Hyper-QKSG: Framework for Automating Query Generation and Knowledge-Snippet Extraction from Tables and Lists

Dooyoung Kim[1], Yoonjin Jang[1], Dongwook Shin[2], Chanhoon Park[2], Youngjoong Ko[1]*
[1]SungKyunKwan University, [2]NAVER,
{kdysunleo, yali98}@g.skku.edu, yjko@skku.edu
{shin.dongwook, chanhoon.park}@navercorp.com

Abstract

These days, there is an increasing necessity to provide a user with a short knowledge-snippet for a query in commercial information retrieval services such as the featured snippet of Google. In this paper, we focus on how to automatically extract the candidates of query-knowledge snippet pairs from structured HTML documents by using a new Language Model (HTML-PLM). In particular, the proposed system is powerful on extracting them from Tables and Lists, and provides a new framework for automate query generation and knowledge-snippet extraction based on a QA-pair filtering procedure including the snippet refinement and verification processes, which enhance the quality of generated query-knowledge snippet pairs. As a result, 53.8% of the generated knowledge-snippets includes complex HTML structures such as tables and lists in our experiments of a real-world environments, and 66.5% of the knowledge-snippets are evaluated as valid.

1 Introduction

The continued expansion of the internet landscape and the explosion of digital content have led to a tremendous increase in the amount of web data, including news, blogs, forums, social media, and other websites. In the vast ocean of information, users are demanding effective and expedient methods of filtering and accessing information, which can return the information that meets the user's needs. In this context, knowledge-snippet ("Featured Snippets" in Google), a compressed excerpt that contains the answer to a user query, are playing an important role in helping users obtain the information they require with greater speed and ease. Users should be able to get enough information relevant to their query from the knowledge snippet, and knowledge snippets should be able to be extracted from different document structures, such as lists and tables, as needed.

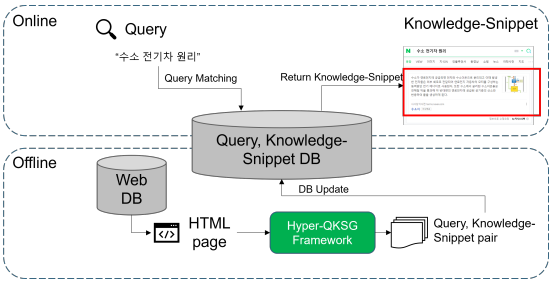


Figure 1: Overview of Knowledge-Snippet Service. For search engines, the return of search results needs to be done in a very short time, so the framework needs to be able to pre-generate knowledge-snippets and queries.

While several studies have demonstrated satisfactory performance in extracting knowledge-snippets from plain text documents, there are two primary limitations in returning knowledge-snippets from real-world web documents. Firstly, it is more challenging for models to identify knowledge-snippets when the user desired answer is located within a section of the structured part, such as a table and a list. Secondly, since the users have limited patience, the system has a limitation in utilizing sophisticated language models, which can significantly prolong the search process.

In response to these limitations, we suggest a novel framework: Hyper-QKSG. Our system trains an HTML-based language model for extracting information from the structured HTML documents. The framework utilizes the model to generate anticipate query-knowledge snippet pairs for each document and score them for further verification and refinement in order to enhance their quality. Our framework are more than 66.5% useful without any post-processing in human evaluation on real-world environments. Figure 1 shows how the Hyper-QKSG can work in real-world web search situations.

*Corresponding author

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 1351–1360

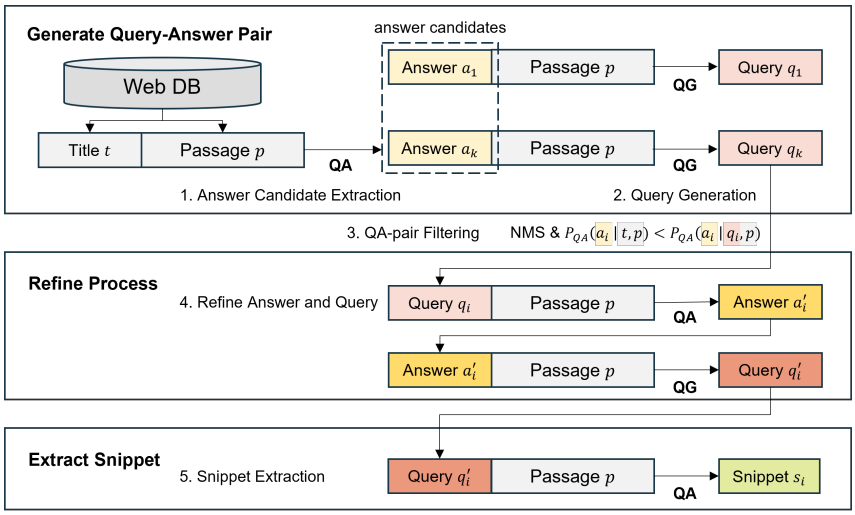


Figure 2: Hyper-QKSG is a framework for generating relevant queries and knowledge-snippets for a document without the user's query information. It consists of three main stages: Generate Query-Answer Pair, Refine Process, and Extract Snippet.

2 Related Work

2.1 Query-aware Snippet Extraction

Query-aware snippet extraction is a commonly employed technique aimed at aiding users in grasping webpage content before clicking (Bando et al., 2010, Chen et al., 2020). Recently, Zhong et al. (Zhong et al., 2021) presented QMSUM, which employs a fixed PLM and CNN to encode sentences and queries, alongside a Transformer to model interactions between sentences. In a similar vein, Zhao et al. (Zhao et al., 2021) proposed QBSUM, which concatenates queries and webpage bodies, leveraging multiple predictors to compute relevance scores. For abstractive generation models, Ishigaki et al. (Ishigaki et al., 2020) utilized an RNN network with a copy mechanism to generate query-aware snippets.

2.2 Visually-rich Document Understanding

There have been many attempts to understand not only unstructured textual data but also structured visually-rich data. The SOTA was achieved in tasks such as complex document understanding (Graliński et al., 2020), document type classification (Harley et al., 2015), and document visual question answering (Mathew et al., 2021) by pretraining using digitally-born PDF files. Meanwhile, attempts have also been made to understand markup-languages such as HTML/XML. Xie et al. (Xie et al., 2021) utilizes an encoder model to encode web content, detects regions of interest, and then extracts relational triples.In addition, Li et al. (Li et al., 2022) jointly pre-learns text and markup languages in a single framework for markup-based Visually-rich Document Understanding tasks.

3 METHODOLOGY

Figure 2 provides the overview of Hyper-QKSG. The overall framework consists of the following three stages: 1) generating Query-Answer (QA) pairs to extract knowledge-snippets, 2) improving QA pairs with refinement, and 3) extracting knowledge-snippets. For the first stage, the HTML-answer extraction model and the query generation model are used to extract answer candidates and generate their corresponding queries, respectively. In the second stage, the answer candidate and query pairs are refined sequentially. In the third step, the HTML-snippet extraction model is developed to extract knowledge-snippets.

Section 3.1 introduces the details of the overall framework, Section 3.2 explains how to pretrain the HTML-PLM model, which is the backbone model of the HTML-answer and HTML-snippet extraction models, and Section 3.3 discusses the finetuning of each pipeline system in the overall framework.

3.1 Hyper-QKSG Framework

3.1.1 Query-Answer Pair Generation

Answer candidates extraction Firstly, the HTML-answer extraction model finds answer candidates that could be the correct answer. The title and passage of a document are used as input to the HTML-answer extraction model, and important keywords of the document are selected as answer

candidates. The HTML-extraction model outputs the probability of each extracted answer candidate, and it will be used as the confidence score of each extraction answer, as Equation 1.

$$s(a_i) = P_{QA}(a_i|t, p) \quad (1)$$

where $s(a_i)$ is the confidence score of answer candidate a_i, P_{QA} is the probability that the HTML-answer extraction model predicts a_i as answer candidate, t means the title of a document, and p denotes each passage of the document. The top K answer candidates are extracted based on these confidence scores. We attempt to extract more diverse answer candidates by changing the token lengths in two types; answer candidates with less than 4 tokens and ones with 4-16 tokens, separately.

Query generation The HyperCLOVA(Kim et al., 2021), Korean pretrained decoder model, is employed for generating queries for each answer candidates. The query generation model predicts a query to induce each answer candidate by using the prompt, "<Context> passage <Target> answer candidate <Query>".

QA-pair filtering Generated query-answer (QA) candidate pairs are double-checked to ensure that they are appropriate or not. During this process, two methods are used to remove inappropriate QA paris: non-maximum suppression (NMS) and answer filtering by confidence score.

Because the HTML-answer extraction model predicts the start and end tokens independently, there may be some overlapping answer candidates in the top-k pairs. Thus we employ NMS technique (Canny, 1983), which was mainly used in the field of object detection in computer vision. We remove one answer candidate with lower answer confidence score, if the Intersection over Union (IoU) value between two answer candidates is calculated with token intersection and the value is greater than threshold t.

Due to the structural information in the document, some unimportant spans are extracted as answer candidates and it lead wrong QA pairs. To eliminate such cases, we propose a confidence score-based answer filtering technique, which is based on the assumption that if the QA pair has a significant relationship, the confidence score of the answer candidate extracted based on the generated query is higher than the score of the answer candidate extracted based on the title. We calculate the confidence score of each answer candidate based on the generated query and remove the QA pairs whose confidence score is lower than the existing confidence score. This allows filtering out meaningless answer candidates or irrelevant generated queries.

3.1.2 Answer and Query Refinement

Since the answer candidates from Section 3.1.1 are obtained by title instead of the query, the generated QA pair could be of poor quality. Therefore, the generated QA pairs should be refined by re-extracting the answer candidates and re-generating their queries. A generated query and its corresponding document are inputted into the HTML-answer extraction model to predict the new answer of the query. To re-extract the answer, we utilize the generated query and the HTML-answer extraction model with the query can more successfully predict the answer span. Then the query can be re-generated by using the re-extracted answers. This refine process can be iteratively done until the extracted answer and generated query do not changed.

3.1.3 Knowledge-Snippet Extraction

For QA pairs generated from a document, the HTML-snippet extraction model searches the relevant knowledge-snippet from the document. The HTML-snippet extraction model finds a knowledge-snippet span that well carries information about the query. The knowledge-snippet span is limited to contain the answer span extracted in the previous step. Moreover, the extraction scope can be limited based on the position of the answer area to prevent the knowledge-snippet from becoming too long. The effect of the limited extraction scope can be seen in the performance part of the HTML-snippet extraction model in Section 4.3. Finally, a knowledge-snippet span can be extracted based on the given document and the QA pair generated in the previous step.

3.2 HTML-PLM

It has been claimed that existing plain text-based pretained language models have limitations in understanding markup languages (Li et al., 2022, Aghajanyan et al., 2022). In addition, some studies claimed that it is more challenging to find the correct answer in documents with more complex structure such as tables than to find the correct an-

Top-10 documents	ratio (%)	ROUGE-1	Human
w/ D_{gt}	98.0	0.205	0.668
w/o D_{gt}	2.0	0.160	0.500
Total	100.0	0.204	0.665

Table 1: The performance of Hyper-QKSG

swer in plain text (Jin et al., 2022, Pasupat and Liang, 2015, Kim et al., 2019). To pretrain the HTML-PLM model, the existing Korean pretrained encoder model, LAYN, is chosen as a backbone model and reformed according to the structure of Longformer (Beltagy et al., 2020) with its input length of 1,024 and attention style. In addition, the segment embedding is added to the model that is trained to distinguish between queries and passages, and it can make our model to well perform on Machine Reading Comprehension (MRC) tasks. To train the model based on the relationship between the HTML document and the query that will be entered, three objective functions are deployed in our HTML-PLM: Masked Markup Language Modeling (MMLM), Node Relation Prediction (NRP), and Query-Page Matching (QPM). The former two functions are from the previous research by Li et al. (Li et al., 2022). With MMLM, the model can enhance the language modeling ability with the markup clues, by randomly selecting and replacing some tokens with [MASK] and recovering the masked tokens with all markup clues, and with the NRP task, the model can find the semantics of Xpath embedding by predicting the relationship between a pair of nodes. In the case of QPM, it is designed for models to efficiently utilize the self-supervised information by predicting whether the query is relevant for the document or not. In pretrain stage, we utilized the queries that returned each document as part of the search results as pseudo queries for each document. Given an input of pseudo query and document, the QPM task is trained to predict whether pseudo query is a randomly sampled or relevant. It allows HTML-PLM to better adapt to the input of the query-passage structure during the finetuning process.

3.3 Finetuning

HTML-answer extraction model This model is fine-tuned to identify a correct answer to a given query. Similar to MRC, the training method requires the model to predict the start and end tokens of the correct answer using the existing MRC dataset that already contains questions and answers labelled for each document. Annotators convert questions into the form of queries used by search systems. The finetuned HTML-answer extraction model is used in answer refinement in Section 3.1.2 as well as answer candidates extraction in Section 3.1.1.

HTML-snippet extraction model The HTML-PLM extraction model is also finetuned to identify the knowledge-snippets that contains the correct answer to each query, similar to the HTML-answer extraction model. When this model has a query-document pair as input, it also predicts the start and end tokens of the knowledge-snippet for the query. For training, the existing MRC dataset is utilized; annotators have manually tagged knowledge-snippets with 2 or 3 sentences for each QA pair. The details of constructed data is described in the dataset of Section 4.1.

Query generation model The HyperCLOVA, a pretrained transformer decoder model, is trained to generate queries for a given answer. The format "<Context> passage <Target> answer <Query>" is used as a prompt for finetuning and inference. The existing MRC dataset is utilized for finetuning.

4 EXPERIMENTS

4.1 Experiment Settings

Dataset To evaluate its applicability to real-world services, we built a knowledge-snippet dataset. In this dataset, one ground-truth knowledge-snippet, one ground-truth document, and top-10 relevant documents retrieved through a search engine for each query are provided; the ground-truth knowledge-snippet is extracted from the ground-truth document and these 10 relevant documents may not include the ground-truth document.

We use the KorQuAD 2.0 dataset (Kim et al., 2019) and Administrative Document Machine Reading dataset (ADMR)[1] for finetuning and evaluate each pipeline system. The KorQuAD 2.0 dataset is Korean wiki based MRC dataset with original HTML documents. It contains 10% and 22% questions that required to find the answer from list and table structures. ADMR dataset is also Korean based MRC dataset and we use Table QA subcategory, which allows to find the correct answer only in tables. The

[1] https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=&topMenu=&aihubDataSe=data&dataSetSn=569

questions in each dataset are converted to query by annotators. We also tagged 2 or 3 sentences containing the answer as a knowledge-snippet span.

Evaluation Metrics We conducted both quantitative and qualitative evaluations to measure the performance of the overall framework. First, we measured the ROUGE-1 similarity between the ground-truth knowledge snippet and generated knowledge snippet on the knowledge-snippet dataset. In addition, we performed human evaluation (Human); human annotators evaluated the generated query-knowledge snippet pairs whether a query was relevant to the document and whether the knowledge snippet contained an informative answer.

For each pipeline system, we adopt the metrics of Exact Match (EM), F1 to evaluate extraction models, and BLEU to evaluate query generation model.

4.2 Query-Knowledge Snippet Generation

Performance of Hyper-QKSG We design the following experimental settings to evaluate the applicability of the query-knowledge snippet pairs generated by the proposed Hyper-QKSG framework to real-world retrieval systems. Given the query and relevant documents, we first construct k pseudo query-knowledge snippet pairs for each document using the Hyper-QKSG framework. Then the most relevant one is selected by query similarities between the given query and the generated pseudo query. The selected knowledge-snippet is quantitatively evaluated by comparing with the ground-truth knowledge-snippet, and also human evaluation is performed to ensure that the returned knowledge-snippet is relevant to the given query.

Table 1 shows the evaluation results of the knowledge-snippets generated by the Hyper-QKSG framework. The D_{gt} indicates whether a ground-truth document, in which the ground-truth knowledge-snippet is extracted, exists in top-10 relevant documents. Comparing the ROUGE-1 with the ground-truth knowledge-snippet, the score is 0.205 when the ground-truth document exists in the top-10 documents and 0.160 without the ground-truth document. The human evaluation results show that 66.8% of the extracted knowledge-snippets are valid when the ground-truth document exists, and 50% are valid when the ground-truth document does not exist. Overall, the probability that the knowledge-snippets generated by the Hyper-QKSG framework actually provide meaningful information to the user is 66.5%. In addition, it is possible to provide more meaningful and trustworthy knowledge-snippets by utilizing features other than the content of the document (e.g., whether the document is from an official site) and constructing training data from more various application domains.

Format	ratio (%)	ROUGE-1	Human
plain-text	46.2	0.200	0.597
table	20.1	0.226	0.725
list	20.1	0.213	0.750
table+list	13.6	0.155	0.667

Table 2: The performance of generated knowledge-snippet according to its format

Model	Table (f1)	List (f1)	Total (f1)
LAYN	20.54	26.05	52.28
HTML-answer	**49.28**	**50.27**	**67.39**
	(+28.74)	**(+24.22)**	**(+15.11)**

Table 3: The performance of HTML-answer extraction model on the KorQuAD 2.0 dev. dataset.

Knowledge-snippet format To evaluate whether the Hyper-QKSG Framework can extract knowledge-snippets from various structures, we analyzed the source structures (i.e., plain text, table, lists, and table+lists) of the generated knowledge snippets because the key information of a document might be concentrated on tables or lists. Table 2 summarizes the statistics. The knowledge-snippets are generated from wide variety of document structures, with 20.1% on table, 20.1% on list, and 13.6% on both table and list (table+list).

In Table 2, the higher evaluation scores for knowledge-snippets including table and list structures prove that the Hyper-QKSG framework better extract knowledge-snippets regardless of the complex structure of given source documents because it can well understand the structure of HTML due to HTML-PLM. Appendix A shows an example of a generated query and extract snippet in real-world scenarios.

4.3 Performance of Each Pipeline System

HTML-answer extraction model

To measure the performance of the HTML-answer extraction model, we evaluated it on the KorQuAD 2.0 dataset. We compare our model with LAYN, the backbone encoder model used to train HTML-answer extraction model, and through

Model	EM	F1
LAYN	34.68	58.62
HTML-snippet	35.91 (+1.23)	61.36 (+2.74)
+ answer position	**47.70 (+13.02)**	**76.34 (+7.72)**

Table 4: The performance of HTML-snippet extraction model on the KorQuAD 2.0 dev. dataset.

this comparison, we check out whether it can well reflect the structural features of HTML documents. In Table 3, the HTML-answer extraction model, $HTML - answer$, achieves much better performance than LAYN; "Total" refers to the performance on the entire QA dev. dataset and "Table" and "List" are the results of extracting and evaluating only the cases where the table or list contains the ground-truth answer.

The "Total" performance shows that our HTML-answer extraction model achieves 15.11%p higher performance. In particular, the performance increases are 28.74%p and 24.22%p for questions that require finding answers in tables and lists, respectively. This shows that existing language models such as LAYN have great difficulty to extract information from tables and lists that need structural information, but our HTML-answer extraction model can effectively extract answers by utilizing enough structural information.

HTML-snippet extraction model Herein, we evaluate the Knowledge-Snippet extraction model using the same QA dataset. Using the ground-truth answers in the QA dataset, five annotators conduct the labeling task for a relevant 2-3 sentence region containing a ground-truth answer as a Knowledge-Snippet region, and each model was trained to predict the Knowledge-Snippet regions for a given query.

Table 4 shows the performance of knowledge-snippet extraction. For knowledge-snippet extraction, the HTML-snippet extraction model performed better, but not much better than answer extraction. The performance reduction is due to the longer length of the region being predicted. To improve the performance, we attempt to use the position information of ground-truth answer. Using the position information, we can limit the scope of the knowledge-snippet search within 100 tokens before and after the ground-truth answer span. In this case, we obtain 47.70 and 76.34 in Exactly Match and F1 scores, respectively. In practice, the proposed framework predicts the answer span first and then extracts the knowledge-snippet span later.

Model	BLEU
Human	0.273
HyperCLOVA-XXXXS	0.291
HyperCLOVA-XXXS (Ours)	**0.322**

Table 5: The performance of the query generation model on the KorQuAD 2.0 dev. dataset. The XXXS and XXXXS indicate model size

Therefore, the results of this experiment show that utilizing the answer position results at the overall pipeline system can achieve much better results.

Query generation model To compare the performance of the query generation model, five annotators create ground-truth queries based on the given context, answer, and question in the dev dataset of KorQuAD 2.0, and two human evaluators participate in this evaluation for comparison with the query generation performance of our models; they are not included in the five annotators who convert questions into queries.

Table 5 shows the performance of the human and our query generation models. Both models of different sizes achieve higher BLEU scores than the queries predicted by humans. Although the BLEU scores of 0.291 and 0.322 are sufficiently meaningful performances, we expect to be able to generate more generalized and robust queries if our model can train more different styles of queries.

4.4 Ablation Study

Table 6 shows the effect of the filtering and refining process in the overall framework. To measure the effect of each process, 1 or 3 most relevant knowledge-snippets are selected for each query and they are evaluated by 5 annotators. When the filtering step is omitted in the Hyper-QKSG framework, it reduces the human evaluation score for the knowledge-snippets by 0.02 and 0.052 in Precision@1 and Precision@3, respectively. When refinement step is omitted, the human evaluation scores reduces by 0.04 and 0.012. This means that filtering and refinement processes have a significant effect on improving the quality of knowledge-snippets. Since Precision@1 is a performance metric for the quality of the most relevant knowledge-snippets, the refinement step, which aims to improve the quality of the knowledge-snippets by re-extracting answers and re-generating queries, has more impacts. On the other hand, since Precision@3 is a metric to evaluate how many noisy knowledge-snippets are among the gener-

Method	ROUGE-1	Precision@1	Precision@3
Hyper-QKSG	0.212	0.665	0.680
w/o Refinement	0.194	0.625	0.668
w/o Filtering	0.208	0.645	0.628

Table 6: Ablation Study

ated knowledge-snippets, the filtering step serves to increase precision more by removing noisy QA pairs.

4.5 Filtering and Refinement

In this section, we analyze the filtering process and the refinement process. We analyzed the output of each step of the Hyper-KSQG framework and found several cases and patterns observed during the process.

During the filtering process, we found two main cases. The first is when the wrong answer candidate is extracted. This is usually caused by documents with the wrong title, or by over-focusing on some tags, such as the head tag, to extract content that is not actually important in document. In the second case, the query is generated incorrectly during the query generation process even though the answer candidates are well extracted. In both of these cases, the query and the answer are not related to each other, so the logit value remeasured based on the query is lower and therefore eliminated.

During the Refine process, we identified three patterns. 1) The correct answer span covers unnecessary territory: In this case, by predicting the correct answer span again based on the Query, the range of the correct answer span is more accurately predicted. 2) The answer span picked up unimportant information as answer candidates: This is the same type of case 1 of filtering process, but it was not removed in the filtering process. In this case, the correct answer to the query often exists around the answer candidate, and the correct answer to the query will be located by finding the correct answer span again. 3) The query becomes more natural and specific as the answer is refined: The quality of the regenerated query increases as the answer is refined during the refinement process. When the answer candidate extracts only a part of the important information area, it is often observed that the query is generated in the form of copying the correct answer and the surrounding context. In this case, when the refine answer is re-extracted and the query is regenerated based on the generated query, the query is modified to be more natural and contain more specific information as the correct answer is modified.

4.6 Documents with wrong title

There are some documents in the web document database that have titles that are not relevant to the content. Our framework, which starts with the process of extracting answer candidates based on title, may behave poorly on these documents.

While most of the inappropriate knowledge-snippets can be refined through the filtering and refinement processes mentioned section 4.5, there may still be documents that do not extract enough knowledge-snippets or have inappropriate knowledge-snippets. However, the impact of these cases on the actual knowledge snippet service is negligible. This is because in a real-world search environment, there are many other documents with similar content and correct title, from which appropriate knowledge-snippets can be extracted. When a search is performed based on a user query, the appropriate knowledge-snippets extracted from a correctly titled document may be more relevant than an inappropriate knowledge-snippets extracted from a wrong title.

5 Conclusion

In this paper, we present the query-knowledge snippet extraction framework, the Hyper-QKSG framework, for the effective web search. To develop this framework, we propose HTML-PLM, which pretrained HTML-based language models for information extraction from diverse HTML structures, and it can significantly enhance the performance of HTML-based MRC downstream tasks in our experiments. In addition, we analyze the knowledge-snippets generated by the framework and find that the proportion of knowledge-snippets with table and list structures is very large in real-world data. Therefore, the proposed HTML-PLM is actively utilized in the knowledge-snippet extraction as more important module. To improve query and knowledge-snippet quality, we propose various filtering, refinement, and verification methods. These are proven as effective methods through the ablation study.

Limitations

The Hyper-QKSG model currently extracts only knowledge-snippets from text-based information such as plain-text, tables, and lists. The ideal information retrieval system should also provide knowledge-snippets based on various modalities, such as math formulas, pictures, and videos. We

will explore the application of multi-modal language models, which are currently being actively researched, to develop our framework.

Acknowledgments

This work was supported in part by [Knowledge-snippet Coverage Extension using QA Generation] funded by Naver corporation, South Korea (60), Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2019-II190421, AI Graduate School Support Program(Sungkyunkwan University), 20) and ICT Creative Consilience Program through the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2020-II201821, 20)

References

Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. 2022. HTLM: Hyper-text pre-training and prompting of language models. In *International Conference on Learning Representations*.

Lorena Leal Bando, Falk Scholer, and Andrew Turpin. 2010. Constructing query-biased summaries: a comparison of human and system generated snippets. In *Proceedings of the third symposium on Information interaction in context*, pages 195–204.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

John Francis Canny. 1983. Finding edges and lines in images.

Wei-Fan Chen, Shahbaz Syed, Benno Stein, Matthias Hagen, and Martin Potthast. 2020. Abstractive snippet generation. In *Proceedings of The Web Conference 2020*, pages 1309–1319.

Filip Graliński, Tomasz Stanisławek, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. 2020. Kleister: A novel task for information extraction involving long documents with complex layout. *arXiv preprint arXiv:2003.02356*.

Adam W Harley, Alex Ufkes, and Konstantinos G Derpanis. 2015. Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995. IEEE.

Tatsuya Ishigaki, Hen-Hsen Huang, Hiroya Takamura, Hsin-Hsi Chen, and Manabu Okumura. 2020. Neural query-biased abstractive summarization using copying mechanism. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*, pages 174–181. Springer.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: recent advances. In *China Conference on Knowledge Graph and Semantic Computing*, pages 174–186. Springer.

Boseop Kim, HyoungSeok Kim, Sang-Woo Lee, Gichang Lee, Donghyun Kwak, Jeon Dong Hyeon, Sunghyun Park, Sungju Kim, Seonhoon Kim, Dongpil Seo, et al. 2021. What changes can large-scale language models bring? intensive study on hyperclova: Billions-scale korean generative pretrained transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3405–3424.

Youngmin Kim, Seungyoung Lim, Hyunjeong Lee, Soyoon Park, and Myungji Kim. 2019. Korquad 2.0: Korean qa dataset for web document machine comprehension. In *Annual Conference on Human and Language Technology*, pages 97–102. Human and Language Technology.

Junlong Li, Yiheng Xu, Lei Cui, and Furu Wei. 2022. Markuplm: Pre-training of text and markup language for visually rich document understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6078–6087.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics.

Chenhao Xie, Wenhao Huang, Jiaqing Liang, Chengsong Huang, and Yanghua Xiao. 2021. Webke: Knowledge extraction from semi-structured web with pre-trained markup language model. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2211–2220.

Mingjun Zhao, Shengli Yan, Bang Liu, Xinwang Zhong, Qian Hao, Haolan Chen, Di Niu, Bowei Long, and Weidong Guo. 2021. Qbsum: A large-scale query-based document summarization dataset from real-world applications. *Computer Speech & Language*, 66:101166.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921.

A Examples of Structured Knowledge Snippet

Figure 3-a) is an example of a knowledge-snippet that utilizes the structural information of lists well. An information-rich document, such as a release note for a game, commonly have a hierarchical structure of various lists. The Hyper-QKSG framework well generate queries for specific contexts within this hierarchical list structure, and extract which part is relevant to the query by exactly understanding the structure of the document.

Figure 3-b) shows a result where the knowledge snippet was extracted from a table. When tables contain several rows or columns, it is necessary to return the other rows or columns near the key information relevant to the query. The knowledge snippet from Hyper-QKSG contains calorie of the rose chicken burrito as well as those of others served at the restaurant.

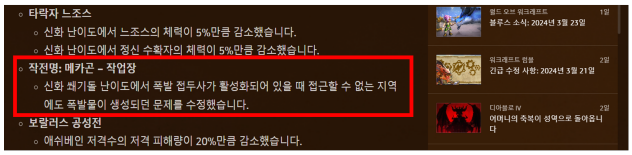
B Experimental Settings

For HTML-PLM, we use the LAYN, a Korean pretrained encoder model, as the backbone model. We follow the settings of Li et al. (Li et al., 2022) for the Xpath embedding of the model, which can provide model with the HTML-tag information of each document. The probability of Masked Language Modeling is 15%. For the Query-Page Matching, we change the query of the document to a random document's query with 50% probability. We train HTML-PLM with 10,000 Korean HTML documents with a batch size of 256 and a learning rate of 1e-5.

For finetuning HTML-answer and HTML-snippet extraction models, we train 3 epochs with batch size 32, learning rate 1e-4, and lr decay 0.8. For the query generation model, we train the Korean pretrained decoder model, HyperCLOVA, in 3 epochs with batch size 8, learning rate 5e-5, and lr decay 0.

In the answer candidate extraction model in Section 3.1.1, 20 answer candidates for each token lengths. The IOU threshold t of NMS is set by 0.5 and 0.25 for 1-4 tokens and 5-16 tokens. Although the refinement process in Section 3.1.2 can be repeated to iterate on the answer span and the query, we only perform the refinement process once in all experiments.

< Knowledge-Snippet >



(EN)
- Operation: Mechagon – Workshop
  - Fixed an issue where Explosives could spawn in inaccessible locations when the Explosive affix is active on Mythic Keystone difficulty.

< Query >
(KO) 작전명 메카곤 작업장 수정 사항
(EN) “Operation Mechagon Workshop Fixes”

a) “World of Warcraft” Release notes (List)

< Knowledge-Snippet >

칠리치즈 포테이토 브리또	1인분 (130g)	327 kcal
토마토파스타소스	1인분 (170g)	150 kcal
로제치킨브리또	1개 (130g)	318 kcal
들깨닭고기 누룽지죽	1인분 (350g)	300 kcal
소고기버섯 누룽지죽	1회 (350g)	285 kcal

(EN)

Tomato Pasta Sauce	1 serving (170g)	150 kcal
Rose Chicken Burrito	1 burrito (130g)	318 kcal
Perilla Chicken Noodle Porridge	1 serving (350g)	300 kcal

< Query >
(KO) 로제치킨브리또 1인분 칼로리
(EN) “Rose Chicken Burrito calories per 1 serving”

b) Food nutrition information (table)

Figure 3: Examples of Knowledge-Snippet from various HTML structures. The red boxes indicate the generated knowledge-snippet by the Hyper-QKSG framework in actual web site. Since the Hyper-QKSG framework generates Korean-based knowledge-snippets and queries, the generated knowledge-snippets and queries are translated into English and labeled as (EN).