Commercial Applications through Community Question Answering Technology

Antonio Uva, Valerio Storch, Casimiro Carrino, Ugo Di Iorio and Alessandro Moschitti

DOI: 10.4000/books.aaccademia.2482
Publisher: Accademia University Press
Place of publication: Torino
Year of publication: 2017
Published on OpenEdition Books: 19 April 2018
Serie: Collana dell’Associazione Italiana di Linguistica Computazionale
Electronic ISBN: 9788899982942

http://books.openedition.org

Electronic reference
Commercial Applications through Community Question Answering Technology

Antonio Uva†, Valerio Storch†, Casimiro Carrino†, Ugo Di Iorio†, Alessandro Moschitti‡

†Department of Computer Science and Information Engineering, University of Trento, Italy
‡Qatar Computing Research Institute, HBKU, Qatar
†RGI Group, Italy
antonio.uva@unitn.it, amoschitti@gmail.com
{valerio.storch,casimiro.carrino,ugo.diiorio}@rgigroup.com

Abstract

English. In this paper, we describe our experience on using current methods developed for Community Question Answering (cQA) for a commercial application focused on an Italian help desk. Our approach is based on (i) a search engine to retrieve previously answered question candidates and (ii) kernel methods applied to advanced linguistic structures to rerank the most promising candidates. We show that methods developed for cQA work well also when applied to data generated in customer service scenarios, where the user seeks for explanation about products and a database of previously answered questions is available. The experiments with our system demonstrate its suitability for an industrial scenario.

Italiano. In questo articolo, descriviamo la nostra esperienza nell’usare i metodi attualmente disponibili per il Community Question Answering (cQA) in un’applicazione commerciale riguardante il servizio clienti in lingua italiana. Il nostro approccio si basa su (i) un motore di ricerca per recuperare le domande candidate precedentemente risposte e (ii) metodi kernel applicati a strutture linguistiche avanzate per riordinare i candidati più promettenti. Mostriamo che i metodi sviluppati per il cQA funzionano bene anche quando applicati ai dati generati nell’ambito dell’assistenza clienti, dove l’utente cerca informazioni riguardo a dei prodotti e una base di dati di domande precedentemente risposte è disponibile. Gli esperimenti sul nostro sistema dimostrano l’appropriatazza del suo utilizzo in uno scenario industriale.

1 Introduction

In recent years, open-domain Question Answering (QA) has been more and more used by large companies, e.g., IBM, Google, Facebook, Microsoft, etc., for their commercial applications. However, medium and smaller enterprises typically cannot invest billions of dollars in achieving the desired QA accuracy: this limits the use of this technology, especially, in case of less supported languages, e.g., Italian. One viable alternative for smaller companies is the design of close-domain systems looking for answers in specific data. For example, most companies require to quickly and accurately search their own documentation or the one of their customers, which are often available in terms of unstructured text. However, even this scenario is complicated as reaching the satisfactory accuracy may require a lot of resources.

An interesting alternative is provided by cQA technology, which uses techniques tailored for answering questions in specific forums. In addition, to the intuitive observation that the forum topics are rather restricted, making the retrieval task easier, cQA offers an even more interesting property: when a new question is asked in a forum, instead of searching for an answer, the system tries to look for a similar question. Indeed, similar questions were asked before and may have received answers, thus the system can provide the users with such responses. The main advantage of this approach is that searching for similar questions is much easier than searching for text answering a given question. Due to this, challenges such as SemEval-2017 Task 3 (Nakov et al., 2017) and QA4FAQ (Caputo et al., 2016), aimed at testing current cQA available technology, have been organized.

In this paper, we show that help desk applications, generally required by most companies, can adopt the cQA model to automatize the answering process. In particular, we describe our QA system developed for RGI, which is a software vendor
specialized in the insurance businesses. One important task carried out by their help desk software regards answering customers’ questions using a ticket system. Already answered tickets are stored in specialized databases but manually finding and routing them to the users is time consuming. We show that our approach, using standard search engines and advanced reranker based on machine learning and NLP technology, can achieve answer recall of almost 85% when considering the top three retrieved tickets. This is particularly interesting because the experimented data and models are completely in Italian, demonstrating the maturity of this technology also for this language.

2 Related Work
The first step for any system that aims at automatically answering questions on cQA sites is to retrieve a set of questions similar to the user’s input. Over time, different approaches have been proposed. Early methods used statistical machine translation to retrieve similar questions from large question archives (Zhou et al., 2011). Other approaches (Cao et al., 2009; Duan et al., 2008) use language models with smoothing to compute semantic similarity between two questions. A different approach that exploits syntactic information was proposed in (Wang et al., 2009). The authors find similar questions by computing similarity between the syntactic trees of the two questions. In this work, we use pairs of similar questions to train our relational model, which detects if two questions have similar semantics.

From an industrial viewpoint, NLP (and especially QA) is one of the hot topics of recent years, although it is still mostly unexplored. Many platforms are emerging in the wide area of chatbot development, e.g., Wit.ai and Api.ai (proposed by Facebook and Google, respectively), which enable intent classification and entity extraction and Meya.ai, which can be used to develop rule-based chatbot systems. However, most of them do not integrate QA models, with the notable exception of Expert Systems’ Cogito Answer, recently adopted by Ing. Direct and Responsa.

3 The RGI application scenario
The scope of the experiments for this research is the evaluation of state-of-the-art QA models to automatize help desk (HD) processes of RGI. RGI is an Independent Software Vendor specialized in the Insurance Industry, counting 800 profession-
Table 1: An example of two similar tickets: the one used as query on the left and one retrieved by a search engine (only using question words) on the right.

<table>
<thead>
<tr>
<th>Question_{org}</th>
<th>Answer_{org}</th>
<th>Question_{rel}</th>
<th>Answer_{rel}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbiamo bisogno delle credenziali di accesso al sistema. Grazie</td>
<td>Buongiorno, questo l’indirizzo mail al quale scrivere per avere le credenziali di accesso al sistema: <a href="mailto:xxx@xxx.xx">xxx@xxx.xx</a> Cordiali saluti</td>
<td>Buongiorno, non troviamo credenziali per accesso sistema. Potete aiutarci? Grazie</td>
<td>Buongiorno, questo l’indirizzo mail al quale scrivere per avere le credenziali di accesso al sistema: <a href="mailto:xxx@xxx.xx">xxx@xxx.xx</a> Cordiali saluti</td>
</tr>
</tbody>
</table>

the related repositories.

3.2 Data description

Data was gathered from the HD support system, where technical issues are tracked and fixed. Basically, we have tickets organized in Question/Answer (Q/A) pairs, along with fields related to specific information, such as ticket ID and the domain problem. The original data size was around 40,000 tickets but most of them do not provide useful information. Thus, we designed a preprocessing phase both to clean and prepare a valid data set: first, we detected and filtered out spurious Question-Answer pairs, concerning unanswered problems, using basic heuristics. Second, we extracted a subset of general-knowledge problems by selecting only tickets belonging to HD1 with a resolution time less than two days. In addition, our data was also reviewed by an expert team to further filter out invalid tickets. As a result, the preprocessing ended with a dataset of 656 Q/A pairs spread over 10 question domains. Examples of our data are shown in Table 1.

4 Our QA System

Our system is constituted by (i) a search engine to retrieve questions (along with their associated tickets) similar to the new input question and (ii) a reranker built with state-of-the-art NLP and machine learning technology.

4.1 Question and Ticket Retrieval

We used a standard keyword-based Search Engine (SE) to retrieve a list of questions from our dataset similar to the input one. The score produced by SE is the standard cosine similarity between the vectors of the new and the candidate questions. In particular, we built our SE using Lucene TF-IDF based indexing, available in the open-source ElasticSearch platform.

In order to improve the retrieval quality, we merged user request description (the question) and solution fields in a single joint text to build the ticket index. It should be noted that we only used the question text to build the query for SE as in a real scenario, the asked question is not associated with any answer yet.

For each question, in the filtered data mentioned above, we created a list of Question_{original} - Question_{related} pairs, by querying each ticket and collecting the first 10 relevant results. The obtained clustered data set resulted in a list \((q_{original}, q_{related})\) of 656 (tickets) x 10 (retrieved questions). These pairs were annotated by a team of experts with relevant vs. irrelevant labels to create the training and test sets. For example, Table 1 shows a question pair: an original ticket with question and answer on the left, and a similar retrieved ticket on the right.

4.2 Reranking Pipeline

Given the initial rank provided by SE, we apply an advanced NLP pipeline to rerank the questions such that those having the highest probability to be similar to the query are ranked on the top.

NLP pipeline. We used various Italian NLP processors of TextPro (Pianta et al., 2008) and embedded them in a UIMA pipeline, to analyze each ticket question as well as the questions of the tickets in the rank. The NLP components includes, part-of-speech tagging, chunking, named entity recognition, constituency and dependency parsing, etc. The result of the processing is used to produce syntactic representations of the ticket questions, which are then enhanced by relational links, e.g., between matching words, between two questions of a pair. The resulting tree pairs are then used to train a kernel-based reranker.

Kernel-based reranker. A kernel reranker is a function \(r: Q \times Q \rightarrow \mathbb{R}\), where \(Q\) is a set of questions. Such function tells if questions are similar or not and can be used to sort a set of questions \(q_r\) with respect to an original one \(q_o\). These functions can be implemented in many ways, but in this work we used (i) a kernel function applied to the syntactic structure of the pair questions, together
with (ii) some features capturing text similarity between two questions.

**Feature Vector model.** This feature vector embeds a set of text similarity features that capture the relationship between two questions. More specifically, we compute a total of 20 similarities such as n-grams, greedy string tiling, longest common subsequences, Jaccard coefficient, word containment, cosine similarity and many others.

**Tree Kernel model.** This model takes in input two tickets and measures the similarity between their syntactic trees. In particular, we build two macro-trees, one for each ticket in the pair, containing the syntactic trees of sentences in each ticket question. In addition, we link two macro-trees by connecting the phrases of two questions, as done in (Da San Martino et al., 2016). Then, we applied Partial Tree Kernel (Moschitti, 2006) and obtain the following kernel:

$$K((q_o, q_r)^i, (q_o, q_r)^j) = TK(t(q_o, q_r)^i, t(q_o, q_r)^j),$$

where $q_o$ is the original ticket question and $q_r$ are the questions of similar tickets. In contrast, the function $t(x, y)$ extract the syntactic tree from the text $x$, enriching it with REL tags.

### 5 Experiments

To evaluate our approach, we performed experiments on a dataset composed of 6,650 pairs of ticket questions annotated with similarity judgment, i.e., Relevant and Irrelevant. We selected only questions having at least one answer in the first 10 retrieved tickets. We performed 5-fold cross-validation and used SVM-Light-TK\(^1\) software to train 5 different reranking models. SVM-Light-TK allows us to learn a reranking model that combines both feature vectors and Tree Kernels. The latter are especially useful because avoid the burden of manually engineering feature for this task. A more detailed description of the Tree Kernel models and Text Similarity features employed by the model is reported in (Da San Martino et al., 2016). Then, we used the learned model to predict similarities for all pairs of questions present in each test fold.

### 5.1 Results

We conducted three experiments to assess the effectiveness of the different feature sets, similarity features (Sim), TK and TK+Sim in the reranking model. The baseline is computed by means of the rank given by Lucene. Following previous work of the SemEval challenge, we evaluated our ranking with Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Precision at $k$ (P@$k$).

The results are reported in Tab. 2. As it can be seen, the best results are obtained by combining Sim and TK in the reranker, which improved the MRR and MAP of the IR baseline by 4.22 and 5.33 absolute points, respectively. In addition, P@1, @2 and @3 improved by 3.87, 6.08 and 6.71 absolute points, respectively. This shows the effectiveness of using syntactic structures in powerful algorithms such as TK.

We analyzed some selected errors of our system, focusing on the cases where the search engine performs better than our reranking model. We note that for each cluster of question_original-question_related pairs, when the P@1 is high, our model does not perform better than the search engine, or performs even worse. However, our reranking model always tends to push relevant results on the top.

### 6 Conclusions

In this paper, we have described our experience in building a QA model for an Italian help desk in the field of insurance policies. Our main findings are: (i) the Italian NLP technology seems enough accurate to support advanced cQA technology based on syntactic structures; (ii) cQA model can boost the retrieval systems targeting text in Italian; and (iii) the achieved accuracy seems appropriate to create business at least in the filed of help desk applications, although it should be considered that our results refer to only questions having an answer in our database.

---

\(^1\)http://disi.unitn.it/moschitti/Tree-Kernel.htm
References


Annalina Caputo, Marco de Gemmis, Pasquale Lops, Francesco Lovecchio, Vito Manzari, and Acquedotto Pugliese AQP Spa. 2016. Overview of the evalita 2016 question answering for frequently asked questions (qa4faq) task. In CLiC-it/EVALITA.


