Identifying Requirements in Requests for Proposal: A Research Preview

Andreas Falkner[1], Cristina Palomares[2], Xavier Franch[2(✉)], Gottfried Schenner[1], Pablo Aznar[2], and Alexander Schoerghuber[1]

[1] Siemens AG Österreich, Vienna, Austria
{andreas.a.falkner, gottfried.schenner, alexander.schoerghuber}@siemens.com
[2] Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
{cpalomares, franch, paznar}@essi.upc.edu

Abstract. **[Context & motivation]** Bidding processes are a usual requirement elicitation instrument for large IT or infrastructure projects. An organization or agency issues a Request for Proposal (RFP) and interested companies may submit compliant offers. **[Problem]** Such RFPs comprise natural language documents of several hundreds of pages with requirements of various kinds mixed with other information. The analysis of that huge amount of information is very time consuming and cumbersome because bidding companies should not disregard any requirement stated in the RFP. **[Principal ideas/results]** This research preview paper presents a first version of a classification component, OpenReq Classification Service (ORCS), which extracts requirements from RFP documents while discarding irrelevant text. ORCS is based on the use of Naïve Bayes classifiers. We have trained ORCS with 6 RFPs and then tested the component with 4 other RFPs, all of them from the railway safety domain. **[Contribution]** ORCS paves the way to improved productivity by reducing the manual effort needed to identify requirements from natural language RFPs.

Keywords: Requirements elicitation · Requirements identification · Request for Proposal · Bidding process · Classification

1 Introduction

In a bidding process, an organization or public agency aims at procuring a technological solution by specifying the requirements in a document called Request for Proposal (RFP), which is written in natural language and can be several hundred pages long. Based on this, companies present their bids that need to be compliant to the RFP. Last, the requesting organization agency will select one of these bids (or a combination of them) for developing the solution.

In spite of their technical nature, RFPs tend to mix text describing the requirements with other text that is merely informative ("prose") and thus is not relevant to the bidding company for compliance evaluation. This characteristic forces the bidder to invest resources to identify the real requirements, with the subsequent impact over productivity.

E. Knauss and M. Goedicke (Eds.): REFSQ 2019, LNCS 11412, pp. 176–182, 2019.
https://doi.org/10.1007/978-3-030-15538-4_13

The goal of this paper is to present the first results on the use of a software component, ORCS (OpenReq Classification Service), aimed at extracting requirements from an RFP in an efficient and effective way. In this context, by "effective" we mean a technique that does not miss any requirement (100% recall) and as a second priority, filters as much prose as possible. As stated by Berry [1], there are different notions of effectiveness, and the one above is justified because missing a requirement could damage the organization bid.

The paper is organized as follows. A typical bidding process for large infrastructure projects is introduced in Sect. 2. Section 3 presents the functionalities and internal structure of ORCS, and Sect. 4 shows the results of the preliminary evaluation of ORCS with real data (from Siemens). Finally, the paper is concluded in Sect. 5.

2 Bidding Processes for Large Infrastructure Projects

When an infrastructure provider such as Siemens decides to participate in a bid for a large infrastructure project, several of its departments and stakeholders (project management, finances, system development, etc.) must work together to find a good solution to cover all requirements. Requirements are usually organized and edited using a commercial RE tool such as IBM DOORS or POLARION REQUIREMENTS.

In a typical scenario, a bid project for the RFP is created within the company at the start of the process. A project team is set up with the bid project manager, the requirements manager, and the relevant stakeholders necessary for assessing the RFP. In the initial phase, the main workload is carried by the requirements manager. In the requirements capturing phase, the requirements manager is responsible for screening the RFP and for identifying all relevant and referenced external documents (e.g. international standards). The documents are then imported into the requirements management (RM) tool. The next step for the requirements manager is to analyse the imported documents and distinguish between merely informative text sections and text which specifies relevant requirements. This task is done for every entry in the RM tool. As there can be thousands of entries to be processed by the requirements manager, this is an important issue for improving the requirements management process. Consecutively, the identified requirements are assigned to the relevant stakeholder(s) which are responsible to evaluate them according to different criteria such as risk, compliance, etc. At the end of the bid project, a list of compliance is compiled, which contains a statement about the compliance of the bid, i.e. if and under what restrictions an offer can be submitted.

One of the potential bottlenecks of the process described above is the classification of the requirements as it is currently manually done by the requirements manager. In the following we describe how to speed up this process.

3 Identifying Requirements with ORCS

ORCS is part of a larger recommendation system, OpenReq [2]. ORCS' goal is to recommend a value to a requirement property that is binary (i.e., there are no more than two values available for that property). ORCS tackles that task by providing an API for a binary classifier. In the context of this paper, the property is *isReq*, and it represents whether a piece of text is a real requirement or prose (not relevant for the RE process). In other contexts, property could be requirement fields such as *component* (part of a system to which the requirement is talking about) or *priority*.

Among all the different possibilities, we decided that ORCS would be implemented as a supervised machine learning classifier [3]. Considering that we want to discover the "correct" label of a text and that we have a labelled dataset from previous projects, unsupervised learning techniques, which are useful for discovering how the data in the model is structured, do not suit properly [3].

From all the available supervised machine learning algorithms for classification, we are using Naïve Bayes (NB) [4]. NB is a probabilistic classifier based on applying the Bayes' theorem with strong (naive) independence assumptions between the features. NB is a good algorithm for working with text classification since, when dealing with text, it is very common to treat each unique word as a feature, and since the vocabulary in RFP comprises many thousands of words, this makes for a large number of features. The relative simplicity of the algorithm and the independent features assumption makes NB a strong performer for classifying texts [4]. Consequently, NB needs less training time (and therefore it is more scalable). In addition, NB needs less data for training than other supervised algorithms (such as Random Forest), which makes it good for classifying requirements in companies that do not have available hundred-thousands of requirements.

We built ORCS upon a component that already provides a NB classifier implementation, Mahout[1]. It offers the basics for different machine learning tasks (e.g., classification and clustering). Mahout currently has two NB implementations. The first is standard Multinomial NB. The second is an implementation of Transformed Weight-normalized Complement NB as introduced in [5], which extends the Multinomial NB that performs particularly well on datasets with skewed classes (which is the case in RFP, where most of the texts are real requirements; just a few pieces of text are non-relevant).

However, as most of the available implementations of classification algorithms, installing, configuring and using it is not an easy process, since deep knowledge of Mahout and how it works is needed (e.g., set up of environment paths, synchronization of the calls made, etc.). Therefore, we added specific code on top of Mahout to ease its integration and use by a final user.

Figure 1 shows the microservice-based internal architecture of ORCS. ORCS allows to *Train (MS1)* a machine learning model with specific data (this data is basically a list of pairs <*text, property value*>) and stores - using the *Data Manager (MS5)* - the training in a database (using as key the name of the requirement property and the

[1] https://mahout.apache.org/.

name of the organization). Thus, when the recommendation for the value of a requirement property is necessary, the user just calls the *Classify (MS3)* microservice passing the piece of text that needs the recommendation, the name of the requirement property and the name of the organization. Then, the data manager takes care of setting up the corresponding machine learning model in the core of Mahout and returning the recommendation. In addition, there are microservices to *Update (MS2)* a machine learning model when new data (again tuples of the kind *<text, property value>*) is available and also to *Test (MS4)* the classifier with a *k*-fold cross-validation passing the same kind of tuples as in the last microservice and the number of tests to *k*. As ORCS is part of OpenReq, all the data exchanged by the microservices is based on the ontology presented in [6].

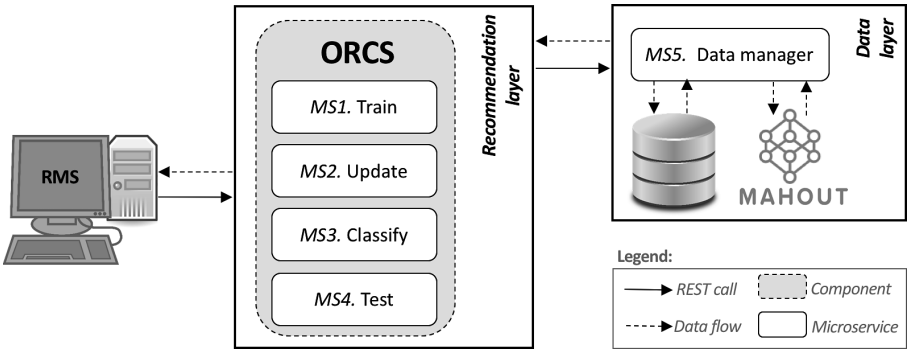

Fig. 1. ORCS' internal architecture

4 Preliminary Evaluation

For the evaluation of the requirements classifier, ten completed bid projects (RFPs) were made available by the Viennese Siemens Mobility department. In total, they comprised 28,200 requirement candidates, all of which had been classified by business experts as either a real requirement (DEF) or a merely informative comment (Prose). An example of a requirement is "A balise group shall consist of between one and eight balises", while an example of Prose is "The purpose of this document is to specify the unified European Train Control System (ETCS) from a technical point of view". The requirements' ID, text, and classification were extracted from the RM tool used at the department and stored in 10 JSON files, one for each project. Six of these projects were randomly chosen as training data set, including cross-validation, and disclosed to the UPC team. These six projects were used by UPC to run a first evaluation of ORCS using a 10-fold cross-validation test (Subsect. 4.1). The remaining four projects were kept secret and only used once for the final evaluation run by Siemens (reported in Sect. 4.2).

In both evaluations we use standard metrics for binary classification considering DEF (i.e., classification of a candidate as a real requirement) as the positive case: recall (true positive rate, number of correct positive results divided by the number of all

relevant sample) and specificity (true negative rate, i.e., number of correct negative results divided by the number of all relevant sample). Recall is most important for the business experts because they want to avoid that some requirement is not detected as such, thus not being checked for compliance during the bid process which may lead to (potentially high) non-compliance costs (a business risk that must be avoided). Specificity is also important because unnecessary efforts arise if many comments are wrongly classified as DEF: experts are invited to check compliance for them although this is not necessary. We refrained from combining those two into a single metric (such as accuracy or F2-metric) in order to give the stakeholders (business experts) the chance to weight the two against each other.

4.1 Evaluation Results During Training and Validation

For first testing the results of ORCS, we used a stratified 10-fold cross-validation, which is a well-known technique to evaluate machine learning models. In this case, we used as sample 6 projects which contained 17,556 requirement candidates. From these candidates, 15,870 (90.4%) requirements were classified as DEF by experts. This means that only 1686 (9.6%) were of type Prose. In the case of ORCS, having unbalanced class labels is not a problem (as explained in Sect. 3).

Table 1 shows the results of calling the ORCS' Test microservice with $k = 10$ and this specific sample. As can be seen in Table 1, average recall is 85.06% and average specificity is 72.04%, showing a good start point for the classifier.

Table 1. 10-fold cross-validation results

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	*Total/ Avge*
# reqs	1734	1742	1689	1801	1788	1697	1792	1788	1719	1806	17556
TP	1335	1346	1280	1391	1372	1284	1389	1394	1327	1382	13500
FP	33	43	47	41	50	64	48	45	46	55	472
TN	133	110	127	131	119	124	129	118	101	122	1214
FN	233	243	235	238	247	225	226	231	245	247	2370
Recall	85.14%	84.71%	84.49%	85.39%	84.74%	85.09%	86.01%	85.78%	84.41%	84.84%	**85.06%**
Specificity	80.12%	71.90%	72.99%	76.16%	70.41%	65.96%	72.88%	72.39%	68.71%	68.93%	**72.04%**

4.2 Evaluation Results on Non-disclosed Test Data

The 4 projects used as test data set comprise 10,700 requirement candidates, 7,300 of which were classified as real requirements by experts. As the resulting prevalence (i.e., occurrence of DEF in the whole set of requirement candidates) of 69% is considerably lower than the prevalence of 84% in the training data set, we investigated and found out that the prevalence in a subset of three projects is 83% (based on micro-averaging [7]) whereas the fourth project shows a prevalence of only 16%. We consider this an outlier which was caused by the fact that the experts rated whole sections of the RFP as Prose because they contained information out of the scope of the bid project (e.g., because their contents were covered by another company in a bidding consortium). Such

constellations cannot be covered easily by text-only classification and we still need to find a way how to deal with them properly.

First results are shown in Table 2. The classifier performs badly on the outlier (Eval 4), with only 21% specificity. However, the micro-average of the rest of the test data has a quite similar performance as the micro-average of the training data (nearly the same recall, specificity smaller by several percentage points). This indicates that the classifier is not overfitted to the training data as long as the prevalence of the test data is similar to the training data.

Table 2. Evaluation results

	Train 1	Train 2	Train 3	Train 4	Train 5	Train 6	*Train avg*	Eval 1	Eval 2	Eval 3	Eval 4	*Eval avg*
# reqs	1525	8123	1854	1382	853	3819	17556	1000	6510	841	2300	*8351*
TP	1112	6234	1390	1056	663	2832	13287	785	4645	675	343	*6105*
FP	59	170	29	51	8	115	432	11	375	13	1520	*399*
TN	257	1073	290	191	99	500	2410	110	784	93	401	*860*
FN	97	646	145	84	83	372	896	94	706	60	36	*987*
Recall	91.98%	90.61%	90.55%	92.63%	88.87%	88.39%	90.30%	89.31%	86.81%	91.84%	90.50%	***87.65%***
Specificity	81.33%	86.32%	90.91%	78.93%	92.52%	81.30%	84.80%	90.91%	67.40%	87.74%	20.87%	***71.21%***

5 Conclusions

In this paper, we present an approach of how identifying requirements in RFP in the setup of Siemens by using ORCS, a component that provides an API for machine learning classification based on NB. In addition, we present preliminary results of testing this component in Siemens. Although the results are good, they need to be improved, especially the recall of component (and therefore the number of false negatives), since we want to avoid that a real requirement is not detected as such, because in that case it would not be evaluated during the bid process, which may lead to non-compliance costs if the bid is won. To achieve this, we aim to improve ORCS in different aspects: NLP preprocessing (mainly stop words removal and lemmatization) and the incorporation of context in the classification process (e.g., the location of the text in the RFP so that we can more precisely differentiate between relevant and irrelevant information).

References

1. Berry, D.M.: Evaluation of tools for hairy requirements and software engineering tasks. In: REW 2017 (2017)
2. Palomares, C., Franch, X., Fucci, D.: Personal recommendations in requirements engineering: the OpenReq approach. In: Kamsties, E., Horkoff, J., Dalpiaz, F. (eds.) REFSQ 2018. LNCS, vol. 10753, pp. 297–304. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77243-1_19
3. Shalev, S., Ben, S.: Understanding Machine Learning. Cambridge University Press, Cambridge (2014)
4. Brink, H., et al.: Real-World Machine Learning. Manning Publications, New York (2016)

5. Rennie, J., et al.: Tackling the poor assumptions of Naive Bayes text classifiers. In: ICML 2003 (2003)
6. Quer, C., et al.: Reconciling practice and rigour in ontology-based heterogeneous information systems construction. In: Buchmann, R.A., Karagiannis, D., Kirikova, M. (eds.) PoEM 2018. LNBIP, vol. 335, pp. 205–220. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02302-7_13
7. Yang, Y.: An evaluation of statistical approaches to text categorization. J. Inf. Retrieval **1**, 69–90 (1999)