Literature Review Paper

--Schema Matching Using Pre-Trained Language Models

Overall

The paper proposes a method called LSM (Learned Schema Mapper) that leverages the natural language understanding capabilities of pre-trained language models to improve the accuracy of data-free schema matching. LSM incorporates active learning and an intelligent attribute selection strategy to obtain precise user feedback, thereby reducing human labelling costs. Experimental results demonstrate LSM's effectiveness, as it outperforms the best baseline approach by requiring fewer labels and reducing reviewing costs. The paper also conducts an ablation study, revealing the substantial contribution of the BERT featurizer to LSM's overall performance.

The motivation in detail, the paper evaluates the effectiveness of state-of-the-art schema matching approaches that rely solely on schema-level information, due to the unavailability of customer data. The paper identifies that these approaches have poor accuracy on real customer schemata (through evaluation results showing significant drops in performance, specifically with these methods struggling to accurately match attributes between the customer data and a target schema), highlighting the need for a new approach. Four representative heuristic-based methods (CUPID, COMA, S- MATCH, and Similarity Flooding) and two machine learning-based methods (LSD and MLM) are selected for evaluation. The baseline methods are adapted to use schema-level information only. Results show that these approaches have near-perfect accuracy on synthetic and publicly available schemata but significantly drop in accuracy on real-world customer schemata. The challenges in matching real-world customer schemata include variations in attribute naming conventions and a larger target schema size. These challenges motivate the need for a new approach that combines reasonably accurate models with human-in-the-loop intervention and leverages the natural language understanding capabilities of pre-trained language models. Moreover, the large size of the Information System Schema (ISS) provides an opportunity to leverage pre-training techniques to better understand the domain. These insights guide the design of a novel linguistic schema matching approach based on semi-supervised and active learning, which is discussed in the following section.

The Learned Schema Matcher (LSM) is a linguistic schema matching approach that addresses the shortcomings of prior work. It uses semi-supervised and active learning to iteratively match attributes in a source schema to a target schema. The LSM matching pipeline consists of several phases:

1. Preparation: Generating a set of candidate pairs by calculating the Cartesian product between attribute sets in the source and target schemas. Candidate pairs start as unlabeled.
2. Featurization: Converting candidate pairs into numerical vectors using different featurizers such as word embedding, lexical, and a fine-tuned BERT featurizer. The

BERT featurizer incorporates a pre-trained language model to capture linguistic similarities.

3. Model Training and Prediction: Training a meta-learning model using partially labeled data and using it to predict matching labels for unlabeled attributes. The model outputs matching scores that are further tuned based on data type compatibility and penalization for introducing new entities.
4. User Interaction: Providing the user with matching suggestions for unlabeled attributes. The user reviews the suggestions and either marks correct matches or indicates that none of the suggestions are correct. The user also labels a set of incorrectly matched attributes selected by LSM. LSM selects attributes to be labeled using a least confident anchor strategy, which prioritizes the most "informative" attributes in the schema.

   Here's a breakdown:

   a. Anchor Attributes: These are considered the most "informative" attributes in the schema. In LSM, the anchor set usually consists of primary keys (PKs) and foreign keys (FKs) as they carry crucial information regarding schema relationships.
   b. Least Confidence Selection: Among the unlabeled anchor attributes, the LSM system selects a subset of N attributes to be labeled by employing the "least confidence" strategy. This strategy determines the N attributes with the least prediction confidence based on the Softmax function applied to the matching scores.
   c. Calculation of Prediction Confidence: The prediction confidence (cs) for each unlabeled anchor attribute is calculated using Softmax on the matching scores generated by the model. This score represents how uncertain or confident the system is about the correctness of a suggested match for a particular attribute.
   d. User Labeling: The system presents the selected subset of attributes to the user and asks for correct mappings to the target schema. Users then provide these mappings. For each correct match (as, at), the label for that pair is marked as 1, while other potential matches (as, a't) are marked as -1.
5. Updating Labels: User feedback is used to update the labels of candidate pairs before the next iteration starts.

LSM aims to improve schema matching accuracy by leveraging linguistic features, active learning, and user feedback to iteratively refine the matching process.

EXPERIMENTAL EVALUATION

In this part the author wants to evaluate the quality of LSM on both customer and publicly available schemata and perform an indepth analysis of the results.

Goals mainly want to answer the following questions:

1. What is the prediction quality of LSM model? (Section V-B)
2. What is the human labeling cost to map the full source schema to the ISS? (Section V-C)
3. What is the contribution of the BERT featurizer and the attribute descriptions to the overall performance? (Section V-D)
4. How is the performance of LSM affected in the presence of noise? (Section V-F)

5. How much time is spent re-training the model after the user provides new labels? (Section V-G).

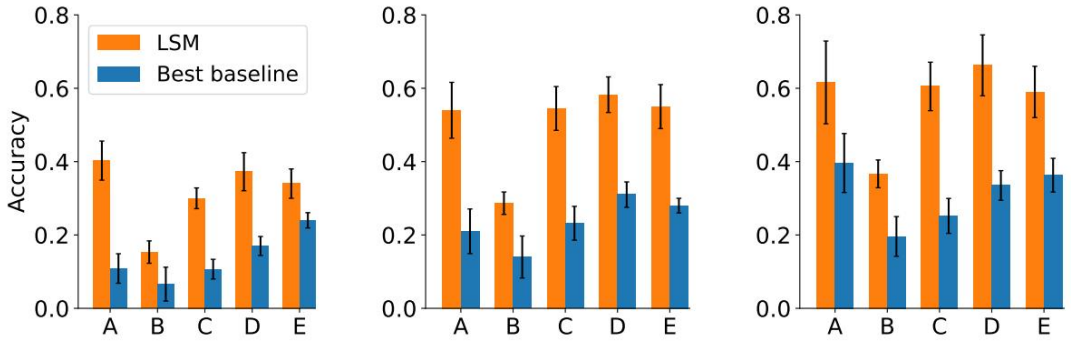
The experimental conclusion:

1. Prediction quality:

In the non-interactive experiment tested on Public Schemata and Customer Schemata, the result of LSM accurate rates is better than the baseline on different database (RDB-Star, IPFQR, MovieLens-IMDB) and on the customer's schema (Table IV).

Table IV: Top-k accuracy of LSM on the public schemata.

	Best Baseline			LSM		
	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
RDB-Star	0.95	1.00	1.00	0.98	0.98	1.00
IPFQR	1.00	1.00	1.00	1.00	1.00	1.00
MovieLens-IMDB	0.53	0.72	0.75	0.65	0.83	0.87



(a) Top-1 (b) Top-3 (c) Top-5

Fig. 4: Top-k accuracy of LSM vs the best base line on customer schemata A-E

For example, the fig.4 the figure illustrate that observation results indicate that for all modes, LSM outperforms the optimal baseline in terms of Top-1, Top-3, and Top-5 accuracy. In addition, as the proportion of labels increases, the accuracy of both methods shows an upward trend, but the increase in LSM is greater, further widening the gap with the optimal baseline. LSM can achieve high matching accuracy when only a small amount of manual labels are used. In contrast, the best baseline method performs poorly when dealing with these patterns, requiring more labels to achieve similar accuracy. Therefore, we can conclude that LSM outperforms the best baseline method in terms of Top-k accuracy in customer mode A-E.

2. Human labeling cost

Correctly match around 70% of the customer schema attributes, and using the LSM method smart selection strategy, our model outperforms not only the best baseline but also the random selection strategy by reducing the total labels required by up to 11% (Fig.5 given).

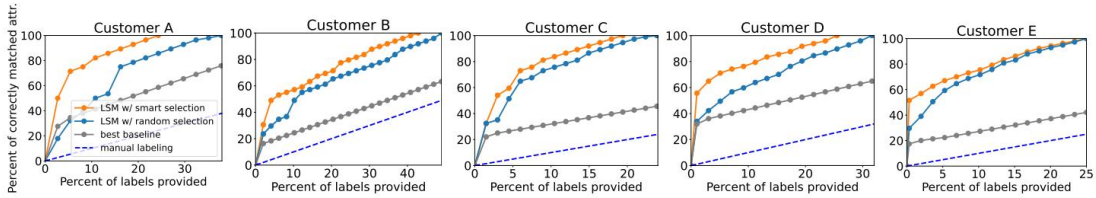


Fig. 5: Percentage of the attributes correctly matched vs. percentage of human labels provided.

3. BERT featurizer contribute overall performance

Figure 6 shows the results of ablation experiments on BERT feature extractors on different customer modes to see the correct matching percentage for each attribute description. The experimental results show that using the BERT feature extractor can significantly improve the accuracy of attribute matching. In all customer patterns, using the BERT feature extractor improved the correct matching rate by an average of 11%. This indicates that the BERT feature extractor is a key component for our model.

So , in this paper , the BERT featurizer is a critical component of LSM model. Inthe figure, disable the BERT featurizer in our model (denoted as LSM w/o BERT), the user may need to provide up to 17% more labels (Customer B) to map their full schema to ISS.

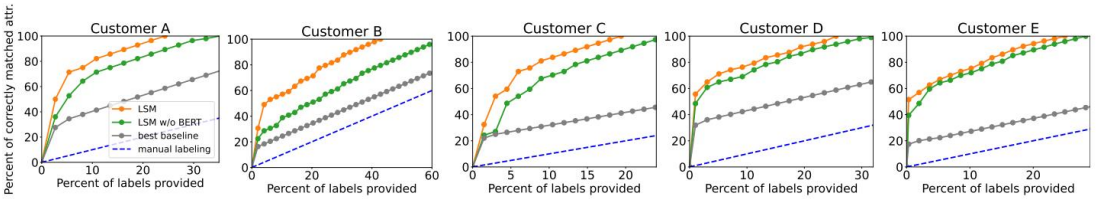


Fig. 6: Ablation study on the BERT Featurizer using various customer schemata.

4. LSM performance affect by noise

Figure 8 shows the performance of LSM under different noise rates. Specifically, Figure 8 shows the percentage of LSM correctly matching attributes when the noise rates are 0 (raw LSM), 0.1, 0.2, and 0.3.

From the graph, it can be observed that as the noise rate increases, the number of attributes correctly matched by LSM decreases. Specifically, when the noise rate is 0.1, the number of correctly matched attributes in LSM decreases to 90%. When the noise rate is 0.2, the number of correctly matched attributes decreases to 80%. When the noise rate is 0.3, the number of correctly matched attributes decreases to 70%. These data points are represented by dashed lines.

However, even at a noise rate of 0.3, LSM still outperforms the best baseline model (represented by a solid line). It should be noted that these results were obtained using the BERT Featurizer.

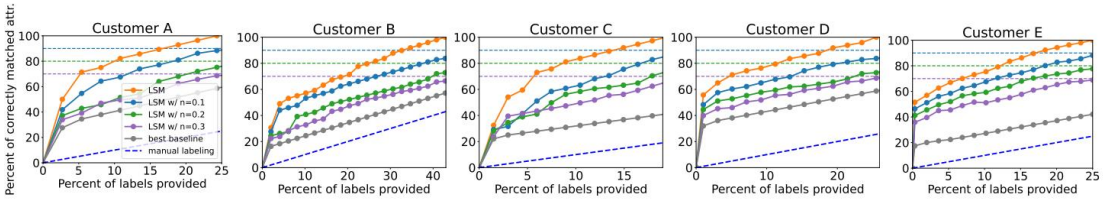


Fig. 8: Performance of LSM in the presence of noise with varying noise rates n.

5. LSM re-training response time

Figure 9 shows the response time of LSM as the number of matching labels varies. The experiment was run on an Azure cloud server with 8 core CPUs, 112 GiB of memory, and a Tesla P100.

In Figure 9, we can see that as the number of tags increases, the response time also increases accordingly. However, it is worth noting that even with an increase in the number of tags, the response time of LSM is still relatively fast, and in most cases, the response time is less than 10 second. This indicates that our model has good performance and efficiency when processing large amounts of data.

In addition, we also noticed that the response time is more affected by the number of candidate pairs than by the number of labels. This is because we used both labeled and unlabeled examples for training in semi supervised settings, so the number of candidate pairs directly affects the complexity of the training process.

In summary, LSM has good response time and the ability to process large amounts of data.

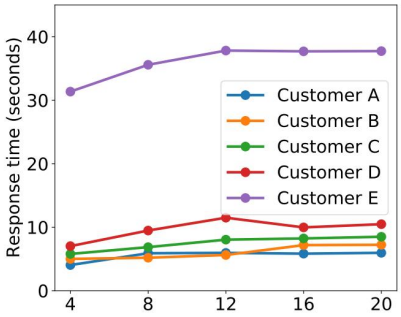


Fig. 9: Response time of the LSM as the number of labels is varied.

VI. Discussion

Then writers discuss various aspects related to the design and limitations of the approach used in the study.

1. Tuning Hyper-parameters: The paragraph highlights the challenges in tuning hyper- parameters and the schema-specific nature of optimal hyper-parameter values. It emphasizes the importance of having a limited number of components to tune for usability and performance.
2. ML in Schema Matching: The paragraph discusses the issue of overfitting due to limited labelled data and the risk of creating a model that is too focused on provided matching patterns. It explains the use of an active learning approach and a light-weight model with a semi-supervised framework to address this problem.

Additionally, the pre-training of a BERT featurizer on the ISS is mentioned as a way to better understand the domain.

3. Response Time: The paragraph acknowledges the challenge of achieving good response time when matching large schemata. It mentions previous approaches like partitioning and parallelization to address this problem. While the proposed model provides efficient ranking of matching pairs, the response time still increases with the number of input attributes. However, considering the manual matching alternative, this cost is deemed acceptable.
4. Design Space: The paragraph discusses alternative designs, such as using large language models with few-shot prompting or performing domain-specific pre-training. The chosen approach is to fine-tune a smaller model like BERT due to the availability of enough data. The other approaches are considered for future exploration.

Limitations of the LSM: The paragraph mentions that the LSM heavily relies on schema-only information for matching and does not address problems where the user has no control or understanding of their schema, such as when it is generated by a third-party application. Data access would likely be required to solve such problems.

Although the Learned Scheme Matcher algorithm performs well in certain aspects, there are still shortcomings in some aspects.

Firstly, the paper did not fully consider the semantic matching problem between the source data and the target data. In reality, due to the different semantics of different data sources, even pattern matching cannot guarantee data consistency. Therefore, future research can consider incorporating semantic understanding into pattern matching to better address this issue. In the original research paper[1], BERT (Bidirectional Encoder Representations) is a new language representation model obtained from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT aims to train deep bidirectional representations in all layers through joint context. Therefore, a finely tuned BERT model only requires an additional output layer to create states that perform well on a wider range of tasks. But the Learn Schema matcher may attempt to consider semantic understanding about the IIS thus the methods may prove the universality.

Secondly, the learned schema matcher algorithm proposed in this paper may not be effective enough in handling complex mapping transformations. When the mapping relationship between source data and target data is very complex, the algorithm (candidate pairs (a_s, a_t), i.e. $c_s = \max_{a_t \in k_s} score(a_s, a_t)$) may experience performance degradation. Therefore, future research can attempt to improve algorithms to better handle complex mapping transformations.

Thirdly, in the user interaction process, the Least Confident Anchor method also have some deficiencies:

1. Ignoring contextual information: The minimum confidence anchor strategy only considers the prediction results of a single attribute, while ignoring the contextual information between attributes. This may result in the inability to find the true anchor attribute in some cases.

2. Dependence on data quality: The minimum confidence anchor strategy requires high data quality. If there is noise or error in the data, the strategy may be affected, resulting in inaccurate selection of anchor attributes.
3. The handling of unknown attributes: The minimum confidence anchor strategy mainly analyzes the prediction results of known five customer cases attributes. However, in some unknown cases, we may need to handle unknown attributes or categories. At this point, the strategy may not having the ideal results.

Looking forward to the future, the LSM method could utilize more efficiency. Because, there also some more confident to realize the advantages of fine-tuning pre trained language models using BERT [2] including:

1. Quick adaptation to LSM tasks: As pre-trained language models have already learned a large amount of language knowledge and contextual information, they can quickly adapt to LSM tasks without the need to start training from scratch.
2. Improving generalization ability: Pre-trained language models can generalize to various tasks and domains because they learn common language structures and knowledge. Data augmentation improves model performance by generating additional training data.
3. Reduce data requirements: As pre-trained language models have already learned a large amount of language knowledge, they can be fine-tuned on smaller annotated datasets, thereby reducing data requirements.
4. Improving model performance: By fine-tuning the pre-trained language model, it can be optimized for specific tasks, thereby improving model performance. Fine-tuning pre-trained language models can accelerate model training, improve generalization ability, reduce data requirements, and improve model performance.

In conclusion, in LSM application scenarios, continuous optimization of algorithms can definitely provide more efficient automated schema matching tools.

Reference:

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.

[2] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, and etc. “Deep Entity Matching with Pre-Trained Language Models”, arXiv preprint arXiv: 2004.00584, 2020.