Preliminary Performance Assessment on Ask4Summary's Reading Methods for Summary Generation

Abstract. Ask4Summary creates summary for students' questions based on text-based learning materials. This study conducts a preliminary assessment on Ask4Summary's performance in terms of generating summaries with different subsets of course materials (e.g., supplement academic papers in PDF only, notes and slides in Word and PowerPoint only, and everything the teacher provides for the students) read and processed by two reading methods: the built-in algorithm based on Python NLTK and AWS Comprehend Keyphrase Extraction and Syntax Analysis. The course materials of a graduate level Academic Writing in English course in an Asian university and twenty-six common questions that students may ask in the class are provided by the course instructor. Each of the questions are read via the two methods and Ask4Summary generates the summaries with the six different datasets created by: (1) Python NLTK reading the academic papers in PDF only; (2) Python NLTK reading notes and slides in Word and PowerPoint format only; (3) Python NLTK reading every course materials; (4) AWS Comprehend reading academic papers in PDF only; (5) AWS Comprehend reading notes and slides in Word and PowerPoint format only; and (6) AWS Comprehend reading every course materials. For the 312 queries (i.e., ask 26 questions in 6 datasets with 2 methods analyzing the questions) made, 117 queries successfully generated the summary, where only 2 of them were read by AWS Comprehend. Among the rest of 115 summaries, 67 of them are from the datasets created via the built-in algorithm and 48 are from the datasets created by AWS Comprehend.

Keywords: Language Learning, NLTK, AWS, Natural Language Processing, Learning Materials

1 Introduction

Information overload is caused by the highly increased education resources and makes users in the online learning environment spend huge time in searching for the suitable

education resources [6]. Helping students retrieve important information from the education resources becomes an important research area in the educational technology domain, such as multi-document, user-specific, and innovative text summarization applications [2]. The research team has developed the Ask4Summary [4][5] to automatically responds students' questions by generating summaries from the course content to help students quickly retrieve the important information, especially when online learning is getting more popular after the pandemic.

To enhance the service, we would like to know which Natural Language Processing (NLP) toolkit and types of course content can generate the summaries successfully. Therefore, the research team worked with the teacher in the Academic Writing in English course in an Asian university to evaluate the system. Section 2 briefly introduces the NLP toolkits used in the service and the system workflow of Ask4Summary. The evaluation method is sketched in Section 3. Section 4 explains the analysis results based on the collected data. A brief summary and the future works are concluded in Section 5.

2 Ask4Summary

The research team created an Ask4Summary website[1] for a graduate level Academic Writing in English course in an Asian university to evaluate the performance of summary generation. Ask4Summary first reads and processes (see Stage i in Fig. 1) the text-based materials that the instructor used in the course. The materials include supplement academic papers in PDF and notes and slides in Word and PowerPoint.

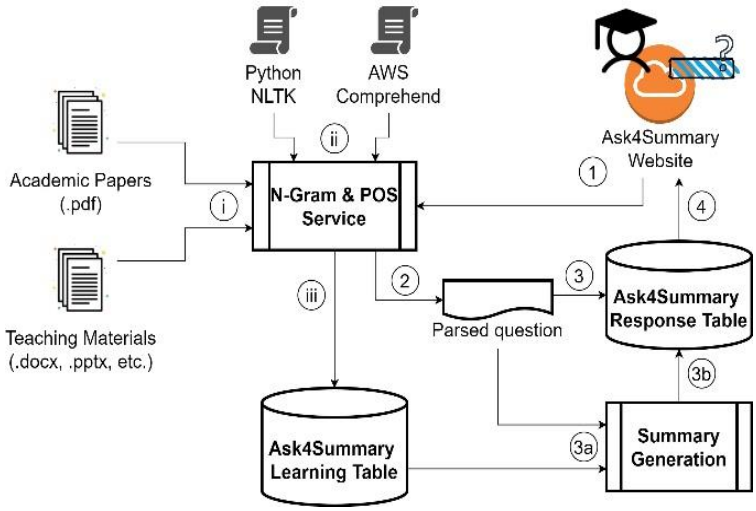

Fig. 1. The system workflow of Ask4Summary.

[1] (removed for blind review)

Ask4Summary identifies the valid N-grams according to their Part-of-Speech (PoS) tags [3]. The research team selects two widely used NLP toolkit/service to support the built-in algorithm, which are Python NLTK [1] and AWS Comprehend Keyphrase Extraction and Syntax Analysis[2] (see Stage ii in Fig. 1). With the NLTK and AWS Comprehend's help, Ask4Summary stores the extracted N-grams in the Ask4Summary Learning Table with the original content (see Stage iii in Fig. 1). The toolkit used in the feature extraction is also recorded in the Ask4Summary Learning Table so teachers or students are able to select which dataset generated by different toolkits they would like to use to generate the summary.

After the text-based learning materials are read and processed, teachers and students can use Ask4Summary (see Stage 1 in Fig. 1) on the website. They can enter their course related question as Fig. 2 shows. The users can choose which "brain" they want Ask4Summary to use for generating summaries. The "brains" are different Ask4Summary Learning Tables created earlier with either NLTK or AWS Comprehend method reading different subsets of learning materials: the academic paper in PDF format only, notes and slides in Word and PowerPoint format only, and all course materials.

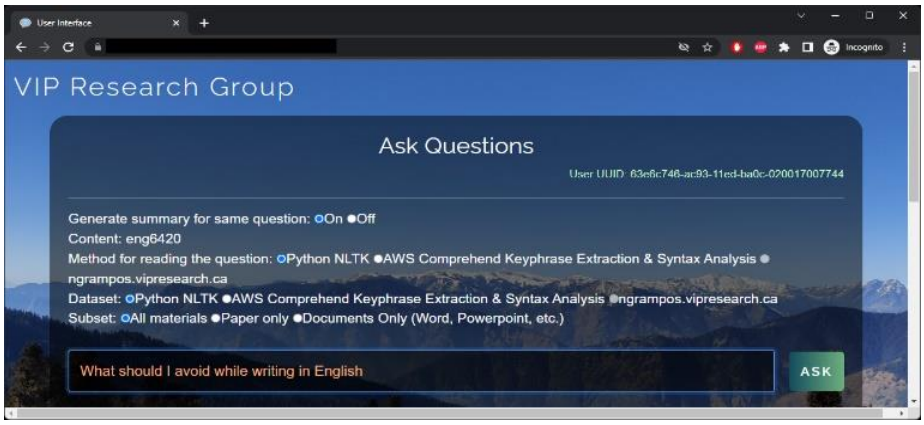

Fig. 2. Users can ask questions related to the Academic Writing in English course on the Ask4Summary website.

There is another table – Ask4Summary Response Table – used to store the questions users asked before as well as the questions' N-grams & PoS tags and the generated summaries. After a question's valid N-grams have been extracted (see Stage 2 in Fig. 1), Ask4Summary first checks whether or not similar question in Ask4Summary Response Table has been asked before (see Stage 3 in Fig. 1). If similar question exists, Ask4Summary simply retrieves the past generated summaries and delivers to the users (see Stage 4 in Fig. 1). On the other hand, it uses Cosine Similarity to find the top Y sentences in the top X documents, where X and Y are pre-defined and can be adjusted, are related to the question for generating the summaries (see Stage 3a in Fig. 1).

[2] https://aws.amazon.com/comprehend/features/

The generated summary will be saved in the Ask4Summary Response Table (see Stage 3b) and dispatches the summary to the user (see Stage 4). Fig. 3 shows the summary generated by the system with the reading method Python NLTK regarding the question "What should I avoid while writing in English." The Ask4Summary first uses Python NLTK to analyze the question entered. According to the selection of the "brain" types in Fig. 2 – that is, using Python NLTK to read all course material and storing the data into the Ask4Summary Learning Table, the generated summary is: "1) Avoid informal verbs. 2) Avoid colloquial expressions. Avoid informal or colloquial expressions such as a lot of in your written assignments. 3) Avoid clichés. Avoid using the expressions below in your written work." Users can give feedback regarding their perceived relevance (scaled from 0 to 10) and perceived satisfaction (scaled from 0 to 10) toward the generated summary.

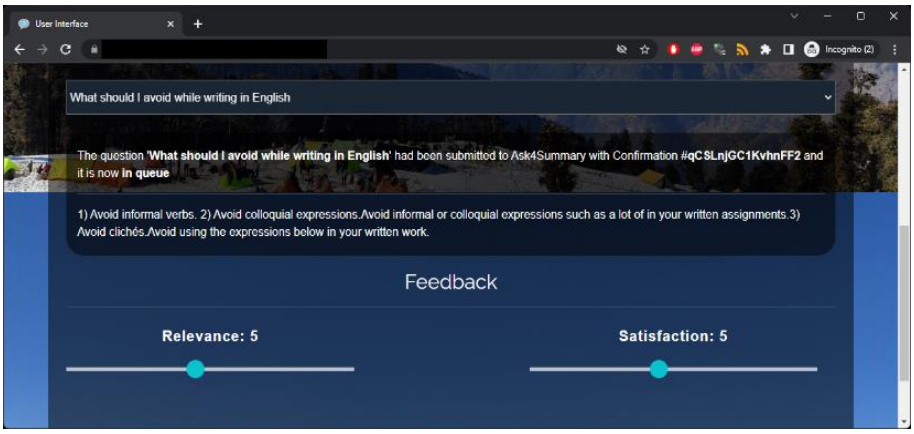

Fig. 3. The summary of the question generated by the Ask4Summary website.

3 Evaluation Design

The Academic Writing in English course is used for assessing the success rate of generating summaries for the course related questions. The course has 45 documents fed into Ask4Summary, include academic papers in PDF and the notes and slides in Word and PowerPoint described in Section 2. The course instructor also provides 26 common questions that students in the class usually ask, such as "*what are the differences between academic and non-academic genres*," "*what is plagiarism*," and "*what are the common stages of the conclusion*."

In the evaluation, Ask4Summary generates the summary from six "brains" pre-created by using: (1) Python NLTK reading the academic papers in PDF format only; (2) Python NLTK reading notes and slides in Word and PowerPoint format only; (3) Python NLTK reading every course materials; (4) AWS Comprehend reading academic papers in PDF only; (5) AWS Comprehend reading notes and slides in Word and PowerPoint format only; and (6) AWS Comprehend reading every course materials. The summary generation results are saved as the format shown in Table 1.

Table 1. The generated summary records for queries (partial).

id	question	read_tool	content_tool	source	success	summary
2	What is genre	NLTK	NLTK	paper	yes	Biber (2006), for example, shows us that …
25	What are the differences between academic and non-academic genres	AWS	NLTK	all	no	
42	What is nominalization	NLTK	AWS	docu-ment	yes	Nominalization refers to …
…	…		…	…	…	

Each query has its own *id* generated by the service and the questions are saved in the *question* field. The method used in reading question is saved in the *read_tool* column and the method used in reading and processing the course materials is saved in the *content_tool* column – the reading method could be *NLTK* (Python NLTK) or *AWS* (AWS Comprehend). Moreover, the course material subset used for generating summaries is recorded in the *source* column, which annotates *paper* (academic papers in PDF format only), *document* (notes and slides in Word and PowerPoint format only), or *all* (all the course materials). The column, *success*, indicates whether or not Ask4Summary generated the summary successfully and the generated summaries will be kept in the *summary* field.

4 Preliminary Assessment Results

First, the success rate in generating summaries is only 37.50% (117 of 312 queries); only 2 of the success queries were using AWS Comprehend as reading method for processing the questions. A successful query in this preliminary assessment means that Ask4Summary can generate summary for the asked question without considering how relevance and/or how satisfaction the generated summary could bring to the user who asked the question.

The research applied Chi-square test on the dataset with SPSS 28.0 showing that using Python NLTK to read questions has significantly higher successful rate (73.72%, 115 of 156) in terms of generating summaries than using AWS Comprehend (1.28%, 2 of 156) where $\chi^2(1, n = 312) = 174.619$, $p < .001$. Among the 117 success queries, 67 summaries were generated from the datasets that were built by using Python NLTK to read course materials and 50 of them were built by using AWS Comprehend.

Because the success queries that use AWS Comprehend to read questions are too few (i.e., only two), the preliminary assessment then focuses on the 156 queries that use Python NLTK to read the 26 questions and would like to see if there is a significant difference in terms of successfully generating summaries with different datasets built by using Python NLTK and AWS Comprehend reading methods, based on subsets of materials. The Chi-square test result is $\chi^2(1, n = 156) = 11.944$, $p < .001$. According to the data in Table 2, the datasets – despite of which subset of materials were using – built by Python NLTK performs better than AWS Comprehend.

Table 2. The Chi-square test for the success rate in data sets using different NLP toolkits in the built-in algorithm when reading questions using Python NLTK

	No	Yes	Total
NLTK	11 (14.10%)	67 (85.90%)	78
AWS	30 (38.46%)	48 (61.54%)	78
Total	41	115	156

To figure out which subset of materials is better for Ask4Summary to generate summaries for the 26 common questions that students would ask in the course, despite of the methods used to read subsets of materials the Chi-square test is applied in the queries that only use Python NLTK reading questions. The Chi-square test reveals significant differences – $\chi^2(2, n = 156) = 6.419$, $p = .040$ – on generating summaries with the datasets built by using different subsets of course materials, *paper* (academic papers), *document* (notes and slides), and *all* (all the course materials). The result only shows that there is a significant difference among the subsets but not telling us which subset is significant having better performance than the others.

The Bonferroni correction is applied on the original Chi-square test and the results are listed in Table 3. Both *paper* and *document* categories are annotated with a subscript – *a*, indicating that there is no significant difference on having success summary generation rates between *paper* and *document* as material source. Similarly, there is no significant difference between using documents only and using all materials as the source when Bonferroni correction annotates a subscript – *b* – on both *document* and *all* categories. However, the category *paper* and *all* have different subscript annotations, suggesting that there is a significant difference between the use of academic papers only and the use of all materials; the results show that the use of all materials has better performance with 82.69% success rate than only using academic papers (61.54%) while generating summaries.

Table 3. The Bonferroni correction applied in Chi-square test when comparing the success rate of generating summaries with different subsets of materials.

		Paper	Document	All	Total
No	Count	20_a	$12_{a, b}$	9_b	41
	% within generation	38.46%	23.08%	17.31%	26.28%
Yes	Count	32_a	$40_{a, b}$	43_b	115
	% within generation	61.54%	76.92%	82.69%	73.72%
Total	Count	52	52	52	156
	% within generation	100.0%	100.0%	100.0%	100.0%

We would further like to investigate the influence that subsets might have impact on the success rate of summary generation when considering reading methods separately. While still only considering the use of Python NLTK to read question, the Chi-square test shows that there is no significant difference on the success rate of generating summary among those subsets all read by using Python NLTK earlier with $\chi^2(2, n = 78) = 2.752$, $p = .253$. Similarly, the Chi-square test also shows that there is no significant difference among the subsets read by using AWS Comprehend with $\chi^2(2, n = 78) = 4.225$, $p = .0.121$. Table 4 lists the Bonferroni correction . All the three subsets are

annotated with a subscript – *a*, indicating that there is no significant difference among the subset after applied the Bonferroni correction on the Chi-square test.

Table 4. The Bonferroni correction applied to Chi-square tests when comparing the success rate of generating summaries with different subsets read by different reading methods.

			Paper	Document	All	Total
NLTK	No	Count	6_a	3_a	2_a	11
		% within generation	23.08%	11.54%	7.69%	14.10%
	Yes	Count	20_a	23_a	24_a	67
		% within generation	76.92%	88.46%	92.31%	85.90%
	Total	Count	26	26	26	78
		% within generation	100.0%	100.0%	100.0%	100.0%
AWS	No	Count	14_a	9_a	7_a	30
		% within generation	53.85%	34.62%	26.92%	38.46%
	Yes	Count	12_a	17_a	19_a	48
		% within generation	46.15%	63.38%	73.08%	61.54%
	Total	Count	26	26	26	78
		% within generation	100.0%	100.0%	100.0%	100.0%

5 Discussion and Future Works

According to the analysis results of the preliminary assessment, using AWS Comprehend to read questions has a very low success rate (1.28%) in generating summaries. It is because the questions are usually short and AWS Comprehend cannot properly extract and identify key phrases from short sentences. When only considers the use of Python NLTK to read questions, the overall success rate of generating summaries for the 26 course related questions is increasing to 73.72% (115 of 156) from 37.50%. The datasets built by using reading method Python NLTK also have better performance (85.90% success rate for generating summaries) compared to AWS Comprehend's 61.54% – despite of which material subset was used. This higher summary generation rate might be caused due to the use of same method for reading the questions and course materials.

The assessment also shows that Ask4Summary has higher success rate in generating summaries from the subset pre-built with more course materials: it has a significantly higher success rate (82.69%) from the subset built with both academic papers and teaching materials than from the subset built with only academic papers (61.54%). However, when we consider the method used for reading the subsets of course materials separately, we find that there is no significant difference in terms of the success rate between the use of academic paper only and the use of everything. The Chi-square tests and Bonferroni correction results (see Table 4) on one hand show the course instructor that no material is more than the others and all materials included in her course are equally important for her students and lectures. On the other hand, the results also show more materials read by Ask4Summary no matter which reading method it uses, better chance it can generate summaries for student questions.

The follow-up works the research team right now works on is to ask the course instructor to evaluate the summaries generated by both Ask4Summary and ChatGPT in

terms of the perceived relevance and satisfaction toward a question-summary pair service. Through the follow-up works we can figure out (1) either Python NLTK or AWS Comprehend is more suitable for Ask4Summary's summary generation algorithm, (2) which subset of course materials can help Ask4Summary to generate more relevant and better summaries, and (3) the course instructor's perceptions toward Ask4Summary and ChatGPT and how different the perceptions would be. We can also analyze those summaries that receive lower perceived relevance and satisfaction from the teacher to understand how to improve the Ask4Summary algorithm.

Reference

1. Bird, S., Loper, E., Klein, E. (2009), Natural Language Processing with Python. O'Reilly Media Inc. (2009)
2. El-Kassas, W. S., Salama, C. R., Refea, A. A., Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. Expert System with Application, 165, 113679.
3. [removed for blind review].
4. [removed for blind review].
5. [removed for blind review].
6. Taratukhina, Y. V., Bart, T. V., Vlasov, V. V.: Machine learning models OF information recommendation system ON individualization OF education. Educational Resources and Technology 2(2), 7–14 (2019).