TableQA: Question Answering on Tabular Data

Svitlana Vakulenko svitlana.vakulenko@wu.ac.at
Vienna University of Economics and Business

Vadim Savenkov vadim.savenkov@wu.ac.at
Vienna University of Economics and Business

ACM Subject Categories
Information retrieval, Retrieval tasks and goals, Question answering

Abstract

Tabular data is difficult to analyze and to search through, yielding for new tools and interfaces that would allow even non tech-savvy users to gain insights from open datasets without resorting to specialized data analysis tools and without having to fully understand the dataset structure. The goal of our demonstration is to showcase answering natural language questions from tabular data, and to discuss related system configuration and model training aspects. Our prototype is publicly available and open-sourced (see demo)

1. Introduction

There is an abundance of tabular data on the web in the form of Open Data tables, which are regularly released by many national governments. Providing their data free of charge, publishing bodies seldom have dedicated resources to support the end users in finding and using it. In many open data portals the search facility remains limited: e.g., no search in the content of data tables is supported.

We attempt to remedy this situation through development of the information retrieval tools tailored specifically to the end users without technical background. Our Open Data Assistant chatbot [3] offers an unconventional interface for cross-lingual data search via Facebook and Skype messaging applications enabling a quick overview of the available datasets collected from various open data portals. However, the current version of the chatbot supports only metadata-based search. In this paper, we work towards extending the chatbot to search within the content of open data tables and answering specific user questions using the values from these tables.

2. Related Work

The task of question answering (QA) from tables is closely related to QA from databases and knowledge graphs (KGs). The major difference, however, is that the open data tables are produced by different authors and do not adhere to a single schema (schema mismatch), i.e. they require disambiguation and integration. The task of QA on tables can be translated to one of the other two tasks via a two-step pipeline process: 1) mapping tables to a database (or a KG, respectively); 2) solving QA task from database (KG). Both of the steps in this pipeline are error-prone, and the errors produced at the first stage, encountered during tables integration, will naturally propagate to the next stage (QA). We are focusing on exploring ways to develop an end-to-end solution that would solve both tasks jointly and will operate on the raw tables as input, e.g. CSV files, which also allows to track the answer provenance in a transparent manner.

Recently, quite a few studies emerged that address the question-answering task on tables using deep neural networks. They involve search across tables [5] and learning to perform aggregation operations [7] [2] . However, all of the proposed systems are very complex, require significant computation resources and are engineered to work exclusively on tabular data.

We contribute to the growing body of work on question answering for tabular data by providing and evaluating a prototype based on the End-To-End Memory Networks architecture [4] . This architecture was originally designed for the question-answering tasks from short natural language texts (bAbI tasks) [6] , which include testing elements of inductive and deductive reasoning, co-reference resolution and time manipulation. In this context the task of question answering over tables can be seen as an extension to the original bAbI tasks. It is very appealing to be able to apply the same type of architecture to querying semi-structured tables alongside the textual data for this could enable question answering on real-world documents that contain a mixture of both, e.g., user manuals and financial reports.

3. Task Description

The task of question answering over tables is given an input table (or a set of tables) *T* and a natural language question *Q* (a user query), output the correct answer *A* .

4. Architecture

The architecture of our system for table-based question answering is summarized in Figure Figure 1 . Each of the individual components is described in further details below.

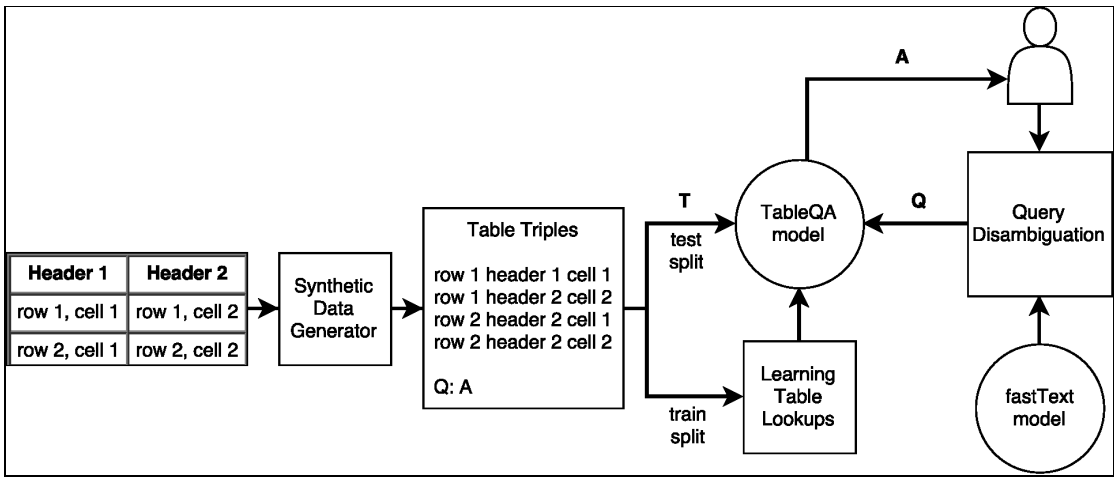

Figure 1. System architecture. T - input table; Q - question; A - answer.

4.1. Table Representation

Training examples consist of the input table decomposed into row-column-value triples and a question/answer pair, for instance:

Row1	City	Klagenfurt
Row1	Immigration	110
Row1	Emigration	140
Row2	City	Salzburg
Row2	Immigration	170
Row2	Emigration	100

Question : What is the immigration in Salzburg?
Answer : 170

This representation preserves the row and column identifiers of the table values. In this way our system can

also ingest and learn from multiple tables at once.

4.2. Learning Table Lookups

Our method for question answering from tables is based on the End-To-End Memory Network architecture [4] , which we employ to transform the natural-language questions into the table lookups. Memory Network is a recurrent neural network (RNN) trained to predict the correct answer by combining continuous representations of an input table and a question. It consists of a sequence of memory layers (3 layers in our experiments) that allow to go over the content of the input table several times and perform reasoning in multiple steps.

The data samples for training and testing are fed in batches (batch size is 32 in our experiments). Each of the data samples consists of the input table, a question and the correct answer that corresponds to one of the cells in the input table.

The input tables, questions and answers are embedded into a vector space using a bag-of-words models, which neglects the ordering of words. We found this approach efficient to work on our training data, since the vocabulary for column headers and cell values are disjoint. In the future work we consider also evaluating the added value of switching to the positional encoding on the real world data as reported in [4] . The output layer generates the predicted answer to the input question and is implemented as a softmax function in the size of the vocabulary, i.e. it outputs the probability distribution over all possible answers, which could be any of the table cells.

The network is trained using stochastic gradient descent with linear start to avoid the local minima as in [4] . The objective function is to minimize the cross-entropy loss between the predicted answer and the true answer from the training set.

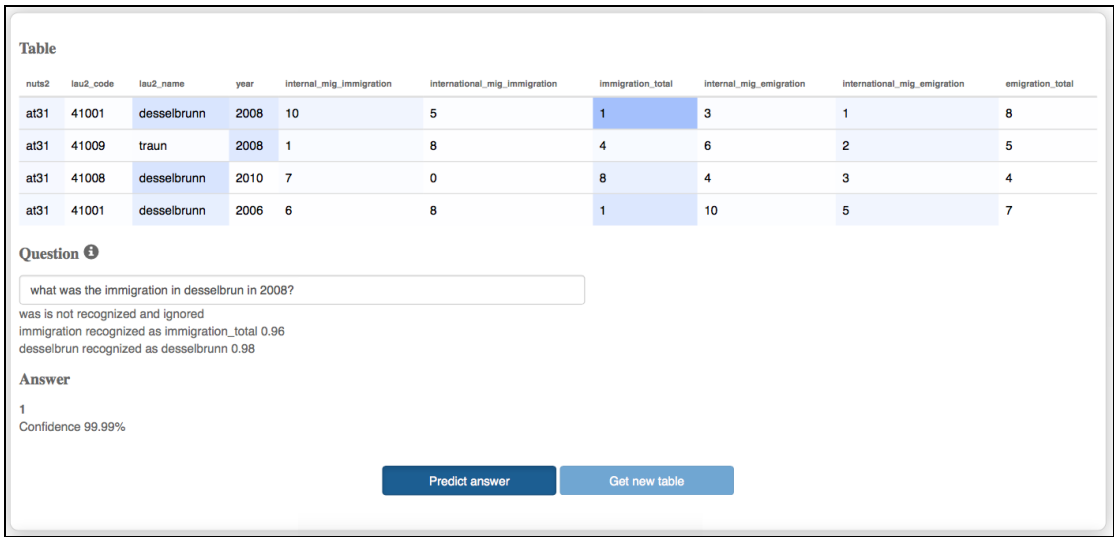

Figure 2. Demonstration of the neural network model trained towards question answering from tables. The highlighted cells and their intensity indicate the attention weights over the input table that were activated to predict the answer to the input question.

4.3. Query Disambiguation

Since users may refer to the columns with words that differ from the labels used in the table headings, we employ a fastText model [1] pretrained on Wikipedia to compute similarity between the out-of-vocabulary (OOV) words from the user query and the words in our vocabulary, i.e. to align or ground the query in the local representation. The similarity is computed as a cosine-similarity between the word vectors embedded using the pretrained fastText model.

fastText provides continuous word representation, which reflects semantic similarity using both the word co-occurrence statistics and the sub-word-based similarity via the character n-grams. For each of the OOV words

the query disambiguation module picks the most similar word from the vocabulary at query time and uses its embedding instead.

In our scenario this approach is particularly useful to match the paraphrases of the column headings, e.g., the word emigration is matched to the emigration_total label. We empirically learned the similarity threshold of 0.8 that provides optimal precision/recall trade-off on our data.

5. Demonstration

The aim of the demonstration is to showcase the power and limitations of the neural model trained to answer questions on semi-structured data. TableQA prototype is implemented as a Flask web application [1] and is publicly available on our web-site (see demo). The user interface (Figure Figure 2) allows to enter a custom question for a sample table provided (alternatively, use one of the questions from the test set held-out during the training phase). The attention weights are visualized by highlighting the corresponding cells in the input table, which provides an insight on the data patterns learned by the neural network. There is also an additional table below, which contains more details about the underlying prediction mechanism. It contains the triple-wise representation of the input table as consumed by the neural network and the attention weights for each of the three memory layers separately.

6. Conclusion

In this paper we propose two new bAbI tasks for question answering from tables and provide an initial evaluation of the performance of the memory network architecture on them. These results can be used towards developing a natural-language interface that will support search in semi-structured data, such as Open Data tables. The role of the demonstration is to provide an opportunity to interactively explore the performance and limitations of the trained model. It helps to understand which patterns the model has actually learned from the provided data samples. This tool will be useful for all who want to learn more about this family of models as well as for the researchers looking for directions to improve the neural network performance. In the future work the initial approach proposed in this paper need to be further extended to scale on the real world open data repositories by handling OOV words, negative sampling, cross-table retrieval and QA from both tables and text simultaneously.

Acknowledgments

This work was supported by the Austrian Research Promotion Agency (FFG) under the project CommuniData (grant no. 855407). The work of Svitlana Vakulenko has received funding from the EU H2020 programme under the MSCA-RISE agreement 645751 (RISE_BPM).

References

1. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606. https://arxiv.org/abs/1607.04606.pdf
2. Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2016. Learning a natural language interface with neural programmer. CoRR abs/1611.08945. http://arxiv.org/abs/1611.08945
3. Sebastian Neumaier, Vadim Savenkov, and Svitlana Vakulenko. 2017. Talking open data. In 14th Extended Semantic Web Conference, ESWC 2017, Portoroz, Slovenia, May 28 - June 1, 2017. https://arxiv.org/abs/1705.00894.pdf
4. Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7- 12, 2015, Montreal, Quebec, Canada. pages 2440– 2448. http://papers.nips.cc/paper/5846-end-to-end-memory-networks.

5. Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016. pages 771– 782. https://doi.org/10.1145/2872427.2883080.
6. Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. CoRR abs/1502.05698. http://arxiv.org/abs/1502.05698.
7. Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural enquirer: Learning to query tables in natural language. In Proceedings of the Twenty-Fifth International Joint Conference on Arti cial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016. pages 2308–2314. http://www.ijcai.org/Abstract/16/329.

Footnotes

1 Implementation based on MemN2N-babi-python [back]