40 %" in red and "Negative: 60 %" in blue. Furthermore, we show a snippet that considers sentiment polarity (indicated by "Snippet" in the figure). This snippet is generated by the method described in Section IV. In this snippet, positive sentences are displayed using a red font, whereas negative sentences are displayed in a blue font. Positive and negative sentences in snippets are the snippets of web documents that are written before positive and after negative sentences in such web documents. Finally, the "Former information of web document" denoted in the figure shows information of a web document that only had sentences predicted as being neutral by the method described in Section **??**. We therefore did not modify this snippet; if a snippet is not modified, our system shows an icon indicating to modify a snippet, until our system finished sentiment polarity analysis.

VI. Evaluation

In this study, we analyzed sentiment polarities of sentences. We therefore need to evaluate the precision of such sentiment polarity analysis in sentences. We evaluated MLRA models using paragraph vectors. In particular, we evaluated precision MLRA models to consider the number of sentences to use in learning, methods to learn paragraph vectors, and the penalty term in L1-norm regularization. We also used the Rakuten dataset noted in Section III-D as the corpus for our evaluation. Furthermore, we used an approximate method of PV, namely: hierarchical softmax (HS) and negative sampling (NG). We also used 4,309 sentences in the Tsukuba corpus, as well as other sentences in Rakuten dataset.

For our evaluation, we considered the following three models: (A) 1,265,255 sentences and HS; (B) 1,265,255 sentences and NG; and (C) 2,060,255 sentences and HS. We also learned paragraph vectors with lengths of 400 in our evaluation. In MLRA, we evaluated via the bootstrap method and divided the dataset into training and test datasets with a 7:1 ratio. We also used L1-norm regularization in MLRA; the MLRA equation that applies L1-norm regularization is

$$P(l_j|\mathbf{x}) = \frac{\exp(\mathbf{x}\mathbf{w_j})}{\sum_{i=0}^{k}\exp(\mathbf{x}\mathbf{w_i})} + \alpha\sum_{i=0}^{m} w_{ji} \qquad (2)$$

Here, l_j is a polarity label, with k polarity labels $l_0, l_1, \cdots, l_k$, $\mathbf{x}$ is a feature represented as m-length vector $\mathbf{x} = [x_1, x_2, \cdots, x_m]$. $P(l_j|\mathbf{x})$ is a probability that l_j is input as a feature in $\mathbf{x}$, w_j is a weight in MLRA that represents $\mathbf{w_j} = [w_{j1}, w_{j2}, \cdots, w_{jm}]$ and α is a hyperparameter. In our evaluation, we adapted hyperparameter a from 0.5 to 6.0 at 0.5 increments. We used an iMac computer with an Intel Corei3 CPU and 8GB memory running OS X 10.10.1; runtime environments were Python 2.7.6 and Cython 0.21.2. We also use the scikit-learn module of Python.

We show a result of evaluation in Fig.4, wherein the precision of MLRA is on the y-axis and α is on the x-axis. We show three graphs labeled A, B, and C, which represent results of evaluation models A, B, and C, respectively. In the figure, there is no contrast in precision in adapting parameter α. The largest difference in precision is approximately 2% in evaluation models A and C. In addition, because the precision of model A is also higher than that of model B, it would appear that a model using HS is more useful than a model using NG. Moreover, the more sentences used in a model, the higher the precision.The actual difference between model A and model C in terms of precision is approximately 5%. Therefore, it would appear that precision can increase as we add more sentences.

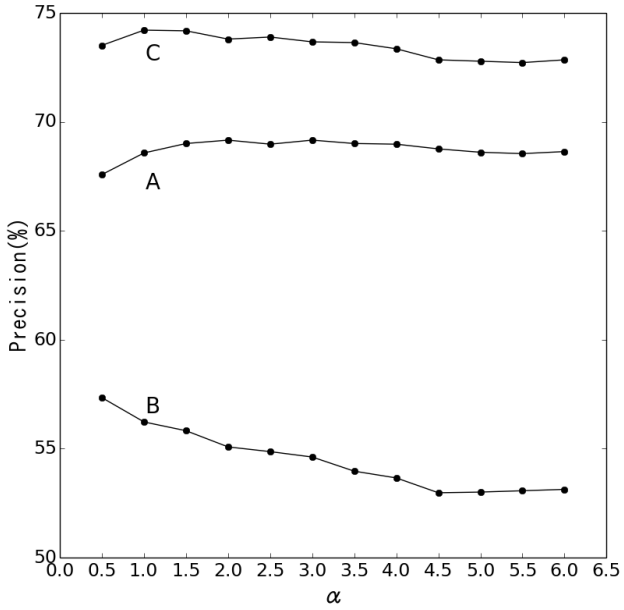

Fig. 4. Evaluation result.

VII. Discussion

In addition to the conclusions presented above, we also found that we should consider memory usage in our evaluation. In terms of memory usage, models using NG use more memory than those using HS. Because of this, we could not evaluate the NG model with 2,060,255 sentences, because the model could not allocate a large-enough memory region. In addition, the precision of model A was also higher than that of model B; considering precision and memory usage, it would appear that using HS is more useful than using NG. We therefore used HS in our system.

In generating snippets that consider sentiment polarity, we found it necessary to consider more than just the search query in sentences. In TABLEI and TABLEII, we show snippets that consider sentiment polarity with adapted hyperparameter λ from 1. In the tables, we show the top three positive sentences scored via this equation; TABLEI shows snippets in Japanese, whereas TABLEII shows the same snippets in English. TABLEI describes the positive sentences our system predicted in the same document; note that the table describes the top three positive sentences scored in 1 with hyperparameter λ set to 0.0 and 0.5. In scoring, the search query was "新穂高温泉", which is the Japanese word means "Shinhodaka hot spring". In both tables, the sentences scored an λ of 0.0 include the word "温泉" is included in the search query and results is only considered by the query. these sentences also includes

λ	Snippets (Japanese)
0.0	温泉ガイドのご利用方法
	30 軒ありました
	６つのお風呂全てが貸切で温泉三昧
0.5	２人で、無料の貸切露天♪飛騨牛付き♪のんびりカップルプラン
	またイワナの骨酒もオススメ！
	焼岳をバックに、穂高、槍ヶ岳などを一望できる最高のロケーション

TABLE I. EXAMPLE OF SNIPPET GENERATED BY OUR SYSTEM (JAPANESE).

λ	Snippets (English)
0.0	How to use guide of hot spring
	There were 30 eaves.
	All six hot spring are chartered and you can give all one's time to.
0.5	The leisurely plan for couple to entertain free hot spring chartered, with Hida beef!
	I also recommend alcohols of bones of trouts!
	There is best location to view Hodaka and Yarigaoka behind Yakeoka.

TABLE II. EXAMPLE OF SNIPPET GENERATED BY OUR SYSTEM (ENGLISH).

not positive and not beneficial because of classifier error. Conversely, the sentences scored with λ set to 0.5 were positive and had beneficial information although these sentences did not include the original search query terms. Therefore, we may need to consider not only the search query terms in sentences used to generate snippets that consider sentiment polarity.

In TABLEI and TABLEII, there is also unbeneficial sentence "30軒ありました", which means "There were 30 eaves," because of a classifier error. Therefore, we need to get rid of such sentences. In future studies, we plan to eliminate such sentences using paragraph vectors via an SVM. We found long Euclidean distances between paragraph vectors of meaningful sentences and paragraph vectors of meaningless sentences. Therefore, we will use this approach for predicting which sentences should be eliminated.

In future studies, we also need to study whether our generated snippets that consider sentiment polarity are useful to support to browse reputation. Therefore, we will evaluate our snippets by comparing them to their original form. In particular, we plan to equip Google search results and our modified search results to adapt λ in Eq.1. Examinees evaluate the search results we equip whether they are useful to browse reputation. Finally, we consider each about evaluation in former search results and search results in our system.

In our future studies, we need to study how our snippets lead to estimate the helpfulness of web documents. In [1], Aula et al. found thumbnails of web documents alone make users significantly underestimate the helpfulness of the web documents; conversely, textual snippets of web documents alone led to the overestimation of their helpfulness in web document previews. If our snippets adequately lead to estimates, our snippets will be shown to be beneficial to our users. We therefore plan to compare our snippets and former snippets or thumbnails of web documents, making improvements to our system based on such comparisons.

VIII. CONCLUSION

In this study, we proposed an algorithm for generating search result snippets that consider sentiment polarity. We implemented this proposed algorithm by showing modified search result snippets based on search results obtained from the Google search engine. We noted that reputation tends to include important information; therefore, we considered support for browsing opinions in a search engine. We also used paragraph vectors, because we found it necessary to analyze the sentiment polarity of sentences and calculate similarities between two words or two sentences to generate our search result snippets. In future studies, we plan to improve our system by identifying and eliminating search result snippets that comprise meaningless sentences using an SVM. We also plan to evaluate the effectiveness of our search result snippets for search engine users.

REFERENCES

[1] Aula, A., Khan, R. M., Guan, Z., Fontes, P., & Hong, P. (2010). A comparison of visual and textual page previews in judging the helpfulness of web pages. In Proceedings of the 19th international conference on World wide web. pp. 51-60.

[2] Le, Q. V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. In Proceedings of The 31st International Conference on Machine Learning, pp. 1188-1196.

[3] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In Proceedings of Workshop at International Conference on Learning Representations.

[4] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pp. 3111-3119.

[5] Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. pp. 79–86. Association for Computational Linguistics.

[6] Dave, K., Lawrence, S., & Pennock, D. M. (2003). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings of the 12th international conference on World Wide Web. pp. 519-528.

[7] Ku, L. W., Liang, Y. T., & Chen, H. H. (2006). Opinion Extraction, Summarization and Tracking in News and Blog Corpora. In AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs, Vol. 100107.

[8] O'Connor, B., Balasubramanyan, R., Routledge, B. R., & Smith, N. A. (2010). From tweets to polls: Linking text sentiment to public opinion time series. In Proceedings of the international AAAI Conference on Weblogs and Social Media, pp. 122-129.

[9] Wang, X., Wei, F., Liu, X., Zhou, M., & Zhang, M. (2011). Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management. pp. 1031-1040.

[10] Kucuktunc, O., Cambazoglu, B. B., Weber, I., & Ferhatosmanoglu, H. (2012). A large-scale sentiment analysis for Yahoo! answers. In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining. pp. 633-642.