# (Semi-) automatic Categorization of Natural Language Requirements

Eric Knauss[1] and Daniel Ott[2]

[1] Department of Computer Science and Engineering
Chalmers | University of Gothenburg, Sweden
eric.knauss@cse.gu.se
[2] Research and Development, Daimler AG
P.O. Box 2360, 89013 Ulm, Germany
daniel.ott@daimler.com

**Abstract. Context and motivation:** Requirements of todays industry specifications need to be categorized for multiple reasons, including analysis of certain requirement types (like non-functional requirements) and identification of dependencies among requirements.This is a pre-requisite for effective communication and prioritization of requirements in industry-size specifications. **Question/problem:** Because of the size and complexity of these specifications, categorization tasks must be specifically supported in order to minimize manual efforts and to ensure a high classification accuracy. Approaches that make use of (supervised) automatic classification algorithms have to deal with the problem to provide enough training data with excellent quality. **Principal ideas/results:** In this paper, we discuss the requirements engineering team and their requirements management tool as a socio-technical system that allows consistent classification of requirements with a focus on organizational learning. We compare a manual, a semi-automatic, and a fully-automatic approach for the classification of requirements in this environment. We evaluate performance of these approaches by measuring effort and accuracy of automatic classification recommendations and combined performance of user and tool, and capturing the opinion of the expert-participants in a questionnaire. Our results show that a semi-automatic approach is most promising, as it offers the best ratio of quality and effort and the best learning performance. **Contribution:** Our contribution is the definition of a socio-technical system for requirements classification and its evaluation in an industrial setting at Mercedes-Benz with a team of ten practitioners.

**Keywords:** requirements, classification, categorization, natural language.

## 1 Introduction

In current industry specifications it is essential to categorize requirements, partly because of their growing size and complexity [1], but also to allow for a number

C. Salinesi and I. van de Weerd (Eds.): REFSQ 2014, LNCS 8396, pp. 39–54, 2014.

of requirements related activities [2]: Identification of requirements of different kinds (e.g. technical or non fun-functional requirements) is a necessity (1) for having specific guidelines for developing and analyzing these requirement types, (2) for architectural decisions, (3) for identifying equipment needed, its quantity and permitted suppliers, and (4) for identifying dependencies among these requirements, especially to detect risks and for scheduling needs during the project. Related to this, Knauss et al. [3] propose an automatic classifier for identifying security-related requirements early in the project, which is crucial in order to prevent substantial security problems later [4–6]. More recent, Ott [7] also reports the need to categorize requirements for inspection tasks to support reviewers with the detection of consistency or completeness defects over large document sets. Note that in this work, we use the term *classification* to refer to the specific algorithmic task of mapping requirements to topics and *categorization* to refer to general goal of establishing a good mapping between requirements and topics for a specification.

Efficient classification can enable focussed communication and prioritization of requirements. As the examples show, categorization of requirements allows filtering relevant requirements for a given important aspect. Considering large specifications, for example in the automotive domain (a single specification at Mercedes-Benz can consist of up to 50.000 requirements and headings [8]), it is necessary to minimize the manual efforts in categorization tasks. Automatic classification is promising [3, 7], but depends on a sufficient amount of high quality training data which is not available in many realistic scenarios.

**Contribution:** In this work, we model a socio-technical system for requirements classification, consisting of the requirements engineering team and their requirements management tool. This socio-technical system allows different modes of operation, ranging from full-automatic over semi-automatic to manual classification of requirements. Our model has a special focus on learning, i.e. gaining shared understanding of a classification scheme in a team and generating high quality training data. For example, our semi-automatic approach learns and adjusts its suggestions with each new requirement according to the user's choices.

We explore the performance of the different operation modes of the socio-technical system in an experiment in cooperation with Mercedes-Benz, driven by the following research questions:

- RQ1: If applied to a new specification domain, how are the *relative quality* levels that can be achieved with the three operation modes *fully-automatic*, *semi-automatic*, and *manual classification*?
- RQ2: If applied to a new specification domain, how high are the *relative efforts* of these operation modes?

Our results suggest that the semi-automatic approach is most promising: it offers significantly better quality than the fully automatic approach, causes less effort than the manual approach, and in addition generates valuable training data as a by-product. In Section 2 we describe related work. Thereafter, we present our model of a socio-technical system (the planned user interactions and

classification mechanisms) in Section 3. Section 4 gives a short overview of our technical solutions and we present the exploratory experiment with Mercedes-Benz in Section 5. We discuss the results in Section 6 and conclude the paper with an outlook on future research in this field.

## 2 Related Works

In this section, we discuss a spectrum of approaches for classification of requirements. On the one side of this spectrum are approaches that are based on purely manual classification, as supported by most state-of-the-art requirements management tools. Analysts specify the classification of requirements in a user-defined attribute. As one such example, Song and Hwong [2] report about their experiences with manual categorizations of requirements in a contract-based system integration project. The contract for this project contains over 4,000 clauses, which are mostly contract requirements.

On the other side of the spectrum are approaches that classify requirements only based on automatic classification. Examples include QuARS tool by Gnesi et al. [9], which automatically detects linguistic defects like ambiguities, using an initial parsing of the requirements. Thereby, QuARS creates a categorization of requirements to topics as a byproduct.

Especially when based on machine learning, such approaches face the problem to obtain large enough training sets in sufficient quality. Knauss et al. [3] evaluate to what extent security-relevant requirements can be automatically identified in specifications based on Naive Bayesian Classifiers. Accordingly, satisfactory results can be achieved, if both training and testing data were derived from the same specification. This is probably due to the fact that writing style and domain specific concepts have a strong impact on the classifier's performance. Ott [7] reports similar results for automatic classification of requirements in multiple categories for supporting review activities. For this reason, Ko et al. [10] propose to automatically create the training data for topic classification. Based on a clustering algorithm they categorize requirements and use these to train Naive Bayesian classifiers. Their evaluation results are promising, but only based on small English and Korean specifications (less than 200 sentences).

Hussain et al. [11] developed the tool LASR that offers an interactive modus for supporting groups in annotation tasks. By not relying on a fully automatic classification approach they mitigate the problem of insufficient training data. In contrast to our work, they try to support a group in collaboratively creating and agreeing on a categorization, whereas we focus on supporting single annotators with a special focus on cost and quality, as well as continuous improvement.

## 3 Socio-technical Requirements Classification

We define a topic as any crosscutting concern that demands for the ability to filter related requirements. Examples include qualitative requirements, such as performance or security-relevance, and crosscutting design issues or constraints

such as regulatory concerns. A requirement can be assigned to a set of topics. Technically, this can be done by adding an attribute *topic* to the requirement and specify relevant topics as a comma separated list.

When requirements are categorized into topics, certain tasks (e.g. creating a security concept, reviewing, prioritizing) become much simpler. As shown by related work, automatic topic classification of natural language requirements is technically feasible but prone to writing style and domain specificity. The main reason for these problems is the lack of sufficient training data in high quality. Thus, the integration of such algorithms in the requirements specification process needs to be considered carefully.

To get a good categorization, the socio-technical system needs to support four main use cases: It should support the author of a requirements document in choosing topics during the documentation of requirements, it should propose relevant topics when the user chooses a topic for a given requirement, it should allow the user to add new topics to the socio-technical system, and it should support assigning topics to a set of requirements that are already documented.

A system for requirements categorization needs to be able to learn, because otherwise it could not adjust to domain specific concepts or writing style. This learning can be observed on several levels. First, users learn a suitable system of topics during working with the requirements. Second, the classification system itself should learn from previous classifications and gain more and more accuracy in proposing relevant topics.

The value of requirements categorization depends on its quality. For example, consider designing a security concept. In this case it is very important that all security relevant requirements are identified. Moreover, the value of the topic classification needs to be higher than the cost to create it.

Figure 1 shows our model of a socio-technical system for requirements classification which can offer different modi of operation. First of all, it allows *manual classification* (automatic classification support = no), the modus with the highest level of freedom. Users can specify a number of requirements, then classify them, before they continue with the specification. We assume that this modus can generate a high quality categorization at high cost.

Secondly, it allows to rely on *fully automatic classification* (in Figure 1: automatic classification support = yes, user confirms classification = no). By eliminating the need for human intervention, the cost to create the classification is minimal (consequently, the dashed transitions in Figure 1 are unusual in this modus. Instead, the user would write the specification and then finalize it by triggering the automatic classification). As has been shown before [7], this approach is highly effective, if enough high quality training data is available, i.e. classified requirements in a closely related domain. Even though this is not unlikely in product centered or software evolution scenarios, there will often be situations where such training data is not available. Consequently, quality of fully automatic topic classification might just be too low for many tasks.

Thus, we are especially interested in a third modus, the *semi automatic classification*. In this case the system recommends relevant topics and allows the user

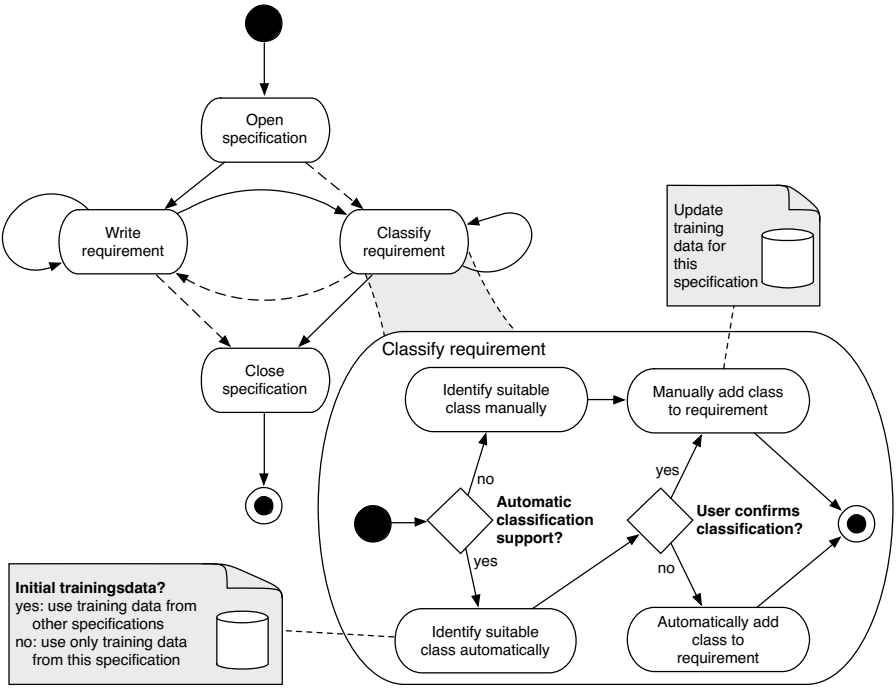


**Fig. 1.** A socio-technical system for requirements classification

to interact by confirming or rejecting recommendations (automatic classification support = yes, user confirms classification = yes). This interaction can be used to train supervised learning algorithms as discussed in the following section, thus generating high quality training data for future versions of this specification.

The screenshot in Figure 2 shows a prototype of a supporting tool for the socio-technical requirements classification system. The prototyp allows working on data from typical requirements management tools, e.g. an authentic, publicly available specification for a door control unit [7] stored in IBM Doors, as shown in the figure. Users can add, remove, and edit requirements. If the user selects or changes a requirement $r_1$, the prototype updates the recommendation list (3 in Figure 2). If the user selects another requirement $r_2$, the prototyp analyses the user's topic classification of requirement $r_1$ and updates its training data accordingly. Depending on the modus, parts of the UI are deactivated and hidden, e.g. the recommendation list in manual modus or both lists in fully automatic modus.

## 4 Text Classification Algorithms

In requirements engineering and management, text classification algorithms can be used to categorize huge document landscapes to certain topics: In past

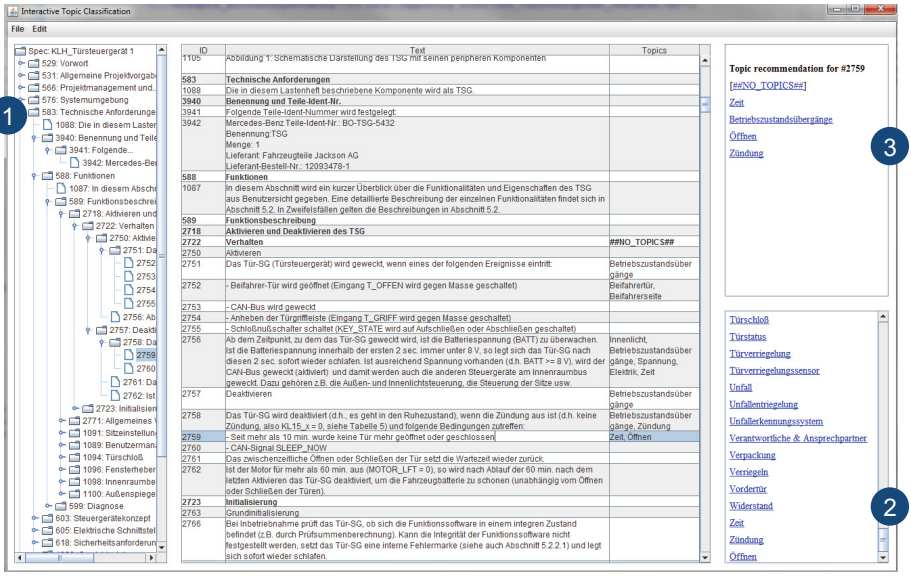


**Fig. 2.** Prototype of a specification tool as part of the socio-technical system. The main area (1) allows editing the requirements specification, (2) allows to assign a category to a requirement from the list of all available categories, (3) a shorter recommendation list allows to asign recommended categories.

research [7], we showed at typical large-scale, German automotive specifications of Mercedes-Benz that an automatic classification using text classification is possible with sufficient quality. Therefore, we identified a well-working combination of pre-, post-processing, and classification steps, out of many alternatives. We will use this combination in the current work, too.

Figure 3 shows details to the individual processing steps. The chosen preprocessing, post-processing, and classification steps have many alternatives, but after a comparison, we got the best results with the illustrated setting in previous work for German natural-language specifications from Mercedes-Benz [7]. A more detailed description to the individual process steps can also be found in this previous work.

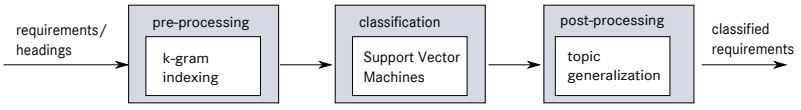


**Fig. 3.** Processing Steps

The **support vector machine** (SVM) approach works as follows (based on Witten et al. [12]) : A nonlinear mapping is used to transform the training data into a higher dimension. Within this new dimension, the classifier searches for the optimal separating hyperplane, which separates the class of topic relevant and topic irrelevant requirements. If a sufficiently high dimension is used, data from two classes can always be separated by a hyperplane. The SVM finds the maximum-margin hyperplane using support vectors and margins. The maximum-margin hyperplane is the one with the greatest separation between the two classes.

The maximum-margin hyperplane can be written as [12]:

$$x = b + \sum_{i \text{ is support vector}} \alpha_i * y_i * a(i) \cdot a$$

Here, $y_i$ is the class value of training instance $a(i)$, while $b$ and $\alpha_i$ are numeric parameters that have to be determined by the SVM. $a(i)$ and $a$ are vectors. The vector $a$ represents a test instance, which shall be classified by the SVM.

Based on external training data (manually classified requirements), the SVM then calculates for each topic such a maximum-margin hyperplane with the greatest separation between the training requirements belonging to the topic and the ones that do not. With this hyperplane the SVM can assign a new requirement to the topic or not.

In the pre-processing step **k-gram indexing** [13], each word of each requirement is separated in each ongoing combination of k letters and the classifier is then trained with these k-grams instead of the whole words. For example, a k-gram indexing with k = 4 separates the word "require" to "requ", "equi", "quir", "uire".

The post-processing step called **topic generalization** takes the structure of Mercedes-Benz specifications into account. All specifications at Mercedes-Benz are written using a template, which provides a generic structure and general requirements, and are filled later with system specific contents. Because of this structure, we assume that if a heading was assigned to a topic, then we can also assign each of the requirements and subheadings below it to this topic. This allows to relate requirements represented by tables or figures (i.e. elements that are not accessible to text classification at all) to the topics of their headings. Tables are implemented as OLE objects in our requirement management tool, so the content of a table is not accessible to our algorithms.

## 5 Evaluation

### 5.1 Research Method

The purpose of this experiment is to test in which way automatic classification helps to reduce effort and to increase quality of topic classifications in industrial requirements specifications. We are also interested in learning effects. For this reason, we explore the impact of *initial training* and *user/tool interaction* on *learning*, i.e. changes of effort and quality over time.

Based on our model of the socio-technical classification system (Figure 1), we defined the following independent variables: *Automatic classification support* can be activated or deactivated. *Initial training* of the automatic classifier might provide better suggestions in the beginning, but take longer to adjust to a special problem domain. *User confirms classification* determines if the user confirms the classification and potentially overrides automatic classifications.

We monitored the following dependent variables: *effort* to provide a requirements specification with topic classification and *quality* of the topic classification. Finally, we controlled for the quality of the requirements specification itself.

In previous work, we showed the fully automatic classification of requirements to multiple topics with satisfactory results, when high quality training data is available [7]. In contrast, we are now applying the socio-technical classification system to new specifications, in different domains, and with new authors. We are especially interested in how difficult it is to adjust the approach to such new environments and if we can derive new training data during that process.

Based on our research questions on effort and quality of the different operation modes of the socio-technical classification system, the independent and dependent variables lead us to the following *hypotheses*:

**H1:** Automatic classification leads to lower quality than manual classification.
**H2:** Automatic classification leads to less effort than manual classification.
**H3:** Starting with an initially trained classifier leads to better classifications than starting with an untrained automatic classifier.
**H4:** An untrained classifier adjusts faster to the problem domain than an initially trained classifier.
**H5:** The combination of automatic classification and user confirmation leads to higher quality of classifications than automatic classification.
**H6:** The combination of automatic classification and user confirmation leads to less effort than manual classification.

We evaluated our hypotheses in an experiment and semi-structured follow-up interviews with the participants (especially for determining a good ratio of $\frac{\text{effort}}{\text{size}}$ for practical use). For this experiment, we define four different (sets of) treatments (Table 1).

Our model of the socio-technical system offers four relevant modi of interaction for categorizing requirements while writing a requirements specification,

**Table 1.** Four relevant sets of treatments of independent variables

| *Treatment* | *Automatic classification support* | *Initial training* | *User confirms classification* |
|---|---|---|---|
| **T1** | no | no | yes |
| **T2** | yes | yes | no |
| **T3** | yes | no | yes |
| **T4** | yes | yes | yes |

which are defined by specific values of the independent variables (treatments). Treatment 1 describes the manual modus of interaction, where requirements are written and classified in parallel without any tool support. Treatment 2 describes the fully automatic operation of the socio-technical system, where users are not involved with classification. This treatment provides us with a baseline on how good the automatic classifier performs on the requirements that are specified during the experiment.

Together, Treatment 1 and 2 allowed us to control for particularities resulting from specific writing styles of the participants. These treatments did not require us to observe the interaction of humans and the system, but provided us with a baseline for the two established operation modi manual and fully automatic: We established a *ground truth* based on (manual) expert classification of all requirements written in the experiment and measured classification quality by comparing against it. To ensure sufficient quality of the ground truth, we let two experts classify the requirements iteratively and measure their agreement in their classification. If the agreement level is below a threshold (based on inter-rater agreement, e.g. Cohen's Kappa [14]), the raters need to discuss situations where they disagree and improve for the next iteration.

Treatment 3 defines the semi-automatic classification modus, where requirements authors write requirements and classify them interactively. In this case, the classification tool was not initialized with any training data and needed to learn the classification from the user. Treatment 4 is equal to 3, except that in this case the classification tool was initialized with training data from other requirements specifications. Both, in Treatment 3 and 4, the socio-technical system learned through interaction between user and classification tool. For Treatment 3 and 4, we randomly assigned the participants in the experiment (controlling only for similar levels of experience in both groups) and provided them with an exemplary implementation of our classification tool prototype that was configured according to the treatment.

### 5.2 Participants and Data Collection

The participants of the experiment were ten developers from Mercedes-Benz with a typical mix of experience (relatively new to expert). Each participant wrote approximately 100 - 300 functional and interface requirements for different parts of two car systems, an outside light system and a speed control system. During the writing tasks, they categorized the requirements with the semi-automatic approaches described in section 5.1. For the categorization tasks we provided the participants with a list of topics relevant for the specification domain in advance. This list consists of 62 topics like, for example, "speed", "ignition", or "communication" which are suitable for supporting specific review tasks [7].

To compare the results of this semi automatic classification with manual classifications, two independent persons manually classified these requirements later. The manual classification of requirements to topics was done by separating the data into parts of 150 objects. Each of these parts was then manually and independently classified by two persons and then synchronized in a review session

using Cohen's Kappa [14] as an aid. Cohen's Kappa is a statistical measure to calculate the inter-rater agreement between two raters, who each classify n items to x categories. If the agreement level is below a threshold ($< 0.9$), the raters need to discuss situations, where they disagree and improve for the next iteration.

For the fully automatic classification (Treatment T2) and the semi automatic classification (Treatment T4) with trained classifier we used the same training data. We derived this training data from additional documents describing the outside light system and the speed control system and from two public specifications of previous work [15]. All in all we used approximately 2,000 requirements for training. The additional documents with descriptions of these two systems were also provided during the experiment to help participants describing the functionality of their parts of the systems.

### 5.3 Descriptive Statistics

Our experiment provided us with three sources of data. Firstly, we compared the endresults with our ground truth (see Section 5.1). Secondly, the prototype of our supporting classification tool was logging all interactions between user and tool. This allowed us to compare how often a user accepts a recommendation. Finally, the questionnaire provided us with insights into the opinion of our expert classifiers.

**Endresults vs. Ground Truth.** Table 2 shows the recall and precision results of of the 10 participants (last three columns). Column 2-4 present the results of Treatment 2 (the fully automatic classifications), applied to the requirements specidied by our participants. The first 5 rows show results for requirements specifications that were created with Treatment 3 (semi-automatic classification without initial training). The last 5 rows show the results for Treatment 4.

**Table 2.** Analyses Results

| | **automatic** | | | **semi-automatic** | | |
|---|---|---|---|---|---|---|
| **participant** | **recall** | **precision** | **f-measure** | **recall** | **precision** | **f-measure** |
| **T3: P1** | 0.42 | 0.36 | 0.39 | 0.30 | 0.77 | 0.43 |
| **T3: P2** | 0.55 | 0.58 | 0.57 | 0.15 | 0.66 | 0.25 |
| **T3: P3** | 0.62 | 0.45 | 0.52 | 0.83 | 0.76 | 0.79 |
| **T3: P4** | 0.48 | 0.46 | 0.47 | 0.80 | 0.89 | 0.84 |
| **T3: P5** | 0.50 | 0.42 | 0.46 | 0.82 | 0.91 | 0.87 |
| **T4: P6** | 0.43 | 0.33 | 0.37 | 0.56 | 0.76 | 0.65 |
| **T4: P7** | 0.47 | 0.56 | 0.51 | 0.49 | 0.73 | 0.59 |
| **T4: P8** | 0.47 | 0.33 | 0.39 | 0.40 | 0.72 | 0.51 |
| **T4: P9** | 0.56 | 0.46 | 0.51 | 0.90 | 0.69 | 0.78 |
| **T4: P10** | 0.46 | 0.33 | 0.38 | 0.34 | 0.57 | 0.43 |

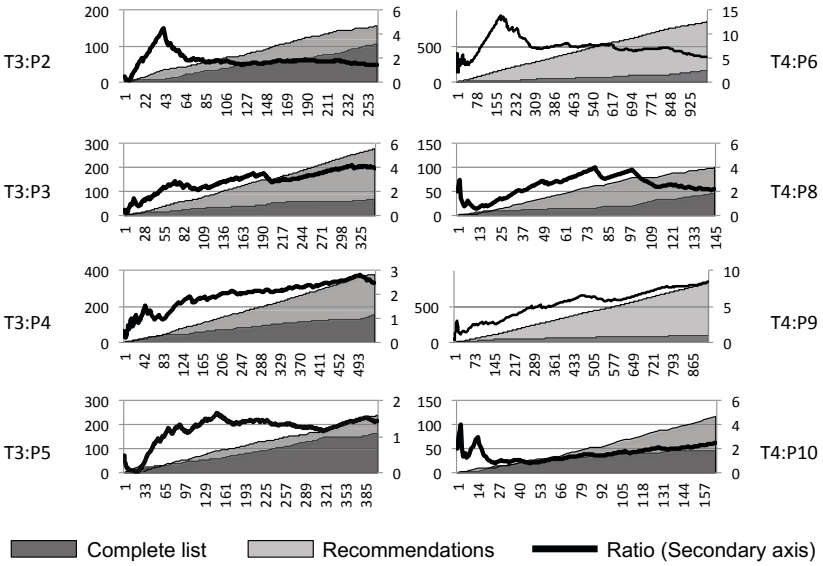


**Fig. 4.** Results of telemetry: Cumulative amount of user picks from *complete list*, *recommendations*, and *ratio* between both

**Telemetry.** During the experiment, each participant could categorize a requirement by either choosing the category from a complete list of all categories, or by accepting a recommendation. Our prototype logs such events and we accumulate how many categories were assigned with each method in Figure 4 (note that telemetry data of two participants was corrupted and is missing).

The figure shows that our participants only preferred the full list over the recommendations for the first 20-30 categorizations, if at all. Later, they tend to assign categories from the recommendation list significantly more often.

Figure 4 also shows the ratio between both ways of assigning categories to requirements. It appears that in both groups this ratio typically ends up around the Factor 2, i.e. twice as many categories are assigned based on recommendations. In Group 3 (left hand side), the ratio is a little bit lower, which might have been caused by the insufficient amount of training. The trajectories in Group 4 seem to have in common that during the first 30 categorizations, a depression occurs (the ratio drops drastically).

**Questionnaires.** Figure 5 shows an excerpt from our questionnaire results, where our participants report on their trust in the automatic classification. Participants in Treatment 4 were more optimistic towards automatic classification. The confidence of participants in Treatment 3 even decreased.

In similar question blocks we asked whether the participants were happy with amount, quality, and improvement rate of the recommendations (Treatment 4:

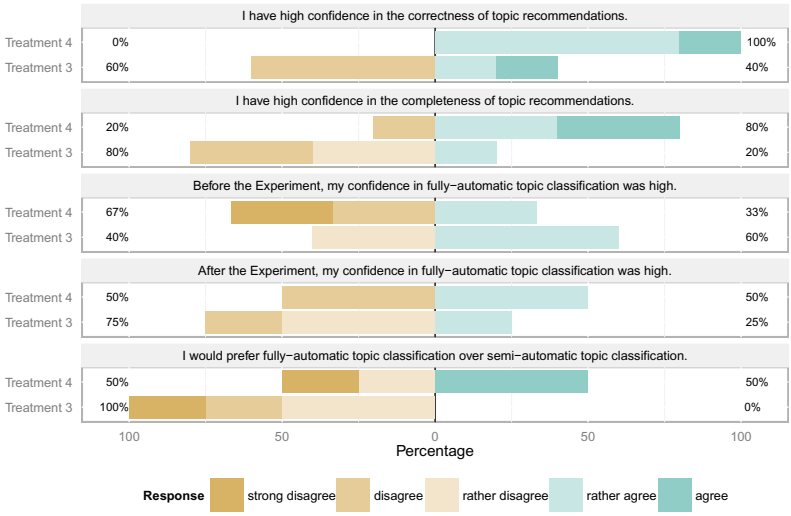


**Fig. 5.** Excerpt from questionnaire: Confidence in automatic classification

answer was generally positive, Treatment 3: less positive and recommendations were not found helpful), whether they think that the semi automatic approach would scale for real world specifications and if they would like to work with such a tool (answer was yes for both treatments), and whether their ability to do the topic classification improved (inconclusive for the scope of this experiment). We also controlled for usability issues of our prototype (see Section 6.4)

# 6 Discussion

## 6.1 Performance of Fully Automatic Classification (H1 and H2)

With respect to (H1) – automatic classification leads to lower quality than manual classification – the results in Table 2 confirm hypothesis and observations we made in previous work [7]. Typical goals for sufficient classification quality (e.g. recall $> 70\%$ and precision $> 60\%$ as proposed for example in [3, 7]) are never met. This is due to the fact that training data from a specification in one domain was used to classify requirements in another specification and domain, leading to a drastic loss of quality of the automatic classification.

Concerning the comparison of effort for manual and automatic classification (H2), the naive answer is of course yes: The effort of automatic classification (ca. 5 min for initialization with training data and $< 1$ min for classification) is lower than the effort of manual classification ($> 1$ hr). However, this does not include the effort to create training data in sufficient quality. For the task of creating training data, we employed similar to the manual classification of

the experiment data two experts that iteratively classified and compared their classification results with Cohen's Kappa to increase the classification quality. From our experience with the public and confidential DCU [7], we know that such a high quality manual classification of a typical specification with 2000-3000 requirements takes at least 150hrs. In the context of this experiment, we used similar amounts of training data and again the total effort for creating this data was at least 150hrs. Since training is not necessarily portable from one application of the tool to another, this effort will not pay off over time.

### 6.2 Cold Start and Ability to Adjust to Domain (H3 and H4)

Concerning the question if initial training of the classifier leads to better classifications (H3), we get mixed results. The boxplot in Figure 6 (left) compares results of semi-automatic classifications by participants with Treatment T3 and T4 with our ground truth. Surprisingly, the deviation of classification results is smaller, but the median of the f-measures is lower, if initial training data (T4) was used.

We were also interested in how quick the recommender system adjusts to a new domain (H4). Our data does not allow to give a clear answer to this. However, it is noteworthy, that participants working with Treatment T4 did encounter some phase of depression during the first 30 classifications, where the ratio of accepted recommendations per classification drastically dropped (Figure 4). This is probably due to the fact that participants were initially having high confidence in the recommendations and realized only later that better classifications were available. The confidence (and ratio) dropped, before it was slowly re-established. This phenomen is not as clearly visible for participants with Treatment T3, where recommendations in the beginning were obviously suboptimal. In addition, participants of T3 mention the phenomen that every topic they choose is automatically recommended in the next requirement, regardless of the content (due to the lack of negative training data). Thus, participants seemingly circumvent the phase of over-confidence.

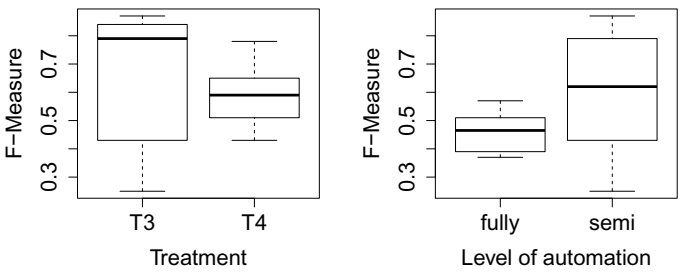


**Fig. 6.** Comparison of classification quality between treatments (left) and level of automation (right)

The overview of the responses in Figure 5 shows a mixed picture for this hypothesis. Participants that were confronted with Treatment 3 (with no initial training data) lost confidence in the automatic classification and strongly oppose fully automatic approaches. Participants that worked with Treatment 4 are indecisive about favouring the semi or fully automatic approach, but slightly gained confidence in the automatic classification.

In summary, all participants generally liked the semi automatic approach and participants with Treatment 4 gave even better ratings than those with Treatment 3. This is no wonder, since the initial suggestions in Treatment 3 were random and therefore not useful. Participants seemed to be more sceptic about recommendations in this situation, leading to a better overall classification performance. In contrast, the higher trust in the recommendations by participants in Treatment 4 lead to more consistent quality and results.

### 6.3 Performance of Semi-automatic Approach (H5 and H6)

We were interested if the semi automatic approach leads to better quality than the fully automatic (H5) and to less effort than the manual approach (H6).

For (H5), the box plot in Figure 6 on the right indicates that indeed the semi-automatic classification is better (in comparison to our ground truth). In fact, the f-measures in Table 2 are always better for the semi automatic approaches compared to the full-automatic classification in the same row.

The telemetry shown in Figure 4 shows that our participants accepted recommendations roughly twice as often as not (ratio $\geq 2$), leaving about $\frac{1}{3}$ of cases where the automatic classification could be improved in the semi-automatic work modus and indicates a quality improvement of the semi automatic approach.

In the questionnaire, we asked our participants to give us the (1) total number of hours, as well as hours (2) for specifying and (3) classifying the requirements.

With respect to (H6) we need to compare these times with typical times for classifying requirements manually. At Mercedes-Benz these times are frequently normalized by number of requirements. Due to their experience, the raters in our experiment were able to classify 150 requirements in 4 $h$, which equals 1.6 $min$ per requirement. Our participants had less experience with classification of requirements. In average, they reported a total effort of 3.76 $min$ per requirement, consisting of 2.99 $min$ for specifying and 0.77 $min$ for classifying it. This significant speedup in classifying requirements is due to the fact that the participants needed less time to read and understand the requirements they had just written.

### 6.4 Threats to Validity

In this section, we discuss the threats to validity based on Runeson et al.'s classification in construction validity, internal validity, external validity and reliability [16]. One obvious threat of the **construction validity** is the manual classification. There is no unique classification and it is reviewer dependent. By using Cohens' Kappa we slightly mitigated this threat, but is still there. Another question is, whether there are no better algorithms for our text classification tasks,

better/additional pre- and post-processing steps or better/additional ways to automatically extract training data, but the results show that we have at least chosen promising candidates. Usability issues of the prototype might be a confounding factor and threat to the **internal validity**. We controlled for this factor by doing System Usability Scale test [17] which showed that no major problems affected the outcome. Concerning the **external validity**, there are limitations in the transferability of our results on natural language specifications drawn from the Mercedes-Benz passenger car development to specifications from other companies in the automotive industry or even to specifications from other industries because of different specification structures, the content and complexity of the specifications, and other company specific factors. In addition, because of the German language, we may have advantages with certain pre-processing steps compared to other languages, while other well known pre-processing steps, for example stemming, do not work on our data sets [7].

In our study we aimed at investigating a socio-technical system for classifying requirements in an industrial setting. We prioritized working in an industrial setting with professional requirements analysts and real world requirements over statistical significance. In our setup, we did not plan to work with enough participants to achieve statistical significance in one of our hypotheses. Thus, the **reliability** of our results has to be considered low, as a different choice of participants with similar background might lead to different results.

## 7 Conclusion and Outlook

In this paper, we introduced a model of a socio-technical system for requirements classification, which consists of a requirements engineering team and their requirements management tool with automatic classification support. The model offers three different operation modes and we argue that the semi-automatic classification can mitigate some major problems in automatic requirements classification: According to our exploratory study it offers a reasonable ratio of effort and quality when compared with the alternatives. It supports organizational learning in that it automatically adjusts to new domains. Thus, it mitigates the problem of insufficient training data that impedes the use of fully automatic classification approaches in many industrial settings. During the evaluation, we learned first hand how difficult it is to agree on a specific classification scheme in a team. The semi automatic approach seems to offer an advantage over the manual approach in that the recommendations transport some shared knowledge across the team. Future work should investigate these effects. We hope that others will profit from our model of a socio-technical requirements classification system as well as from its empirical exploration.

## References

1. Regnell, B., Svensson, R.B., Wnuk, K.: Can we beat the complexity of very large-scale requirements engineering? In: Rolland, C. (ed.) REFSQ 2008. LNCS, vol. 5025, pp. 123–128. Springer, Heidelberg (2008)

2. Song, X., Hwong, B.: Categorizing requirements for a contract-based system integration project. In: Requirements Engineering Conference (RE), pp. 279–284. IEEE (2012)
3. Knauss, E., Houmb, S., Schneider, K., Islam, S., Jürjens, J.: Supporting requirements engineers in recognising security issues. In: Berry, D. (ed.) REFSQ 2011. LNCS, vol. 6606, pp. 4–18. Springer, Heidelberg (2011)
4. Neumann, P.G.: Requirements-related risks in critical systems. In: Proceedings of the 4th International Conference on Requirements Engineering (ICRE 2000), Schaumburg, IL, USA, p. 3 (2000)
5. Chung, L.: Dealing with Security Requirements During the Development of Information Systems. In: Rolland, C., Cauvet, C., Bodart, F. (eds.) CAiSE 1993. LNCS, vol. 685, pp. 234–251. Springer, Heidelberg (1993)
6. Dubois, E., Wu, S.: A framework for dealing with and specifying security requirements in information systems. In: Katsikas, S., Gritzalis, D. (eds.) SEC. IFIP Conference Proceedings, vol. 54, pp. 88–99. Chapman & Hall, Boca Raton (1996)
7. Ott, D.: Automatic requirement categorization of large natural language specifications at mercedes-benz for review improvements. In: Doerr, J., Opdahl, A.L. (eds.) REFSQ 2013. LNCS, vol. 7830, pp. 50–64. Springer, Heidelberg (2013)
8. Houdek, F.: Challenges in Automotive Requirements Engineering. In: Industrial Presentations at REFSQ 2010, Essen (2010)
9. Gnesi, S., Lami, G., Trentanni, G., Fabbrini, F., Fusani, M.: An automatic tool for the analysis of natural language requirements. International Journal of Computer Systems Science & Engineering 20(1), 53–62 (2005)
10. Ko, Y., Park, S., Seo, J., Choi, S.: Using classification techniques for informal requirements in the requirements analysis-supporting system. Information and Software Technology 49, 1128–1140 (2007)
11. Hussain, I., Ormandjieva, O., Kosseim, L.: Lasr: A tool for large scale annotation of software requirements. In: Second IEEE International Workshop on Empirical Requirements Engineering (EmpiRE), pp. 57–60. IEEE (2012)
12. Witten, I., Frank, E., Hall, M.: Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques. Morgan Kaufmann (2011)
13. Hollink, V., Kamps, J., Monz, C., De Rijke, M.: Monolingual document retrieval for european languages. Information Retrieval 7(1), 33–52 (2004)
14. Carletta, J.: Squibs and discussions assessing agreement on classification tasks: The kappa statistic. Computational Linguistics 22(2), 249–254 (1996)
15. Ott, D., Raschke, A.: Review improvement by requirements classification at mercedes-benz: Limits of empirical studies in educational environments. In: IEEE Second International Workshop on Empirical Requirements Engineering (EmpiRE), pp. 1–8. IEEE (2012)
16. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Eng. 14(2), 131–164 (2009)
17. Brooke, J.: SUS: A quick and dirty usability scale. In: Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, A.L. (eds.) Usability Evaluation in Industry. Taylor and Francis, London (1996)