Intuitive Access to Smartphone Settings Using Relevance Model Trained by Contrastive Learning

Joonyoung Kim, Kangwook Lee, Haebin Shin, Hurnjoo Lee, Sechun Kang, Byunguk Choi, Dong Shin, Joohyung Lee

Samsung Research
56, Seongchon-gil, Seocho-gu, Seoul, Republic of Korea
{joon0.kim, kw.brian.lee, haebin0.shin, hurnjoo.lee, sechun.kang, byunguk.choi, d0104.shin}@samsung.com, joolee@asu.edu

Abstract

The more new features that are being added to smartphones, the harder it becomes for users to find them. This is because the feature names are usually short, and there are just too many to remember. In such a case, the users may want to ask contextual queries that describe the features they are looking for, but the standard term frequency-based search cannot process them. This paper presents a novel retrieval system for mobile features that accepts intuitive and contextual search queries. We trained a relevance model via contrastive learning from a pre-trained language model to perceive the contextual relevance between query embeddings and indexed mobile features. Also, to make it run efficiently on-device using minimal resources, we applied knowledge distillation to compress the model without degrading much performance. To verify the feasibility of our method, we collected test queries and conducted comparative experiments with the currently deployed search baselines. The results show that our system outperforms the others on contextual sentence queries and even on usual keyword-based queries.

Introduction

Every new smartphone release is accompanied by many new features to attract users. Ironically, it becomes harder for the users to access them because the feature names in Settings are usually concise, each manufacturer calls them differently, and there are too many of them for the users to remember the exact terms. This differs from the standard document search, where the target document is long enough to contain words that capture the users' intent.

For finding a menu in Settings, users may want to ask contextual queries that describe the features they are looking for, but the traditional keyword-based search engines such as TF-IDF (Salton and McGill 1986) and BM25 (Robertson et al. 1995) cannot process them. To overcome the limitation of handling the diversity of users' contextual queries, the current search engines in Android mobiles use look-up tables, but capturing all variations of users' utterances into the look-up tables is not a scalable solution. Also, it is hard to maintain such tables. The problem becomes more severe with the new release with more features.

This paper presents a novel retrieval system for mobile features that accepts intuitive and contextual search queries. The proposed approach can distinguish the relevance among target candidates by understanding contextual semantics via the relevance model trained in a contrastive manner. Furthermore, we applied knowledge distillation to make it run on-device using minimal resources (Hinton, Vinyals, and Dean 2015). By transferring the knowledge in the large model to a compact model with less layers and smaller hidden dimensions, we could reduce the model size to about 1/5 of the original one with only 5% performance degradation.

To verify our method's feasibility and efficiency, we conducted comparative experiments with the currently deployed keyword-based search systems such as OneUI 3.1 and iOS 15.6, which shows that our retrieval system performs better not only on relaxed keyword queries, but also on keyword queries by handling synonyms and compound nouns well.

In summary, this paper makes the following contributions.

- We propose the contextual retrieval system for mobile features using a relevance model trained in a contrastive manner.
- To deploy our method on-device, we successfully applied knowledge distillation to reduce the model size without degrading much performance.
- We demonstrate the advantages and robustness of our system through comparative experiments on various types of queries.

Related Works

Keyword Search

The classical keyword-based search methods, such as TF-IDF (Salton and McGill 1986) and BM25 (Robertson et al. 1995), depend on the exact match between terms in a query and indexed documents while considering Inverse Document Frequency (Luhn 1957; Jones 1972). However, the more features that are being added to a smartphone, the harder it becomes to retrieve relevant search results: the retrieval quality of keyword-based search highly depends on a user's prior knowledge of a target domain to create relevant queries.

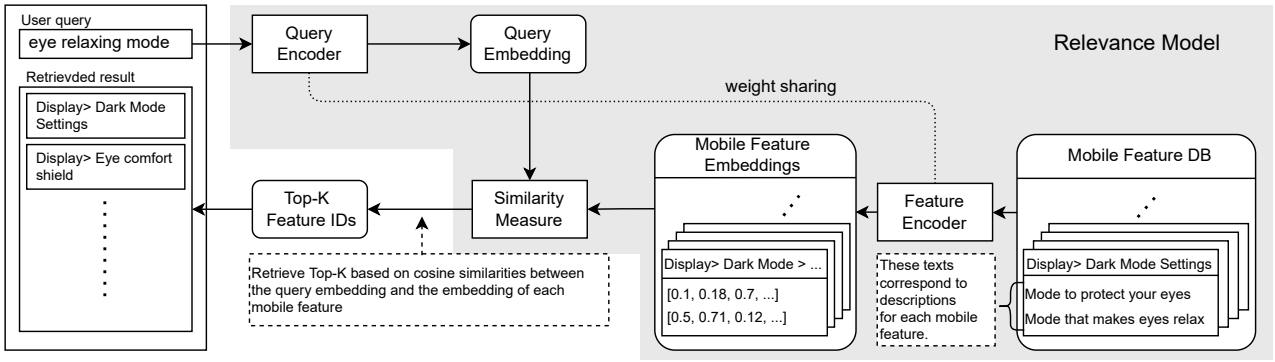


Figure 1: The overview of the retrieval system for mobile features. When a user enters a query, it computes the query embedding through a query encoder. Then, it retrieves top-K relevant candidates based on the cosine similarities between the query embedding and pre-computed embeddings of candidate features via the relevance model.

Neural Search

The neural search engine, which works with vector representations containing textual meanings, can retrieve semantically similar documents even if none of the query terms are matched lexically (Mitra and Craswell 2017, 2018; Mitra, Diaz, and Craswell 2017; Xiong et al. 2017; Dehghani et al. 2017). In (Horita, Júnior, and Júnior 2019), the authors utilized Word2Vec (Mikolov et al. 2013) to support the search on the features in the Settings app with their semantic embeddings. However, the approach is vulnerable to out-of-vocabulary and cannot consider the contextual semantics of a query. Two types of neural search architectures that utilize pre-trained language models, such as BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019), and ELECTRA (Clark et al. 2020), have been proposed to overcome the limitation of the fixed embedding-based approaches. Cross-encoder (Wolf et al. 2019; Vig and Ramea 2019) shows good accuracy but has a high computational complexity in computing the embeddings of the query-indexed data combinations each time a new query enters. On the other hand, Bi-encoder (Mazaré et al. 2018; Dinan et al. 2018; Reimers and Gurevych 2019) seeks efficiency with little accuracy degradation by measuring the relevance of a query embedding over the pre-computed embeddings of the indexed data.

Knowledge Distillation

One effective way to compress a large model is knowledge distillation (KD). It exploits a large model's logits as soft labels for a student model. In a pioneering work, (Hinton, Vinyals, and Dean 2015) proposes this mechanism assuming output logits represent the knowledge of neural networks. Subsequent studies on KD have been proposed: Knowledge types (Kim and Rush 2016; Turc et al. 2019; Park et al. 2019), distillation algorithms (Zhang et al. 2019), online KD (Anil et al. 2018), and a theoretical explanation (Phuong and Lampert 2019).

Effective Retrieval

Here we propose a retrieval system to ease users in finding a desired feature. It has a bi-encoder architecture utilizing pre-computed embeddings of indexed data for computational efficiency. To train an encoder, we design a Siamese network (Koch et al. 2015) and train it in a contrastive learning manner with refined relevant query-document pairs. To exploit the prior knowledge of language understanding, pre-trained language models are used for initialization. Specifically, we follow the training strategy of RoBERTa (Liu et al. 2019).

As Figure 1 describes, in an offline manner, the database of mobile features is pre-computed as feature embeddings through the feature encoder to reduce inference latency. When a user enters a query, the query encoder turns it into a query embedding. Then the relevance model estimates the cosine similarities (*relevance*) between the query embedding and the target features. Based on these relevance scores, it retrieves Top-K mobile features for the query. The system is able to capture the contextual meaning by exploiting an expressively abundant representation of a language model.

Pre-trained Language Model

To estimate the relevance score between embeddings of a query and indexed data, it is essential to understand the semantics. A common method is to build a pre-trained language model as an initial point for a relevance model instead of pre-training from scratch. Some off-the-shelf language models, such as BERT (Devlin et al. 2019) and RoBERTa (Liu et al. 2019), represent promising results in various downstream tasks. However, they are not suitable for retrieving mobile features since those models were trained with a general corpus without specific domain knowledge that we require. Thus, we build our own language model as a backbone for the retrieval system. More precisely, we built two language models, one in English and the other in Korean, using Transformer encoder (Vaswani et al. 2017). To inject domain knowledge to the language models, we construct a training corpus consisting of Wikipedia and smartphone-related articles on the websites such as Samsung Newsroom[1] (an official website introducing Samsung Electronics products), Samsung Members[2] (a community website for

[1]https://news.samsung.com

[2]https://r1.community.samsung.com

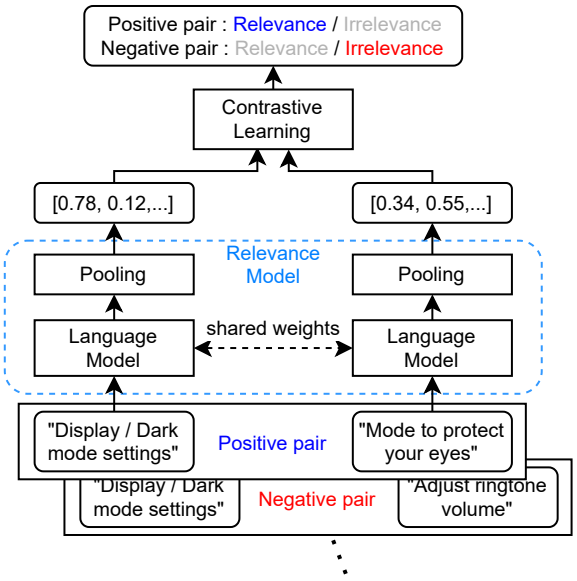


Figure 2: Relevance model with Siamese structures. Each bi-encoder consists of a pre-trained language model and a pooling layer. This model is trained in a contrastive manner with the 1:4 ratio of positive and negative pairs.

Lang.		Eng.		Kor.	
Engine		OneUI	Ours	OneUI	Ours
Exact keyword	P	88.7	93.0	74.2	93.2
	R	100.0	100.0	100.0	100.0
	F1	91.2	94.7	80.7	94.8
Relaxed keyword	P	22.7	36.8	23.6	48.0
	R	35.7	61.9	38.1	72.0
	F1	24.3	42.1	25.5	52.9

Table 1: Performance on keyword queries. Macro-average values of precision (P), recall (R), and F1 score are reported with top-5 retrievals. For each query, irrelevant retrieved results are regarded as negatives.

users of Samsung products), and the E-manual website[3] for Galaxy smartphones.

Moreover, as Korean Wikipedia is much smaller than English Wikipedia, we utilize the newspaper and book corpora released by the National Institute of Korean Language[4] as extra training data.

Relevance Model

The neural search engine retrieves relevant documents based on the cosine similarity between vector representations. We need a specialized encoder called a *relevance model* to compute a vector representation that captures the relevance in a specific target domain. It is usual to train the relevance model with query-document pairs from users in the target search domain. However, we cannot collect such query-document pairs for mobile features due to privacy issues. As an alternative, we generated synthetic training pairs by combining the name of each feature and its descriptions instead of actual user queries. We first extract a feature name and its hint text from the hierarchy in the feature tree. For example, the "*Eye comfort shield*" feature can be paired with "*Keep your eyes comfortable by limiting blue light*". We also added the descriptions in the product manual, such as "*Eye comfort shield can help prevent eye strain, especially when you use your phone at night or in low-light settings*." In total, the training data consist of 862 English and 911 Korean descriptions of 563 mobile features.

Inspired by (Reimers and Gurevych 2019), we built a Siamese network (Koch et al. 2015) with the pre-trained language model and used the synthetic pairs to train the relevance model. As Figure 2 illustrates, each output of the language model is average pooled into a single vector. Then, the relevance score is computed in terms of cosine similarity. We apply contrastive learning to capture the relevance between a query and documents. As for soft negative pairs, we sample irrelevant texts from mini-batch during training as proposed in (Henderson et al. 2017).

Experiments

Here we demonstrate the effectiveness of our neural search engine on both keyword and sentence queries.

Experimental Settings

For a quantitative evaluation, we compared our engine with OneUI 4.0, the latest version of the customized Settings for Samsung mobile devices based on Android 12, which contains 563 mobile features as the search target. To build the search index, we concatenate the hierarchical path from the root to each node (e.g., *"Display - Touch sensitivity"*). We acquired all the experimental results in a single run and drew the results with a confidence threshold. OneUI 4.0 does not provide ranked results since it searches features by matching text in the query and features (Full-Text Search).

We also compare our system with iOS 15.6. Since its menu tree differs from the Android's, a quantitative comparison is not feasible. Instead, we performed the qualitative study using the features common to Android and iOS and examined the first screen of search results, as shown in Figure 3. Overall, it is clear that the iOS retrieval system also heavily relies on term-matching and cannot do well on semantic search.

Evaluation

To measure the performance of our proposed retrieval system on smartphone features, we collected two kinds of test queries: Keyword queries and sentence queries. We hired seven annotators who are experts in Android and asked them to write the keyword and sentence queries for each feature. In detail, the keyword queries are also categorized into *"exact keyword query"* and *"relaxed keyword query"*. The former consists of the exact name of each feature as keywords, while the latter contains alternative keywords describing the feature. The examples and the test queries' statistics are shown in Tables 2 and 3, respectively.

[3]https://www.samsung.com/mobile
[4]https://corpus.korean.go.kr/

Keyword query		Sentence query
Exact keyword query	Relaxed keyword query	*"Touch is not working properly."*
"Touch sensitivity"	*"Mistouch, Block touch"*	*"Screen touch doesn't work when covering screen protector."*

Table 2: Examples of keyword and sentence queries for *"Display - Touch sensitivity"*.

	English	Korean
# Keyword queries in total	1,438	1,442
# Sentence queries in total	1,119	1,140
Avg. # words in a keyword query	3.0	2.9
Avg. # words in a sentence query	6.8	4.6

Table 3: Statistics of two types of test queries. On average, a sentence query consists of twice the words of a keyword query.

Lang.	Engine	H@5	H@10	H@20	H@all
Eng.	OneUI	17.5	19.2	20.0	20.4
	Ours	76.4	83.0	89.8	100.0
Kor.	OneUI	22.0	24.7	26.2	27.4
	Ours	83.3	89.3	93.9	100.0

Table 4: Performance on sentence queries. Since each sentence query usually has one ground truth, Hits@K is adopted as an evaluation metric

Keyword Queries

We conducted comparative experiments on keyword queries with the currently deployed search system, OneUI 3.1, as a baseline. As shown in Table 1, our system considerably outperforms the baseline in both English and Korean queries. Moreover, even when users do not know the exact name of the feature (Relaxed keyword query), it shows better performance. Since the proposed search system utilizes the relevance model, which learns the relevance with a bi-encoder structure, it can retrieve lexically and semantically relevant results.

Synonym Since the keyword-based search in OneUI 3.1 relies on the term frequency, it retrieves disparate results for different queries with similar meanings. However, users may expect the relevant results according to the semantics of the query rather than lexical matching. With contextualized embeddings of queries and contents, our system can capture the meanings of synonym queries. As shown in Figure 3 (top), our search engine consistently retrieves the desired feature, *"Adapt Sound"*, for the semantically similar query (*"Optimize sound"*) at the ranks 4 and 5 while the baseline misses synonyms and the desired result is ranked at 9.

Compound nouns Permutation of compound nouns is common obstacles for the retriever because they can easily confuse the standard search algorithms, which depend on the exact match. Since the order of nouns determines the semantic meanings, it is difficult for them to capture what the compound stands for without context information. For instance, *"Sound notification"* means a type of notification, whereas *"Notification sound"* implies adjusting the volume or style of sound. As in Figure 3 (bottom), ours ranks the relevant results based on the queries, while the baseline just retrieves naive results, ignoring the order of nouns.

Sentence Queries

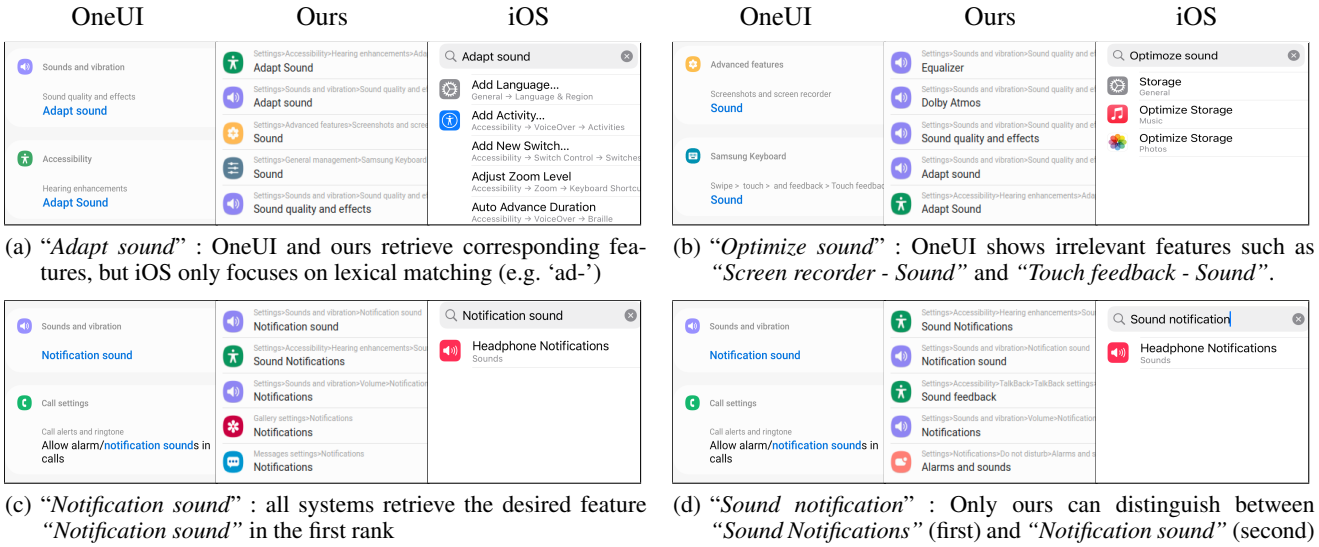
We also evaluate the search performance on sentence queries (e.g., *"I want to share internet through my phones"*). This type of queries is also called *interactive queries* since they are often the questions that the user asks to voice assisant systems. In contrast to keyword queries, each sentence query often has only one ground truth due to its specific intention. Thus, precision and recall are no longer effective in estimating retrieval performance. Instead, we adopt Hits@K as a metric for sentence queries, representing the ratio of the queries which can retrieve at least a single relevant feature in the Top-K results.

As shown in Table 4, OneUI has trouble perceiving the context of sentence queries. Since term frequency-based approaches rely only on whether the feature includes specific query terms, they are vulnerable to lexically close but semantically different queries. Notably, the performance on Korean queries is better than the performance on English queries. This gap is attributed to the performance of the underlying language models since we additionally trained the Korean language model with extra corpora.

For qualitative studies, we sample a few sentence queries to verify the retrieving ability of our system. As shown in Figure 5, ours can retrieve relevant results such as *"Font size"* and *"Screen brightness"* according to the queries *"The letters are too small"* and *"How to dim screen"*, while the baseline only focused on the specific terms *"the"* and *"to"*, respectively. In particular, our system is robust to even long queries such as *"I wanna share internet through my phone"* and *"How to check the remaining battery level"*. Note that the unexpected results of the baseline are not specific to OneUI. For efficiency and simplicity, most search applications adopt the Full-Text Search (Bast and Buchhold 2013) as an auxiliary tool that utilizes a look-up table of frequent words. Although this kind of shortcut is advantageous for frequently used queries, it could negatively affect understanding long sentence queries.

Knowledge Distillation

Although retrieving relevant results is the most important, another essential matter for deployment is the size and complexity of the system. Since mobiles have limited computing resources, it is mandatory to compress the model to minimize response latency and use a small memory. One of the effective methods to compress a neural model is knowledge distillation (Hinton, Vinyals, and Dean 2015), which transfers the knowledge from the original model (teacher) to a smaller one (student). In knowledge distillation, the student



(a) "*Adapt sound*" : OneUI and ours retrieve corresponding features, but iOS only focuses on lexical matching (e.g. 'ad-')

(b) "*Optimize sound*" : OneUI shows irrelevant features such as *"Screen recorder - Sound"* and *"Touch feedback - Sound"*.

(c) "*Notification sound*" : all systems retrieve the desired feature *"Notification sound"* in the first rank

(d) "*Sound notification*" : Only ours can distinguish between *"Sound Notifications"* (first) and *"Notification sound"* (second)

Figure 3: Qualitative comparison of keyword queries, including synonym (top) and compound noun (bottom).

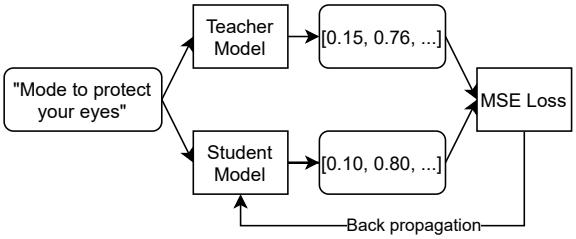


Figure 4: The overview of knowledge distillation process. The MSE loss of the output vectors of student and teacher models is used for the back-propagation of the student model.

model is trained to imitate the output vectors or logits of the teacher model. Figure 4 shows the brief process of knowledge distillation.

The size of our original relevance model is 377MB, which is too large to be deployed on a mobile device. So, we compress it using knowledge distillation. In particular, we employ the technique for compressing transformer-based encoder models proposed in (Turc et al. 2019). First, we set a small language model with fewer layers and hidden dimensions as an initial point of the student model. Then, we train the student model to generate the same embedding with the teacher model. To this end, we feed entire sentences in the Wikipedia dumps and calculate the Mean Squared Error (MSE) loss between the teacher and student model output embeddings.

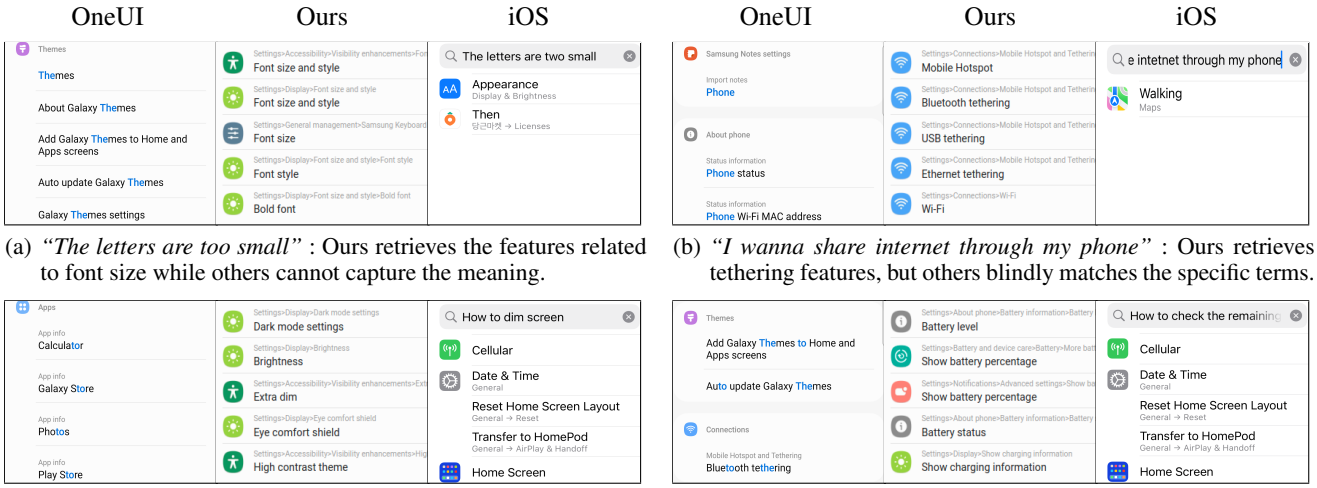
Evaluation We applied knowledge distillation to compress our language model with various combinations. We conducted additional experiments on knowledge distillation to investigate the trade-off between distilled models' size and their performance. In transformer-based models, the number of layers (L) and hidden dimensions (D) determine

Lang.	Model			Performance		
	L	D	Size	H@5	H@10	H@20
Eng.	12	768	377MB	76.4	83.0	89.8
	4	512	82MB	74.7	83.0	89.4
	4	256	29MB	72.7	80.8	87.3
	2	128	9.8MB	67.4	76.2	84.1
Kor.	12	768	377MB	83.3	89.3	93.9
	4	512	82MB	82.4	88.6	93.1
	4	256	29MB	80.1	86.9	92.2
	2	128	9.8MB	71.9	81.0	88.1

Table 5: Performance of distilled models on sentence queries. L and D denote the number of layers and hidden dimensions. Even after the original model is distilled to student one (up to 2.6%), it maintains performance. The original model consists of 12 layers and 768 dimensions.

the complexity of the model. Note that we adopt smaller pre-trained models as the initial state to transfer the learning effect. Table 5 describes the retrieval performance of the distilled models in detail.

The distilled models show competitive performance with moderate sizes compared to the original one. The distilled English model with 4 layers and 512 hidden dimensions maintained 99.1% of the performance of the original one in Hits@20 while reducing the model's size to about 20%. Even in the extreme case when the size is reduced to 2.6% of the original model, the tiny model still achieves 84.1% in Hits@20. Similarly, the distilled Korean one with 4 layers and 512 hidden dimensions maintained 99.1% of the performance of the original one in Hits@20. In the extreme case, the tiny model which is 2.6% of the original model still achieves 88.1% in Hits@20.



(a) *"The letters are too small"* : Ours retrieves the features related to font size while others cannot capture the meaning.

(b) *"I wanna share internet through my phone"* : Ours retrieves tethering features, but others blindly matches the specific terms.

(c) *"How to dim screen"* : Ours retrieves results about screen brightness while others focus on lexical matching

(d) *"How to check the remaining battery level"* : Only ours retrieves the features related to battery level

Figure 5: Qualitative comparison on sentence queries.

Conclusion

We presented a novel search engine that can retrieve relevant results based on semantic similarity in smartphone settings application. We built a Siamese architecture with a pre-trained language model and trained the relevance model via contrastive learning with text pairs of the features. We constructed our own test queries to compare the search performance with the currently-deployed search engine, which adopts Full-Text Search. The experiments show that the proposed system showed better results than the baseline on both keyword and sentence queries. Furthermore, we applied the knowledge distillation techniques to compress the models to board on mobile devices achieving reliable search performance. Clearly, the proposed system is a response to the request to improve the user experience, and we are discussing the deployment plan with the system software group. As the proposed software should be part of the system software (OneUI), rather than a standalone application. The discussion also examines various other issues not directly related to the presented system, such as how to support legacy search results with the neural search and the UI to show the results, etc. We expect the proposed system to be applied to improve settings search, app store search, and troubleshooting search, where users know the various descriptive and semantic information rather than exact terms.

References

Anil, R.; Pereyra, G.; Passos, A.; Ormandi, R.; Dahl, G. E.; and Hinton, G. E. 2018. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*.

Bast, H.; and Buchhold, B. 2013. An index for efficient semantic full-text search. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 369–378.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Dehghani, M.; Zamani, H.; Severyn, A.; Kamps, J.; and Croft, W. B. 2017. Neural Ranking Models with Weak Supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, 65–74. New York, NY, USA: Association for Computing Machinery. ISBN 9781450350228.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*.

Dinan, E.; Roller, S.; Shuster, K.; Fan, A.; Auli, M.; and Weston, J. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.

Henderson, M.; Al-Rfou, R.; Strope, B.; Sung, Y.-H.; Lukács, L.; Guo, R.; Kumar, S.; Miklos, B.; and Kurzweil, R. 2017. Efficient Natural Language Response Suggestion for Smart Reply. *ArXiv*, abs/1705.00652.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Horita, L. R.; Júnior, J. B. P. M.; and Júnior, M. D. P. 2019. Enhancing the Search Tool of the Android Settings through Natural Language Processing. In *Anais Estendidos do IX Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, 83–88. SBC.

Jones, K. S. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Kim, Y.; and Rush, A. M. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Koch, G.; Zemel, R.; Salakhutdinov, R.; et al. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692.

Luhn, H. P. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4): 309–317.

Mazaré, P.-E.; Humeau, S.; Raison, M.; and Bordes, A. 2018. Training millions of personalized dialogue agents. *arXiv preprint arXiv:1809.01984*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 3111–3119. Red Hook, NY, USA: Curran Associates Inc.

Mitra, B.; and Craswell, N. 2017. Neural Models for Information Retrieval. *CoRR*, abs/1705.01509.

Mitra, B.; and Craswell, N. 2018. An Introduction to Neural Information Retrieval. *Found. Trends Inf. Retr.*, 13: 1–126.

Mitra, B.; Diaz, F.; and Craswell, N. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, 1291–1299. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee. ISBN 9781450349130.

Park, W.; Kim, D.; Lu, Y.; and Cho, M. 2019. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3967–3976.

Phuong, M.; and Lampert, C. 2019. Towards understanding knowledge distillation. In *International Conference on Machine Learning*, 5142–5151. PMLR.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982–3992.

Robertson, S. E.; Walker, S.; Jones, S.; Hancock-Beaulieu, M. M.; Gatford, M.; et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp*, 109: 109.

Salton, G.; and McGill, M. J. 1986. *Introduction to Modern Information Retrieval*. USA: McGraw-Hill, Inc. ISBN 0070544840.

Turc, I.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Vig, J.; and Ramea, K. 2019. Comparison of transfer-learning approaches for response selection in multi-turn conversations. In *Workshop on DSTC7*.

Wolf, T.; Sanh, V.; Chaumond, J.; and Delangue, C. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

Xiong, C.; Dai, Z.; Callan, J.; Liu, Z.; and Power, R. 2017. End-to-End Neural Ad-Hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, 55–64. New York, NY, USA: Association for Computing Machinery. ISBN 9781450350228.

Zhang, L.; Song, J.; Gao, A.; Chen, J.; Bao, C.; and Ma, K. 2019. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3713–3722.