Semi-Structured Query Grounding for Document-Oriented Databases with Deep Retrieval and Its Application to Receipt and POI Matching

Geewook Kim\textsuperscript{a,∗}, Wonseok Hwang\textsuperscript{b,†}, Minjoon Seo\textsuperscript{c,†}, Seunghyun Park\textsuperscript{a}

\textsuperscript{a} Clova AI Research, NAVER Corp.
\textsuperscript{b} LBox Co., Ltd.
\textsuperscript{c} Korea Advanced Institute of Science and Technology

Abstract

Semi-structured query systems for document-oriented databases have many real applications. One particular application that we are interested in is matching each financial receipt image with its corresponding place of interest (POI, e.g., restaurant) in the nationwide database. The problem is especially challenging in the real production environment where many similar or incomplete entries exist in the database and queries are noisy (e.g., errors in optical character recognition). In this work, we aim to address practical challenges when using embedding-based retrieval for the query grounding problem in semi-structured data. Leveraging recent advancements in deep language encoding for retrieval, we conduct extensive experiments to find the most effective combination of modules for the embedding and retrieval of both query and database entries without any manually engineered component. The proposed model significantly outperforms the conventional manual pattern-based model while requiring much less development and maintenance cost. We also discuss some core observations in our experiments, which could be helpful for practitioners working on a similar problem in other domains.

1 Introduction

Querying a database with semi-structured natural language (e.g., in JSON format) has many real-world applications (Arora and Aggarwal 2013; Bhardwaj 2016). One particular application that we are interested in is Place of Interest (POI) Match, which is an active product where the user input is a paper bill image and the desired output is a database entry that corresponds to the POI where the financial transaction is deemed to have taken place. Most commercial receipts have place information, and thus the receipt and POI matching can be done by extracting the information from the receipt image and querying the POI database. This can be done in steps as follows: (1) An optical character recognition (OCR) is first applied to extract text and its geometric location from the receipt image. (2) A document parsing (information extraction) process is applied to determine which pre-defined category (e.g., store name and store address) the obtained text belongs to (see the details in Figure 1). (3) Finally, the grounding process uses the extracted

Figure 1: Results of our OCR and Parsing web demo. The system successfully extracts structured information of the input paper bill image (upper). However, given a noisy sample (we added common noise to the sample, e.g., low image resolution, perspective view and blur), text recognition (red) and word order prediction (orange) failed (bottom).

\textsuperscript{∗}Corresponding author: gwkim.rsrch@gmail.com

\textsuperscript{†}This work was done while the authors were at NAVER Corp.

\textsuperscript{1}https://clova.ai/ocr
queries and database entries are represented by vectors. (Karpukhin et al. 2020) into database querying. In our system, both embedding-based retrieval (Karpukhin et al. 2020; Huang et al. 2019) and many franchises, so that it is difficult to differentiate. Hence, for robust grounding in a real-world scenario, the model should consider multiple fields simultaneously and should handle both semantic and lexical similarities among the data to distinguish similar entries. Also, the model should deal with the issues of missing values and noises of the data. Because of these difficulties, our current rule-based model has a more than 30% failure rate for all incoming queries.

To address these issues, we combine an idea of embedding-based retrieval (Karpukhin et al. 2020; Huang et al. 2020) into database querying. In our system, both queries and database entries are represented by vectors. Given a query, the corresponding database entries are retrieved by searching nearest neighbors in the vector space. To make a good system for grounding, we first introduce a framework for grounding that consists of several replaceable modules. And then, we conduct extensive experiments to assess the contribution of individual modules more rigorously and find the best module combination over the model architectures. The proposed model successfully alleviates the issues of missing values and noises of the data, and significantly outperforms the rule-based baseline model by more than 8% point of top-1 matching accuracy in POI Match (this corresponds to approximately 2 million user queries per month). While our work primarily focuses on one particular application (POI Match), we believe that our findings can be easily extended to other real applications that rely on semi-structured queries on databases.

2 Background

2.1 OCR and Parsing

Information extraction (IE) on semi-structured document images is a core step towards automated document processing. In general, document IE systems consist of two stages: OCR and Parsing.

OCR The OCR process consists of two sub-steps: detection and recognition. First, in the detection procedure, all text regions in the given image are predicted (Baek et al. 2019b). The recognition module receives the detected image patch as its input and extracts all texts in the patch (Baek et al. 2019a).
### 2.2 Database Querying

In document-oriented databases, the data is stored in the form of semi-structured data that consists of several key-value pairs, e.g., JSON or XML. Database querying is done by retrieving corresponding database entries that satisfy the input conditions. For instance, in MongoDB (Krishnan, Elayidom, and Santhanakrishnan 2016), the database entries whose the value of “color” is “gold” is retrieved by using the input query `find(color: "gold")`. Querying with multiple conditions is also supported, e.g., `find(color: "gold", type: "necklace")`.

Database querying often fails in real-world applications since both queries and database entries are noisy in most practical scenarios, e.g., misspellings in a query or missing values in database entries. A common solution to this problem is using a text-based search engine (e.g., Apache Lucene) (Gormley and Tong 2015; Gupta and Rani 2016). However, the performances are limited as the core idea is based on simple string distance calculation algorithms which are not good at capturing semantic similarities among the data (Robertson and Zaragoza 2009; Gormley and Tong 2015).

### 2.3 Embedding-based Retrieval

To retrieve textual data such as words, sentences, or documents, many modern NLP applications convert the data into vector representations, i.e., embeddings (Mikolov, Yih, and Zweig 2013; Devlin et al. 2019; Karpukhin et al. 2020). Once the data is embedded in vector space, retrieval can be done efficiently by calculating similarities among the vectors. For instance, if the data is embedded into an inner-product space, the retrieval can be done with maximum inner product search algorithms (Ding, Yu, and Hsieh 2019; Tan et al. 2019; Johnson, Douze, and Jégou 2019). The embeddings are expected to hold characteristics, properties, or even semantics of the data so that retrieval targets can be found simply by calculating the distances in the vector space. To obtain such good embeddings, a range of representation learning methods has been studied (Mikolov, Yih, and Zweig 2013; Kim 2014; Bojanowski et al. 2017) and most of the modern methods utilize BERT-based models to embed the data (Devlin et al. 2019; Karpukhin et al. 2020).

### 3 Grounding

In this section, we aim to introduce our model for grounding. Before introducing the model, we formally define grounding and provide some preliminaries.
are learned by maximizing
$$\sum_{i,j,k} \{s(y_i, y_j) \propto \exp(s(y_i, \hat{y}_j))\}.$$ 

The summation in the denominator makes training difficult, especially when \(m\) is large, which is often the case in real-world environments. The problem can be circumvented by using negative sampling, leading to a modification of the above objective as follows,
$$\exp(s(y_i, y_j)) \sum_{k \sim p_{neg}} \exp(s(y_i, y_k)),$$

where \(p_{neg}\) is a distribution for negative sampling, such as uniform, empirical frequency, etc. For efficient model training, we train the models with mini-batch gradient descent and use other samples in a mini-batch as negative samples (Gillick et al. 2019; Karpukhin et al. 2020).

### 3.3 Implementation

We implement a library grounder that includes all fundamental functionality for training and deploying a grounding system. grounder is implemented in PyTorch (Paszke et al. 2019) and built upon two public projects. To use a range of BERT-based models, we use transformers (Wolf et al. 2020) developed by huggingface. For an efficient retrieval of nearest neighbors, faiss (Johnson, Douze, and Jégou 2019) developed by Facebook AI is used. Our implementation will be publicly available on GitHub\(^2\).

### 4 Experiments

In experiments, we study two architectures of backbone (MBERT and XLMR), two similarity functions (IPS and NSD), two options in a separator token (Single and Multi), and three options in masking missing values (None, Single and Multi). All possible grounding module combinations (2x2x2x3 = 24 in total) are evaluated to find the best module combination for our application POI Match. We also assess the efficacy of each module, and we believe that our findings in the experiments can be easily extended to other real-world applications that rely on grounding.

**Common Settings.** Given a receipt image and a database of stores, our task is to find a corresponding store in the database that matches the image. We evaluate all models with top-1 matching accuracy. In experiments, we use 1 million receipt images and a database that contains approximately 6 million POI information. Each query has 4 valid fields, where each field corresponds to the name, telephone number, address, and business number of the store respectively. Each database entry has 4 valid fields, where each field corresponds to the name, telephone number, address, and street name (i.e., another type of address) of the store respectively. There are many missing values in the data, for example, 21% of the database entries have null values on the telephone number field and 17% on the street name field respectively. Each receipt image is linked to a database entry with the help of our rule-based model that has been in service in our application; the model is first applied to the image to find a corresponding database entry, and the links are refined by a human annotator. The application is deployed in South Korea and the main language of the data is Korean.

### 4.1 Comparison Models

**Baseline** To assess the efficacy of our proposal, we use our conventional rule-based model as a baseline model. The model queries a store with regular expression-based searching algorithms on specific fields, such as telephone number

---

\(^2\)https://github.com/clovaai
or address. The model may not be able to return a single candidate, for example, franchise stores tend to have the same values in some fields, e.g., telephone number, so matching tends to be more difficult. If multiple entries are returned, predefined rules are applied for re-ranking. For example, the most visited store is returned based on the history. These heuristic rules are hard to cope with the various exceptions in real-world environments.

**Module Combinations** As explained in Section 3.1, we build a grounding model by combining several replaceable modules as explained below.

**Backbones.** To embed the multilingual data, we test two BERT-based models; Multilingual BERT (MBERT) (Devlin et al. 2019; Pires, Schlinger, and Garrette 2019) and XLM-Roberta (XLMR) (Conneau et al. 2020).

**Similarity functions.** To score the similarities between the data (i.e., JSON objects), two similarity functions are tested: inner-product similarity (IPS) \( \langle y_i, y_j \rangle \) and negative squared distance (NSD) \(-||y_i - y_j||^2\).

**Separator token for field concatenation.** In our pipeline, each JSON object is converted into a sequence of tokens. During the conversion, we simply concatenate all values in the valid fields separated by a single separator token between the values (Single) (Devlin et al. 2019; Karpukhin et al. 2020). To emphasize the distinctions between values from different fields, we also test multiple field-wise separator tokens (Multi). See Figure 3 and Section 3.1 for more details.

**Masking token for missing values.** In Devlin et al. (2019), a special mask token is used to train the model to capture the associations among input tokens. During the training, the input tokens are randomly replaced by the mask token and the model tries to recover the masked values from its neighbors. Inspired by this, we use the mask token to mitigate the negative ramifications of missing values in the data. When the value is missing, we use either a single masking token (Single) or multiple field-wise masking tokens (Multi) instead of leaving it as blank (None). See Figure 3 and Section 3.1 for more details.

### 4.2 Results and Analysis

We conduct extensive experiments to assess the effectiveness of each module and to find the optimal combination.

**Experiment 1 (Module Combinations).** We evaluate all module combinations (2×2×2×3=24 in total). The results are shown in Table 1. In this experiment, we split the 1M receipt images into test (10K) and train (the rest). We use the database entries (390K) that are associated with the 1M receipts. The models are trained to learn the associations between trainset receipts and database entries. And the trained models are used to predict unobserved associations from the test set receipts. The batch size is set to 32, the number of steps is 40K, and the learning rate is 2e-5. As can be seen in Table 1, there are considerable performance gaps among the combinations ranging from 85.6 to 91.6. The best module combination is MBERT-NSD-Multi-Multi. To show the effect of each module more comprehensively, we plot the module-wise performances in Figure 4. MBERT has a large performance variance, and the combination of MBERT and NSD seems to be effective. The result also shows that using multiple special tokens (Multi) is effective for mitigating the issues of multiple field concatenation as well as missing values in the data.

**Experiment 2 (Valid Field Selection).** We also investigate the effects of using multiple fields by changing the number of valid fields during the training. The results in Table 2 show the matching accuracy becomes higher as the number of valid fields grows. This highlights the importance of using multiple fields to differentiate similar entries in the database, leading to a performance improvement in grounding.

**Experiment 3 (Results on POI Match).** To find out how effective the proposed system is in our application environment, we evaluate the model on the daily queries from real-world environments. The results are presented in Table 1. In this experiment, we split the 1M receipts into test (10K) and train (the rest). We use the database entries (390K) that are associated with the 1M receipts. The models are trained to learn the associations between trainset receipts and database entries. And the trained models are used to predict unobserved associations from the test set receipts. The batch size is set to 32, the number of steps is 40K, and the learning rate is 2e-5. As can be seen in Table 1, there are considerable performance gaps among the combinations ranging from 85.6 to 91.6. The best module combination is MBERT-NSD-Multi-Multi. To show the effect of each module more comprehensively, we plot the module-wise performances in Figure 4. MBERT has a large performance variance, and the combination of MBERT and NSD seems to be effective. The result also shows that using multiple special tokens (Multi) is effective for mitigating the issues of multiple field concatenation as well as missing values in the data.

![Table 1: Performances of all module combinations.](image)

| Backbone | Sim. | Sep. | Mask. | Acc.  |
|----------|------|------|-------|-------|
| MBERT    | IPS  | Single | None  | 86.61 |
| MBERT    | IPS  | Single | Single | 87.45 |
| MBERT    | IPS  | Single | Multi  | 85.38 |
| MBERT    | IPS  | Multi  | None  | 88.91 |
| MBERT    | IPS  | Multi  | Single | 88.01 |
| MBERT    | IPS  | Multi  | Multi  | 88.15 |
| MBERT    | NSD  | Single | None  | 85.61 |
| MBERT    | NSD  | Single | Single | 87.09 |
| MBERT    | NSD  | Single | Multi  | 90.26 |
| MBERT    | NSD  | Multi  | None  | 90.85 |
| MBERT    | NSD  | Multi  | Single | 90.73 |
| MBERT    | NSD  | Multi  | Multi  | 91.61 |
| XLMR     | IPS  | Single | None  | 87.36 |
| XLMR     | IPS  | Single | Single | 89.36 |
| XLMR     | IPS  | Single | Multi  | 90.36 |
| XLMR     | IPS  | Multi  | None  | 89.14 |
| XLMR     | IPS  | Multi  | Single | 89.36 |
| XLMR     | NSD  | Single | None  | 89.11 |
| XLMR     | NSD  | Single | Single | 91.05 |
| XLMR     | NSD  | Single | Multi  | 90.94 |
| XLMR     | NSD  | Multi  | None  | 90.55 |
| XLMR     | NSD  | Multi  | Single | 89.87 |
| XLMR     | NSD  | Multi  | Multi  | 89.81 |
Figure 4: Module-wise comparative analysis. We run each setting three times and average accuracies are reported.

Table 2: Analysis on the number of valid fields. The matching accuracy becomes higher as the number of valid fields grows.

| Valid Fields                  | Acc.  |
|------------------------------|-------|
| Store Name                   | 33.66 |
| + Address                    | 81.40 |
| + Telephone Number           | 91.02 |
| + Business Number            | 91.61 |

Table 3: Matching accuracies of the daily queries from the real users in POI Match. The proposed matching algorithm outperforms the heavily engineered baseline method by 8% point of top-1 matching accuracy.

| Models                                  | Acc.  |
|-----------------------------------------|-------|
| Baseline Model                          | 67.0  |
| Proposed (MBERT-NSD-Multi-Multi)       | 75.3  |

5 Concluding Remarks and Future Work

In this paper, we propose a new type of grounding system for querying large-scale document-oriented databases with semi-structured natural language. The proposed system utilizes embedding-based retrieval to alleviate several practical concerns in the semi-structured query grounding problem. We examined the proposed system on our application POI Match which aims to find the corresponding POI entry for a user input receipt image. Despite various OCR, Parsing, and DB noises, the proposed method successfully matches the receipt image with the corresponding DB entry. In our experiments, the proposed model significantly outperforms the heavily engineered baseline model that has been used in our product for the last few years while requiring much less development and maintenance cost.

As future work, testing the proposed grounding pipeline on different applications or domains would help to get a general understanding of each module. Investigating the connections between some recent theoretical analyses on Siamese neural-network-based retrieval and our empirical results would also be interesting. We believe our findings in this work can easily be extended to other real-world applications that are dependent on a similar problem.

Acknowledgements

The authors would like to thank anonymous reviewers for their insightful comments.

References

Arora, R.; and Aggarwal, R. R. 2013. Modeling and querying data in mongodb. *International Journal of Scientific and Engineering Research, 4*(7): 141–144.

Baek, J.; Kim, G.; Lee, J.; Park, S.; Han, D.; Yun, S.; Oh, S. J.; and Lee, H. 2019a. What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis. In *International Conference on Computer Vision (ICCV)*.

Baek, Y.; Lee, B.; Han, D.; Yun, S.; and Lee, H. 2019b. Character Region Awareness for Text Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9365–9374.

Bhardwaj, N. 2016. Comparative Study of CouchDB and MongoDB – NoSQL Document Oriented Databases. *International Journal of Computer Applications*, 136: 24–26.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics, 5*: 135–146.

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1994. Signature Verification using a "Siamese" Time Delay Neural Network. In Cowan, J.; Tesauro, G.; and Alspetector, J., eds., *Advances in Neural Information Processing Systems*, volume 6, 737–744. Morgan-Kaufmann.

Cohan, A.; Beltagy, I.; King, D.; Dalvi, B.; and Weld, D. 2019. Pretrained Language Models for Sequential Sentence Classification. 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)*, 3693–3699. Hong Kong, China: Association for Computational Linguistics.

Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grae, E.; Ott, M.; Zettlemoyer,
and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32, 8026–8037. Curran Associates, Inc.

Pires, T.; Schlinger, E.; and Garrette, D. 2019. How Multilingual is Multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4996–5001. Florence, Italy: Association for Computational Linguistics.

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. Hong Kong, China: Association for Computational Linguistics.

Rigaud, C.; Doucet, A.; Coustaty, M.; and Moreux, J.-P. 2019. ICDAR 2019 Competition on Post-OCR Text Correction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 1588–1593.

Robertson, S.; and Zaragoza, H. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Tan, S.; Zhou, Z.; Xu, Z.; and Li, P. 2019. On Efficient Retrieval of Top Similarity Vectors. 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)*, 5236–5246. Hong Kong, China: Association for Computational Linguistics.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.