Machine Learning for Question Answering from Tabular Data

Mahboob Alam Khalid Valentin Jijkoun Maarten de Rijke
ISLA, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{mahboob,jijkoun,mdr}@science.uva.nl

Abstract

Question Answering (QA) systems automatically answer natural language questions in a human-like manner. One of the practical approaches to open domain QA consists in extracting facts from free text offline and using a lookup mechanism when answering user's questions online. This approach is related to natural language interfaces to databases (NLIDBs) that were studied extensively from the 1970s to the 1990s. NLIDB systems employed a range of techniques, from simple pattern-matching rules to formal logical calculi such as the lambda calculus, but most were restricted to specific domains. In this paper we describe a machine learning approach to querying tabular data for QA which is not restricted to specific domains. Our approach consists of two steps: for an incoming question, we first use a classifier to identify appropriate tables and columns in a structured database, and then employ a free-text retrieval to look up answers. The system uses part-of-speech tagging, named-entity normalization and a statistical classifier trained on data from the TREC QA task. With the TREC QA data, our system is shown to significantly outperform an existing rule-based table lookup method.

1 Introduction

Automatic Question Answering (QA) systems return answers—not documents—in response to a user's query. One special type of QA systems extensively studied during the 1970s–1990s comprised Natural Language Interfaces to Databases (NLIDB), which used structured databases (DBs) as the information source and aimed at hiding complex database query languages from the user; see [5] for an overview.

Since the late 1990s, and motivated by the TREC QA track [20], the attention of the QA community has changed to include access to open domain text collections (such as newspapers). The canonical architecture of many open domain textual QA systems consists of a three-stage pipeline: question analysis, document retrieval, and answer extraction [16]. There are several approaches to implementing these stages. We focus on an offline approach [12, 9, 3], where facts corresponding to commonly occurring question types are extracted and stored in a databases for lookup at question time. The main advantage of this approach is that by moving the expensive text analysis and information extraction stages offline (to the table creation phase), systems can achieve good run-time behavior even with a large amount of information. In this paper we focus on the lookup stage of this approach to QA. Our system uses a knowledge base consisting of several tables. Answering a user's question consists in mapping the question to a knowledge base query, executing the query, and presenting the results to the user. Specifically, we map an incoming question to an SQL-like query "select AF from T where $sim(QF, Q)$," where T is the table that contains the answer candidate in field AF and its other field QF has a high similarity with the input question Q.

Similarly to NLIDB systems, the main issue for our approach to QA is *question analysis*, i.e., the task of identifying for a given question which tables and fields in the knowledge base should be searched and from which fields answer candidates should be extracted. In our query formalism, the task consists in mapping an incoming question Q to a tuple $\langle T, QF, AF\rangle$ (a *table lookup label*) and defining and efficiently implementing the similarity function $sim(QF, Q)$. We view the generation of table lookup labels as a classification task and apply a standard machine learning approach to it. Furthermore, we implement $sim(QF, Q)$ using keyword-based relevance functions from Information Retrieval. In this paper we report on experiments with the classification, with different retrieval models and text normalization methods, including named entity normalization.

Unlike NLIDB systems, in the context of open-domain QA we have to deal with noisy data that has usually been obtained automatically through information extraction tools, from different sources, and that has some structure (relational tables) with unknown semantics. For a real-world open domain QA system it is important to integrate as many information sources (tables and databases in our

1529-4188/07 $25.00 © 2007 IEEE
DOI 10.1109/DEXA.2007.119

case) as possible and to keep the information up-to-date, so the ability to handle the data as a black box, without imposing strict semantics, is crucial. Our main contribution is the description of a system that addresses these issues.

The rest of the paper is structured as follows. Section 2 describes related work. Section 3 introduces our approach. Sections 4 and Section 5 describe the two learning stages: selecting the best retrieval method and training the query generator. Section 6 presents an extrinsic evaluation on TREC questions, and we conclude in Section 7.

2 Related work

Machine learning is used by many QA systems to extend the standard pipeline architecture in different ways. E.g., [18] describes a QA system where passages identified by an information retrieval engine are re-ranked by a machine learning component trained of a corpus of questions and answers to classify passages for "answerhood." Machine learning is also applied to question classification, often understood as identification of the expected answer type for a question. E.g., [14] represents a question with syntactic and semantic features and applies a supervised SVM classifier to the question classification task; [13] presents a cascaded classifier and describes a training corpus of 5,500 questions manually annotated with expected answer types.

There has also been much research in defining the similarity or relevance functions for short text segments (e.g., questions and answer sentences). E.g., [4] presents an in-depth exploration of the TREC Novelty task [10], i.e., identifying relevant and novel sentences in a ranked list of relevant documents. In [6], different query expansion methods are compared; [17] uses WordNet to disambiguate between different word senses in order to assign appropriate sense to each query term for query expansion.

3 Our approach

Our QA system uses a set of databases as the source of answers. For our experiments we use tables automatically extracted from the TREC QA corpus [20] by the information extraction module of an open domain QA system, QUARTZ [1]. There are 23 tables containing 8M rows (403MB of text) in total. For example, the *Roles(role,name)* table contains role of *George Bush* as *United States president*, *Birthdays(name, birthdate)* contains *1962* as the birthdate of *Tom Cruise*. Our system views the content of the database as a black box and has no prior knowledge of the semantics of the table and field names.

When a user posts a question, we first extract features and apply a statistical classifier that assigns a *table-lookup label*, i.e., a tuple $\langle T, QF, AF \rangle$. Then we translate the question into a retrieval query which is run against an index that contains values of all fields of all rows in our database as separate documents. We restrict the retrieval engine so that it only considers values in the field QF of the table T predicted by the classifier. We take the top n field values returned by the IR engine and return the values of the AF fields from the corresponding rows of T as the top n answer candidates. Figure 1 shows the architecture of our system.

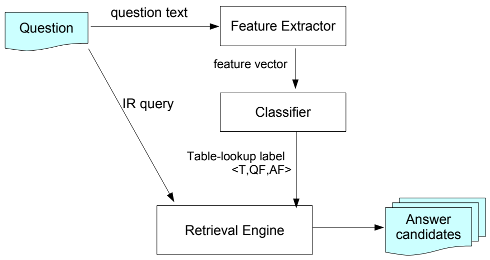

Figure 1. QA from tabular data

Our architecture depends on two modules: the classifier that predicts table lookup labels and the retrieval model along with the text representation and the retrieval query formulation. We use a corpus of questions with correct answers available from TREC[1] to determine the best query formulation and retrieval models, and to train a statistical classifier. As to the former task, we apply several retrieval models, text representations and query formulation methods to the TREC questions and our database of facts. For each candidate retrieval method M and each question q, we find the most relevant field values in all rows of all tables. We then check whether other fields of the rows contain the correct answer. If so, we consider the question q as correctly answered. We select the retrieval method that allows our system to answer the largest number of questions.

At the second stage, we use the selected retrieval method to generate training data for the classifier. For each q we have a ranked list of field values, each value is associated with a table and a field (T and QF). For some field values, the answer can be found in field AF of the same table T. We select the top most label $\langle T, QF, AF \rangle$ such that the table T and the field QF were first encountered. In other words, for each q we find labels such that the query "select AF from T where $sim(Q, QF)$" returns the correct answer at the top rank. Questions for which such labels are found form the training set for our question classifier.

The next two sections we describe the stages in detail.

4 Selecting the retrieval method

As described above, in order to select the best retrieval method for our QA system, we use a collection of ques-

[1] http://trec.nist.gov/data/qa.html

syn♯1. shrub, bush (a low woody planet)
syn♯2. bush (a large wilderness area)
syn♯3. scrub, bush
syn♯4. public hair, bush, crotch hair
syn♯5. George Walker Bush, (43rd Presidentof United States)
syn♯6. George H.W. Bush, (41st President of United States)
syn♯7. Vannevar Bush (United States electrical engineer)

Figure 2. WordNet synsets of noun "Bush".

tions with known correct answers. For each IR method M, we count the number of questions for which at least one of the retrieval results at rank $\leq k$ ($k = 1, 10, 100$) is a field value from a table row that contains the correct answer in some other field. When selecting a retrieval method for our QA system, we are faced with several choices: (1) how to represent the text of documents, in our case, field values, (2) how to formulate queries for the retrieval, and (3) which text retrieval model to use. We consider two choices for the retrieval model: a standard vector space model with TF.IDF weighting (we use the implementation of [2]) and a statistical language model [11]. For the representation of queries and documents, we consider standard stemming and named entity normalization as possible choices.

Named entities like "U.S." and "United States" are difficult for keyword-based retrieval systems because they are semantically close, but have no common terms. To address this problem, [15] proposed a query expansion technique. They expanded short text segments by posting them against a commercial search engine's index and used the top 200 titles and snippets as expanded representation.

We apply a different approach to reducing the semantic gap: we normalize named entities (NEs) to canonical forms. We use WordNet synsets[2] as canonical forms of NEs. E.g., "U.S." and "United States" belong to the same WordNet synset and thus would become identical after normalization. One problem with this approach is ambiguity: a single string of characters can be used to refer to different entities. In WordNet, some terms belong to multiple synsets; e.g., "Bush" has seven synsets as a noun, three as a proper noun and four as a common noun (Figure 2).

To address the ambiguity problem, we exclude synsets that contain non-capitalized entries and apply a simple heuristic to disambiguate between the remaining synsets, whenever an ambiguity arises. If a named entity belongs to multiple synsets, we pick the synset with the highest frequency in a reference corpus,[3] calculated as:

$$\mathsf{score}(synset) = \sum_{ne_i \in synset} \mathsf{count}(ne_i) \qquad (1)$$

If a named entity cannot be normalized using WordNet in this way (as WordNet does not cover all names), the terms of the entity are stemmed and left as-is. Since stemming can also be applied to common words, we consider three versions of text normalization: stemming, NE normalization, and both. See Figure 3 for an example.

exact	New York's Triangle Shirtwaist factory
stemmed	new york triangl shirtwaist factori
normalized	WnEn21n432 's Triangle Shirtwaist factory
stemmed & norm.	WnEn21n432 triangl shirtwaist factori

Figure 3. Different text representations

expansion	**Top-1**		**Top-10**		**Top-100**	
	lm	**vsm**	**lm**	**vsm**	**lm**	**vsm**
exact	232	244	618	607	968	945
normalized	231	232	554	568	935	905
stemmed	259	257	645	650	986	994
stem&norm	250	244	587	601	960	957
exact+norm	252	249	638	596	985	950
exact+stemmed	274	259	**682**	643	1018	993
exact+stem&norm	**275**	258	669	621	**1023**	982

Table 1. The number of questions answered at ranks 1, 10 and 100 using different text representations and retrieval models (see text). Best scores in boldface.

In order to select the best retrieval method, we evaluated the performance of different methods on the TREC QA dataset. We consider the four text representations mentioned previously as well as their combinations, and experimented with vector space retrieval and language modeling-based retrieval. When combining representations for retrieval, we sum the relevance scores of individual representations: e.g., for the combined *exact+stem&norm* representation, we sum the relevance scores obtained with the *exact* and *stemed&norm* representations. Table 1 shows the evaluation results on the set of 2,136 TREC QA questions with correct answers. We show the results both for the vector space model (vsm) and language modeling (lm). The evaluation, as described above, uses $k = 1, 10$ and 100. Retrieval using language modeling with *exact+stem&norm* shows the best performance at rank 1, substantially improving over the baseline text representation ("*exact*" row). Hence, this model is used in our subsequent experiments.

5 Learning table lookup labels

We consider the mapping of a question to a table lookup label $\langle T, QF, AF \rangle$ as a classification task and apply a supervised machine learning method. Since we consider our knowledge base a black box and make no assumptions about the semantics of table and field names, we need to

[2] A group of words gathered under one sense in WordNet.
[3] We use the TREC QA newspaper corpus

Qword	When/What/When/Name etc.
Qtype	First noun after Qword
QfocusNoun	First noun of the question $\neq$ Qtype
Qverb	First verb in the question
freq(type of NEs)	Number of occurrences of Locations/ Persons/Organizations etc.

Table 2. Question features

automatically generate data to train the classifier.

We generate training data using the model selected in Section 4. For each question q we use the retrieval model to generate a ranked list of field values from our database. We select field values whose table name T and field name QF occur for the first time in the ranked list, and the value of some other field AF of the corresponding row of T contains the correct answer. I.e., we find T, QF and AF such that the query "select AF from T where $sim(QF, Q)$" returns a correct answer to question q at the top rank. For training we use label $\langle T, QF, AF\rangle$ as a correct class for question q. This way, we may find multiple labels for a question; therefore, we implemented two modes of creating labeled data. In *single-class* mode a question fits in only one class and selects as correct the table lookup label found first in the ranking. In *multiple-class* mode we allow multiple classes for a question and assume all labels found to be correct.

Next, in order to train a classifier on the labeled data, we represent each question as a set of features (see Table 2). We use a stastical part-of-speech tagger TnT [7] and a named entity tagger based on TnT to generate question features. Finally, we train Timbl [8], a memory-based classifier, and use a parameter optimization tool [19] to find the best setting for Timbl; see Figure 4 for an overview.

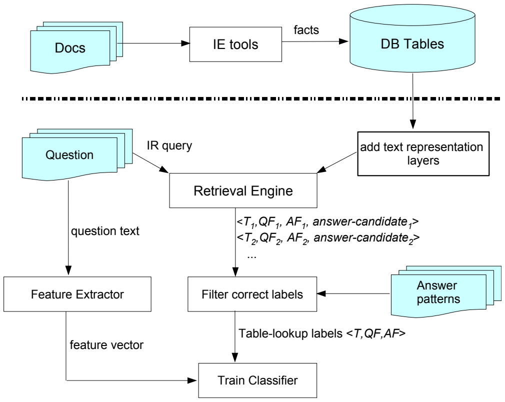

Figure 4. Learning table lookup labels

6 Evaluation

We have described how we select the best retrieval method and train a classifier that assigns table lookup labels to questions. Now we turn to the evaluation of the resulting QA system. Our research question in setting up the experiments is whether our machine-learning method for answering questions from tabular data can compete with a system that uses manually created lookup rules.

QUARTZ	This paper				
a@1	a@1	a@5	a@10	a@20	MRR
11.09%	21.3%	25.4%	27.1%	28.7%	0.234

Table 3. Evaluation of the QA system.

	Single class		Multiple class	
	−norm	+norm	−norm	+norm
a@1	17.9%	21.3%	13.8%	15.8%
a@5	22.8%	25.4%	18.6%	21.0%
a@10	24.5%	27.1%	20.7%	23.1%
a@20	26.5%	28.7%	22.2%	24.6%

Table 4. The impact of training data generation mode and NE normalization.

We used a standard set of question/answer pairs from the TREC QA tasks of 2001–2003 and a knowledge base with tables extracted from the AQUAINT corpus using the information extraction tools of QUARTZ [1] (see Section 3). We index these tables using a vector space model [2] and a language modeling approach [11], using different layers of text representation as described in Section 3.

We split our training corpus of 2,136 TREC questions with answers into 10 sets and run a 10-fold cross-validation. For each 9/1 split we train a classifier to predict table lookup labels of questions using the 9 sets and evaluate the resulting QA system on the questions of the remaining set. The performance of the system is measured using the Mean Reciprocal Rank (MRR, the inverse of the rank of the first correct answer, averaged over all questions) and accuracy at n ($a@n$, the number of questions answered at rank $\leq n$).

Table 3 shows the evaluation results averaged over our 10 folds. We measure the performance of the system, using the accuracy at n (for $n = 1, 5, 10$ and 20) and the mean reciprocal rank. For comparison, we also give the results of the same 10-fold cross validation for the Table Lookup stream of the QA system QUARTZ [1], that uses exactly the same knowledge base but employs a series of manually created pattern-based rules to map questions to SQL queries to the database and return the answers. The machine learning-based system clearly outperforms the rule-based method of QUARTZ on the same set of tables.

The results shown in Table 3 use the single-class mode of training data generation (see Section 5). Table 4 also shows the results for the multi-class mode, with and without named entity normalization.

The experimental results show that named-entity normalization improves accuracy of our QA system, even given the incompleteness of WordNet.

7 Conclusions and further work

We described an open domain QA system that answers natural language questions using tabular data that has been automatically extracted from a newspaper collection. The system maps an incoming question to a table lookup label $\langle T, QF, AF\rangle$ and returns as an answer the content of the field AF of the row of table T for which the similarity between the content of the field QF and the question is the highest, according to some retrieval model. Effectively, the system retrieves the answer by executing a query "select AF from T where $sim(Q, QF)$." We have compared several retrieval models (i.e., implementations of $sim(\cdot,\cdot)$) and we have described a classifier that maps questions to table lookup labels, using a training corpus of question/answer pairs. Our evaluation results show that our data-driven method for answering questions from tabular data outperforms a rule-based QA system on the same tabular data, and moreover, a simple named entity normalization step has a positive effect on the system's performance.

Unlike other NLIDB approaches, our QA method does not require any information about the semantics of the knowledge base, but treats it as a black box. Given a set of tables containing short textual facts, the only input that our system needs is a set of questions with correct answers that can be used for training. The evaluation shows that the system copes well with the noise in the data which is unavoidable if the tables are generated automatically.

In future work we will look at more sophisticated named entity normalization techniques (e.g., using Wikipedia) and a proper word sense disambiguation module. We will also experiment with different classifiers and feature extraction modules and evaluate our approach on different datasets.

Acknowledgements

We thank Katja Hofmann for her help with the named entity normalization. This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 354-20-005, 600.065.120, 612-13- 001, 612.000.106, 612.066.302, 612.069.006, 640.001.501, 640.002.501, and by the E.U. IST programme of the 6th FP for RTD under project MultiMATCH contract IST-033104.

References

[1] D. Ahn, S. Fissaha, V. Jijkoun, K. Müller, M. de Rijke and E. Tjong Kim Sang. Towards a Multi-Stream QA-as-XML-Retrieval Strategy. In *Proc. TREC 2005*, 2006.

[2] Jakarta Lucene text search engine. `http://lucene.apache.org`, 2002.

[3] D. Ahn, V. Jijkoun, K. Müller, M. de Rijke, and E. Tjong Kim Sang. Towards an offline XML-based strategy for answering questions. In *Accessing Multilingual Information Repositories*, pages 449–456, 2006.

[4] J. Allan, C. Wade, and A. Bolivar. Retrieval and novelty detection at the sentence level. In *Proc. SIGIR 2003*, pages 314–321, 2003.

[5] I. Androutsopoulos, G. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Journal of Language Engineering*, 1(1):29–81, 1995.

[6] M. W. Bilotti, B. Katz, and J. Lin. What works better for question answering: Stemming or morphological query expansion? In *Proc. IR4QA Workshop at SIGIR 2004*, 2004.

[7] T. Brants. *TnT – A Statistical Part-Of-Speech tagger*. In *Proc. of the 6th Applied NLP Conference*, 2000.

[8] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. Timbl: Tilburg memory-based learner - version 4.0 reference guide.

[9] M. Fleischman, E. Hovy, and A. Echihabi. Offline strategies for online question answering: answering questions before they are asked. In *Proc. ACL '03*, pages 1–7, 2003.

[10] D. Harman. Overview of the TREC 2002 novelty track. In *Proc. TREC 2002*, pages 17–28, 2002.

[11] D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. In *Proc. ECDL 1998*, pages 569–584, 1998.

[12] V. Jijkoun, M. de Rijke, and J. Mur. Information extraction for question answering: Improving recall through syntactic patterns. In *Proc. COLING 2004*, 2004.

[13] X. Li and D. Roth. Learning question classifiers. In *Proc. COLING 2002*, 2002.

[14] D. Metzler and W. Croft. Analysis of statistical question classification for fact-based questions. *Journal of Information Retrieval*, 8:481–504, 2005.

[15] D. Metzler, S. Dumais, and C. Meek. Similarity measures for short segments of text. In *Proc. ECIR 2007*, pages 16–27, 2007.

[16] C. Monz. *From Document Retrieval to Question Answering*. PhD thesis, University of Amsterdam, 2003.

[17] M. Negri. Sense-based blind relevance feedback for question answering. In *Proc. IR4QA Workshop at SIGIR 2004*, 2004.

[18] G. Ramakrishnan, S. Chakrabarti, D. Paranjpe, and P. Bhattacharyya. Is question answering an acquired skill? In *Proc. WWW*, pages 111–120, 2004.

[19] A. van den Bosch. Wrapped progressive sampling search for optimizing learning algorithm parameters. In *Proc. BNAIC 2004*, 2004.

[20] E. Voorhees and D. Tice. The TREC-8 question answering track evaluation. In *Proc. TREC-8*, 1999.