Mining User-Generated Repair Instructions from Automotive Web Communities

Thiemo Wambsganß
Karlsruhe Institute of Technology
thiemo-wambsganss@gmx.de

Hansjörg Fromm
Karlsruhe Institute of Technology
hansjoerg.fromm@kit.edu

Abstract

The objective of this research was to automatically extract user-generated repair instructions from large amounts of web data. An artifact has been created that classifies a web post as containing a repair instruction or not. Methods from Natural Language Processing are used to transform the unstructured textual information from a web post into a set of numerical features that can be further processed by different Machine Learning Algorithms. The main contribution of this research lies in the design and prototypical implementation of these features. The evaluation shows that the created artifact can accurately distinguish posts containing repair instructions from other posts e.g. containing problem reports. With such a solution, a company can save a lot of time and money that was previously necessary to perform this classification task manually.

1. Introduction

The volume of data generated on the Internet is still exponentially growing. More than 80 percent of it consists of unstructured or semi-structured data (Talib et al. 2016). Thus, it becomes increasingly difficult to find, organize, and analyze the data to obtain relevant information.

New digital information is generated by Internet users 24 hours per day. This data is called user-generated content (UGC) and exists in diverse forms such as blog posts, tags, tweets, or survey responses, often containing opinions, ratings, recommendations, needs, experience reports etc. (Krumm et al. 2008). Companies consider this data as a valuable source of information (Byrum and Bingham 2016). However, user-generated data is typically unstructured or semi-structured, and thus incapable of being evaluated by standard data mining techniques (Sullivan 2001).

A domain that has rarely been studied is the automated extraction of user-generated repair instructions from web sources. A global community of people is helping each other by sharing repair experiences over the Internet. This is done for various consumer products, most popular is auto repair. One of the largest automotive web communities is the German Internet platform motor-talk.de, where users can post questions, problem descriptions, solution recommendations, and comments as blog entries (Figures 1, 2). Motor-talk.de has more than 2.9 million registered users (March 2018), more than 52 million posts in nearly 800 forums, and records more than 10,000 daily activities. It is clear that the content of this forum represents a highly valuable repository not only for the private user community, but also for the automotive industry.

Even if the automotive industry has their own expert-generated repair manuals, their interest in user-generated content is immense. Many problems and their solutions are not described in the official repair manuals. Repair instructions have typically been created by development engineers and are largely focused on certain mechanical or electrical systems and components. If a problem is not attributable to a specific component, the repair guide is often of little value and it is up to the experience of the repair person to conclude from the symptoms to the problem and to a possible solution. User experience can provide a valuable and important supplement to the professional repair guides. Valuable content is often related to older models which have exceeded their service lifetime or models for which no computer diagnostics is available. And sometimes, users find quick and easy problem solutions which are simply smarter than the solutions recommended by the manufacturer.

For these reasons, the automotive industry has started to exploit the user-generated content of web forums. We worked with a company that spent significant effort to extract repair instructions from automotive web communities. What they have today is a semi-automated process. In a first step, posts from different automotive domains are extracted with a crawler. Then, a filter is used to separate out the posts that presumably do not contain repair instructions.

URI: https://hdl.handle.net/10125/59558
ISBN: 978-0-9981331-2-6

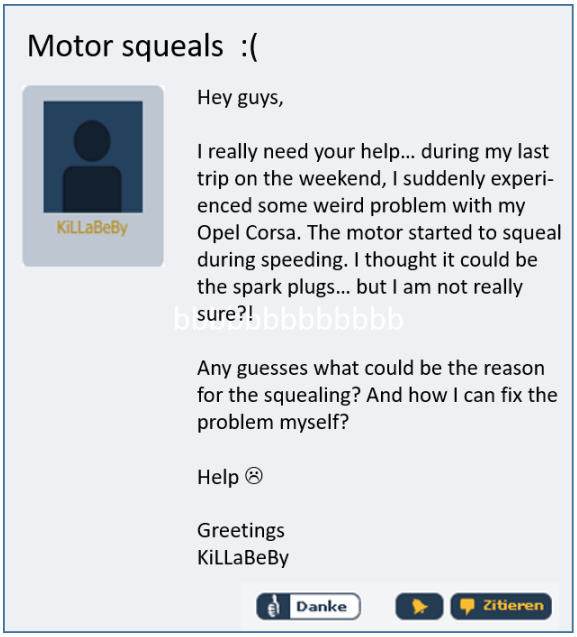

Figure 1: Problem description from *motor-talk.de*

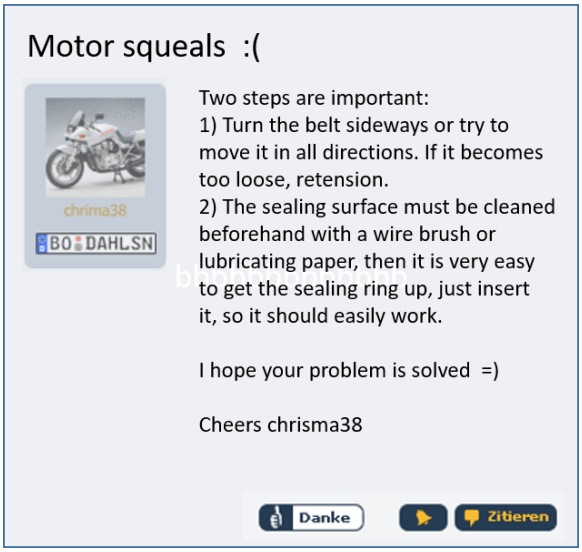

Figure 2: Problem solution from *motor-talk.de*

Afterwards, the candidate posts are reviewed by experts who make a decision on the usefulness of the post. Finally, useful posts are stored in a knowledge repository.

This process is not very efficient, since the quality of the preselection step is poor. The filters are based on simple rules, e.g. if a post contains the words "repair solution", then the post is classified as positive. This has the consequence that on one hand, the experts receive a lot of incorrectly classified posts (classified as "positive", but not containing repair instructions) which they have to sort out manually. This is very time consuming. And on the other hand, posts that contain repair instructions may have been lost since they had been erroneously removed by the rule-based filter.

The motivation of our research was to replace the current rule-based preprocessing system by a more accurate and reliable solution based on machine learning techniques. Any detection rate of over 80% was considered favorable. It was not the intention to replace the human expert. A fully automated classification process was not in the scope of this research. It will be shown later that this decision has a consequence on the performance criterion of the proposed solution.

2. Research Method

Our research work is driven by the "Design Science Research" (DSR) methodology (Hevner et al. 2004). DSR formalizes the creation of an innovative and purposeful artifact for a specific problem domain. The problem should be a relevant business problem and the artifact should be rigorously evaluated. According to Gregor and Hevner (2013), a DSR project is seen as a knowledge contribution if (a) a new solution is developed for a known problem, (b) if a new solution is invented for a new problem, or (c) if a known solution is extended (or adopted from other fields) to a new problem.

In our case, we solved a problem that has not been addressed so far in the research literature. The utilized knowledge base consisted of constructs, techniques, and evaluation criteria used in Natural Language Processing (NLP), Linguistics, and Machine Learning (ML). Text mining methods successfully used in application areas like sentiment analysis, spam filtering, and authorship attribution suggested a similar design for our problem, extracting repair instructions. Our contribution to the knowledge base is an engineered set of features that proves to be useful for instruction extraction. Our hypothesis is that this particular instantiation can be generalized to a much wider range of applications: from automotive to other domains like consumer electronics, from repair instructions to other areas of interest like purchase recommendations. The overarching theory which still needs to be proven is: It is possible to automatically distinguish relevant from irrelevant information on open data sources like the Internet.

The architecture and process flow of our solution is depicted in Figure 3. The purpose of the artifact is to take a collection of web posts (the "corpus") as input and decide for each post if it contains repair instructions or not. For this classification task, text data (unstructured data) has to be transformed into numerical data (structured data). This is done in the

first building block of our solution which uses methods from natural language processing to extract numerical "features" from the text which serve as an input to the machine learning algorithm (second building block).

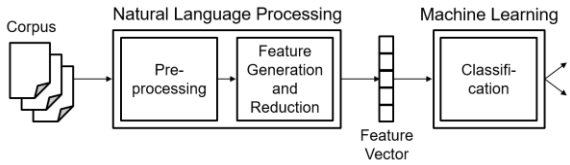

Figure 3: Architecture and Process Flow

Feature extraction is the most essential step in text classification, often "more important than the choice of the learning algorithm itself" (Mishne, 2005). The predominant technique to generate features from text is the bag-of-words approach (Salton, McGill 1983). Bag-of-words counts the number of occurrences of each word and thus represents the text as a multi-dimensional vector. The size of the vector corresponds to the cardinality of the vocabulary of the corpus and the vector is typically very sparse. Feature reduction methods have been proposed to reduce the dimensionality of the feature vector (Yang, Pedersen 1997). Even if bag-of-words entirely neglects the position, order and context of words in a text, and neglects the textual structure such as clauses, sentences or paragraphs, its predictive power for classification is in some applications amazingly high. Instead of single words, sequences of n consecutive words, so-called n-grams (bigrams, trigrams, etc.), can be considered. This leads to a bag-of-n-grams representation of text. Just as bag-of-words, bag-of-n-grams are sparse vectors and call for reduction techniques. Studies showed that bag-of-bigrams are "more powerful than bag-of-words, and in many cases prove very hard to beat" (Goldberg, 2017, p. 75).

Attempts have been made to derive features that extend the simple bag-of-words model. These features could better reflect the content, style, quality and meaning of a text and could additionally consider domain-specific knowledge. There is an ongoing debate in the research community if additional features can improve the simple bag-of-words model. Some authors find significant improvements (Canuto et al. 2014), and others assert that NLP-derived features are about as good as bag-of-words (Godbole 2006). It is a fact that due to the predictive power of bag-of-words and bag-of-n-grams and their ease-of-use, especially in the predominant case of sentiment analysis, little research has been devoted to the investigation of more complex, NLP-based features.

NLP-based features have often been used in fields other than text mining, primarily in linguistics. They can be as simple as counting the number of words in a document and as complex as analyzing the full parse tree of a sentence including semantic information. Typically, we distinguish features according to their linguistic level (lexical, syntactic or semantic) and their dimension (multi-dimensional vector or one-dimensional scalar).

Bag-of-words is a multidimensional feature vector on the lexical (word) level. Number of words is a scalar feature on the word level. More sophisticated features on the lexical level are the well-known readability indices (Kincaid et al. 1975). Readability indices measure how difficult a text is to understand. They use factors like sentence length, syllable count, or percentage of multi-syllable words to calculate a single readability score.

Deeper analysis can be done on the syntactical (grammar) level. Part-of-speech (POS) taggers identify the grammatical type (verb, adjective, noun, etc.) of each word so that certain types of words can be selected for further processing. For sentiment analysis, adjectives (Hatzivassiloglou, McKeown, 1997) or adjective-noun combinations (Turney, 2004) have proven to be more discriminative than other types of words. For analyzing instructional texts, verb-noun combinations seem very promising. These word combinations have been called part-of-speech n-grams (Lioma, van Rijsbergen 2008). POS n-grams can be treated like ordinary n-grams and transformed into a feature vector. Deeper linguistic features require analysis of the complete parse tree (Massung et al. 2013), e.g. the scalar feature tree depth. Parse tree features have been suggested for nationality detection, authorship attribution, essay scoring, but also for sentiment analysis.

Further potential can be expected from features that consider the semantics (meaning) of a text. It is well known that semantic information on words can be obtained by corpus-based or lexicon-based methods (Mihalcea et al. 2006). Lexicon-based methods draw the semantics of a word from existing vocabularies, lexicons or taxonomies. Corpus-based methods draw information on the semantics of words by analyzing the available corpus itself. E.g. for sentiment analysis, a lexicon would provide the words that bear positive or negative sentiment. For detection of user-generated repair instructions in the automotive domain, it could be expected that action verbs and technical terms ("remove the bearing" vs. "ask a question") indicate a repair instruction.

The main objective of our research was to design, combine, and evaluate different features for discriminating posts containing repair instructions

("positive") from posts not containing repair instructions like problem reports and comments ("negative"). The implementation of the entire feature extraction block can largely be based on standard NLP packages. New features require some own software development. Before features can be extracted, the text must be preprocessed. Preprocessing techniques are widely known in the text mining literature. Typical steps are tokenization, stop-word removal, stemming and filtering (Weiss et al. 2010; Uysal, Gunal 2013). Each individual application has its own preprocessing requirements.

For the classification task, it is not known in advance which machine learning algorithm will perform best. Researchers have intensively studied machine learning algorithms for text classification and reported about their experiences in the literature. There is no single algorithm that consistently proves to be superior over others. Therefore, we have implemented the most popular algorithms known to perform well for text classification. The process that follows is the learning or training phase, in which the algorithm pairs the input (features) with the expected output (positive/negative) and gradually improves itself with respect to a given performance criterion. This process is called "supervised learning". After the training phase, the algorithm is evaluated (test phase) before it can be released to classify new web posts in a productive environment (utilization phase).

We followed a three-cycle DSR approach as suggested by Kuechler and Vaishvani (2008) to conduct this research. Each cycle consisted of the phases awareness, suggestion, development, and evaluation. In the first design cycle, we implemented unigrams and bigrams as our "base features" and tested different machine learning algorithms on this basis. In the second design cycle, we added domain-specific lexical features like post length, readability index, and occurrences of enumerations, greetings, and URLs, and evaluated these with the best performing algorithms from cycle one. In the third design cycle, we added a syntactical analysis (POS tagging) to extract verb-noun combinations that were subsequently transformed into a feature vector. We evaluated the impact of these new features against the features from cycle one and cycle two using the best-performing machine learning algorithms.

3. Related Research

The extraction of repair instructions from user-generated web posts has not been addressed in the research literature so far. We found related research only covering certain aspects of our problem. Tagechi et al. (2003) use support vector machines (SVM) to distinguish procedural from non-procedural text in structured lists (passages easily identified by HTML tags <OL> or <UL>) found on web pages. They use unigrams and POS-tagged n-grams as features with which they reach a high classification accuracy. Tagechi et al. deal with instructions preformatted in HTML lists by the web page provider (professional content), whereas we have largely unstructured, amateurish formulations that additionally have to be distinguished from problem reports and comments. So the two situations are hardly comparable. Yet, we used similar POS-tagged features as will be shown below.

Yin and Power (2006) are browsing the home pages of universities and other educational organizations to identify documents containing procedural texts. They use the occurrence of simple phrases like "only if", "as long as", "so that" to conclude on the existence of an instruction. Naïve Bayes and other classifiers based on these occurrence features yielded acceptable results. The situation is again not comparable with ours.

There is an interesting group of papers that aim at the extraction of deeper knowledge from texts containing instructions. This is not a classification task as in our case, since instructional texts have already been identified as such. The main goal is to "understand" what the instruction exactly says. This can be done by breaking the text down into its functional constituents like preconditions, steps (actions), post-conditions, purpose, instruments, etc. Syntactic (grammatical) and semantic analysis is used to analyze the "discursive and rhetorical structure" (Aouladomar, 2005) of procedural texts. Zhang et al. (2012) have applied this technique to (professionally generated) cooking recipes from the BBC website, car maintenance instructions from the Fiat car owner manual and example procedural descriptions from the Airbus A380 maintenance manual. Delpech and Saint-Dizier (2008) analyzed web pages containing e.g. cooking recipes, do-it-yourself tips, and medical recommendations to answer "how-to" questions.

Remarkable, even if not directly relevant for our endeavor, are attempts to automatically generate structured workflows (flow graphs) from instructional texts. This has been applied primarily to cooking recipes so far (Walter et al. 2011, Schumacher et al. 2012, Yamakata et al. 2013, Mori et al. 2014, Maeta et al. 2015)).

A key for the success of our classification is the choice of appropriate features. As already mentioned, vector features of the bag-of-words type are predominant in most text mining applications. Little is reported on the combination of these "raw" features with more complex, knowledge-based features. Uysal

et al (2013) describe the combination of bag-of-word features with "structural" features such as message length, uppercase character ratio, presence of URLs for SMS spam filtering. Jijkoun et al. (2010) present a statistical analysis of syntactic and semantic scalar features such as subjectivity and polarity scores, number of words and sentences per post, for mining user experiences from online forums (no classification). Bui et al. (2016) use a combination of BoW and knowledge-based scalar features to annotate text snippets from scientific publications such as title, abstract, body, etc. Cossu et al. (2016) use 41 features related to user profiles, but also to tweets (including BoW-type and scalar features) to categorize twitter users.

4. Building the Solution

Our solution was developed using the programming language Python 2.71, since it is widely known, easy to use and supports major libraries for NLP and ML tasks (Bird et al. 2009; Swamynathan 2017). For NLP techniques the library Natural Language Toolkit (NLTK)2 is imported. The ML-related algorithms are called from the Google-supported tool Scikit-learn (Pedregosa, Varoquaux 2011) and its major ML packages (Buitinck et al. 2013). In the following, the major building blocks of our solution, feature extraction and machine learning, will be described.

Feature Extraction

The text of our posts was transformed into lower case and tokenized in order to build sequences of single words for each post. Next, every element is stemmed by using the NLTK stemmer based on the German Porter stemming algorithm. Then, regular expressions are used to replace numbers and special characters. Afterwards, all strings with less than two characters and German stop words are found and removed by using the NLTK-provided German stop-word list. After these preparations, the text is ready for feature extraction.

Bag-of-words and bag-of bigrams were generally created for all posts. Domain-specific features were designed together with experts who were asked how they would classify posts manually. They found that posts containing repair instructions were often opened with a greeting and finished with a farewell. Moreover, such posts often contained enumerations either in words (first, second, then, afterwards, etc.) or numbers (1., 2., 3., etc.). Moreover, repair instructions appeared to be better formulated and longer in number of words than problem reports and comments. This gave rise to include post length and a readability index as features. Table 1 summarizes the features that we have used in the three consecutive design cycles.

Table 1: Domain-specific features in addition to bag-of-words and bag-of-bigrams

Feature	Explanation
Readability Index	Repair instructions are usually written in more elaborate language. Sentences are on average longer and more complex words are used.
Post Length	Repair instructions are significantly longer than problem reports or comments.
Enumerations	Repair instructions are often structured by enumerations or bullet points (Figure 2).
URLs	Authors of repair instructions often refer to other Internet pages to complement their advice.
Greetings and Farewells	Repair solutions often start with a greeting phrase and end with a farewell phrase (Figure 2).
Verb-Noun Combinations	Repair solutions often contain verb-noun combinations like "turn belt", "clean surface", "insert ring" (Figure 2).

It should be noted that bag-of-words, bag-of-n-grams, and bag-of-POS n-grams are vectors of features or multi-dimensional features, whereas readability index and post length are one-dimensional or scalar features.

Feature selection (reduction) is only necessary for multi-dimensional features, in our case bag-of-words, bag-of-bigrams, and bag-of-VN bigrams. For this purpose, the term frequency-inverse document frequency (TF/IDF) technique is used which automatically assigns a weighting factor that reflects the importance of a word within the corpus (Ramos 2003). If the term appears in many documents, it has less discriminating power.

On top of TF/IDF, features can be further reduced by removing terms which appear either with very low frequency or very high frequency in a certain corpus, e.g. remove all tokens which occur in more than 50 percent of all documents. This process of reducing features according to some cut-off threshold is called pruning (Andrews and Fox 2007).

[1] https://www.python.org, last accessed on 01-02-2018

[2] http://www.nltk.org, last accessed on 01-02-2018

Machine Learning

We have implemented the supervised machine learning algorithms that have most frequently been used in this domain: Multinomial Naïve Bayes (MNB), Support Vector Machine (SVM), Artificial Neural Networks (ANN) with single or multilayer perceptrons, Decision Tree, Random Forest and k-Nearest Neighbor.

Machine learning algorithms have two kinds of parameters: parameters that are gradually being adjusted during the learning process to "fit the model to the data" (this is the essence of machine learning), and parameters that describe the unchanged structure and characteristics of the model. The latter are called hyper-parameters. An example of a hyper-parameter is the number of hidden layers in a multi-layer neural network. In contrast, the (regular) parameters of a neural network are the weights that are attached to each connection of neurons. These weights are changed during the learning phase. Hyper-parameters must be determined before the actual learning process starts. A widely used method for optimization is grid search over the input space (Bergstra et al. 2011). By providing a specific parameter grid for each classifier, the Python-based method GridSearchCV(), provided by Scikit-learn, applies an exhaustive search for the best parameter configuration of a classifier (Buitinck et al. 2013).

Machine learning algorithms are trained with labeled data (supervised learning). This labeling must be available before the learning process begins. In our case, it could only be done manually. To obtain a reliable labeling, four experts were hired and appropriately incentivized. One expert starts with the manual labeling of a post, and two other experts review this label. Only if there is consensus among all that the post should get a "positive" label, i.e. contains a repair instruction, this label is kept in the dataset.

To evaluate the performance of the algorithm, a fraction of the labeled data is set aside as a test set. Different approaches to determine the training and tests sets have been proposed. The use of a k-fold cross-validation is widely recommended (Han, Kamber 2006). With this approach, training and testing is performed k times, after the labeled data has been partitioned randomly into k "folds". Each fold serves once as a test set, with the rest of the folds as the training set. 10-fold cross-validation has become an accepted standard in the data mining community (Refaeilzadeh et al. 2009).

Hyper-parameter tuning and parameter training can be combined in a process called nested cross-validation (Statnikov et al. 2005). Nested cross-validation is conducted in two nested loops: the outer loop trains the parameters, and the inner loop tunes the hyper-parameters. In both loops, k-fold cross-validation with possibly different k's is applied. We used a nested cross-validation with 10 outer folds and 4 inner folds, known as a 10x4 cross-validation (Raschka 2015). After each training, the algorithm is used to classify the posts in previously unseen test set. The classification results (positive/negative) are compared with the true labels. Four results are possible: true positive, true negative, false positive, and false negative.

The percentage of correctly classified posts compared to the total number of posts is called accuracy. Since accuracy depends much on the ratio of positive and negative posts in the data set, it can be a misleading measure if the data is unbalanced (Chawla 2005). Therefore, we used two methods, oversampling and undersampling (Rahman, Davis 2013), to generate a balanced dataset. Besides accuracy, other measures are used: precision is the fraction of the truly positive posts among all positively classified posts, and recall is the fraction of truly positive posts that have been classified as positive ("discovery rate"):

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Often, we consider the "costs of misclassification". In our case, false positive posts are less critical since they are anyway presented to the expert who can easily sort them out. False negative posts are lost, since the expert doesn't see them at all. This means that false positive posts are less critical than false negatives. Precision penalizes false positives and recall penalizes false negatives. Therefore, a good measure for our application is one that weights recall more than precision. This is the F2 score:

$$F_2 = \frac{5}{4} \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

In the case of fully automated classification, both false positives and false negatives are equally unwanted. In this case, the F1 score would be appropriate which balances precision and recall:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

We have recorded accuracy, recall, precision, F1 score, and F2 score for all our classification experiments. It should be kept in mind that recall and F2 score are more relevant for the semi-automated classification process that is currently in place, and accuracy, precision and F1 score would become more important as soon as the process will be fully automated.

5. Results

In the following, we present the results of our three design cycles. In the first design cycle, we used a base set of vector features built on bag-of-words (unigrams) and bag-of-bigrams and using TF/IDF. Table 2 shows the performance measures for classification for different machine learning algorithms. The table is sorted by the F2 score in descending order. As mentioned above, we consider recall more important than precision (and accordingly, the F2 score more important than the F1 score), since the posts will be manually post-processed by experts. As mentioned earlier, the benchmark for recall was 80%.

Table 2: Performance of different classifiers for features including unigrams and bigrams

Classifier	Accuracy	Recall	Precision	F_1 score	F_2 score
Multinomial Naïve Bayes	77.05%	96.92%	69.36%	80.86%	89.79%
Linear SVM	86.44%	87.39%	85.76%	86.56%	87.06%
Single-Layer Perceptron	87.61%	85.63%	89.16%	87.36%	86.31%
Multi-Layer Perceptron	87.76%	82.70%	92.01%	87.10%	84.41%
Bagging Classifier	76.69%	67.60%	82.62%	74.35%	70.15%
AdaBoost	67.30%	79.14%	72.74%	69.38%	69.38%
Nearest Neighbor	75.59%	64.08%	83.24%	72.41%	67.17%
Decision Tree	67.38%	58.94%	70.90%	64.37%	61.00%

The best result is achieved by the Multinomial Naive Bayes (MNB) classifier, reaching an F2 score of 89.79 percent with 77.05 percent accuracy, 96.92 percent recall and 69.36 percent precision. The F2-Score largely depends on the high recall of 96.92 percent, which means almost all posts containing solutions are recognized. However, the precision and accuracy are rather low compared to other classifiers.

Both Neural Networks show very promising results when considering a balanced recall and precision combined with a high accuracy. The Single-Layer Perceptron reached a 87.46 percent F1-Score with a precision of 89.16 percent, a recall of 85.73 percent, and an accuracy of 87.61 percent. These results are, from a practical perspective, very promising, since no drawbacks in accuracy and precision have to be taken for a high recall.

Also, the Linear Support Vector Machine (SVM) shows good results with an F2-Score of 87.06 percent, an accuracy of 86.44 percent.

In design cycle 2, we implemented and evaluated the different domain-specific features that are presented in Table 1. The effects of these features vary substantially. It should be noted that features exhibit different frequency distributions which are typically binomial or multinomial for bag-of-words types and normal (Gaussian) for compact types. Most machine learning algorithms make no assumption about the distribution of their input variables except Naïve Bayes. For Naïve Bayes, one has to decide which type of features are used. Multinomial Naïve Bayes (MNB) has proven to work well with features of the bag-of-words type, even if the use of TF/IDF leads to fractional counts (Kibriya et al. 2004). However, the MNB algorithm does not work with compact features due to their different distribution. One could consider an approach combining MBN and Gaussian NB in an appropriate way. We left that to future research and simply skipped MNB for cases with mixed distribution features.

The use of the readability index (Flesch index) yielded noticeable improvements in recall and F2 score for all top performing classifiers (Table 3). The F2 score increases with the linear SVM from 87.06% to 87.99%, with the single-layer perceptron from 84.41% to 88.21%, and with the multi-layer perceptron from 84.42 to 88.21%. At the same time, accuracy increases. This confirms the hypothesis that users are writing repair instructions more thoughtfully and legibly than they formulate problems or comments.

Table 3: Performance of different classifiers for features including unigrams, bigrams, and Flesch readability index

Classifier	Accuracy	Recall	Precision	F_1 score	F_2 score
Multi-Layer Perceptron	86.22%	89.30%	84.12%	86.63%	88.21%
Linear SVM	85.04%	89.59%	82.12%	85.69%	87.99%
Single-Layer Perceptron	87.54%	86.80%	88.10%	87.44%	87.06%

Figure 4 shows the distribution of post lengths (in number of words) for the balanced, undersampled dataset. Short posts are a strong indicator for non-solutions. Since there exists a considerable number of short solution posts as well, these are likely to be overlooked by the classifier, once the length feature is added. Thereby the number of false negatives increases, and, consequently, the recall decreases (Table 4). On the other hand, the number of true negatives increases, the number of false positives decreases, and, consequently, the precision increases. Since we are looking for a high recall and F2 score, the length feature must be dismissed for further classification.

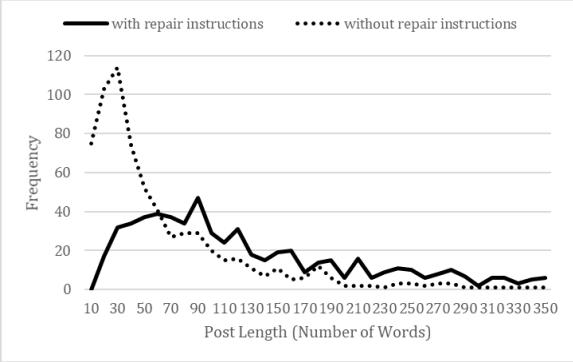

Figure 4: Length distribution of posts containing and not containing repair instructions

Table 4: Performance of different classifiers for features including unigrams, bigrams, and post length

Classifier	Accuracy	Recall	Precision	F_1 score	F_2 score
Linear SVM	85.56%	80.21%	89,82%	84.74%	81.96%
Multi-Layer Perceptron	84.90%	78.45%	90.07%	83.86%	80.52%
Single-Layer Perceptron	84.90%	76.25%	92.20%	83.47%	78.98%

The other features related to the presence of enumerations, greetings, and URLs have little effect on the classification. Obviously, they are used in solution posts as well as in non-solution posts.

In the third design cycle, the syntactical analysis using POS tagging was implemented to identify verb-noun combinations. We didn't count the number of verb-noun combinations in a post (as a scalar feature), but formed a "bag of verb-noun combinations" (a feature vector) with the corresponding TF/IDF values. This vector was appended to the already existing vectors for unigrams and bigrams. Using a vector instead of a scalar feature has the advantage that Multinomial Naïve Bayes can still be applied for reasons described above.

The addition of verb-noun combinations improved the results of all classifiers (Table 5). This is in accordance with results e.g. from sentiment analysis, where bi-tagged phrases also enhanced the performance of classification (Agarwal, Mittal 2015).

Table 5: Performance of different classifiers for features including unigrams, bigrams, and verb-noun combinations

Classifier	Accuracy	Recall	Precision	F_1 score	F_2 score
Multinomial Naïve Bayes	79.18%	96.48%	71.68%	82.25%	90.24%
Linear SVM	86.14%	91.35%	82.74%	86.83%	89.49%
Multi-Layer Perceptron	86.22%	86.80%	85.80%	86.30%	86.60%
Single-Layer Perceptron	87.90%	84.31%	90.84%	87.45%	85.54%

6. Conclusion

Our research has demonstrated that valuable repair instructions can be extracted from automotive forums that contain large amounts of user-generated posts. We used the Design Science Research methodology to create an artifact in three design cycles. The artifact is based on techniques from Natural Language Processing and Machine Learning. The performance of the solution is largely dependent on the appropriate generation and selection of features, which are numerical representations of the text. The identification and combination of these features was the main contribution of this research. In the first design cycle, we used features that are known from other text mining applications, the vector features bag-of-words and bag-of-bigrams. Different machine learning algorithms were trained with the available data, which had been manually labeled by experts. The F2 score and recall were identified as the most important performance measures, since the selected posts will be finally inspected by experts after automatic classification. As expected, the different machine learning algorithms perform differently. In the first design cycle, Multinomial Naïve Bayes was best algorithms with an F2 score of 89,79%. In the second design cycle, we tested different domain-specific features such as the post length, the Flesch readability index, and features related to the presence of enumerations and greetings. From all these, only the readability index improved the results noticeably. The F2 score of the Multi-Layer Perceptron was increased gexplained by the fact that users are writing repair

instructions more thoughtfully and legibly than they formulate problems or comments. In the third design cycle, we were looking for verb-noun combinations, which we believe are a typical characteristic of instructional texts. The inclusion of the vector feature "bag of verb-noun combinations" improved the F2 score of Multinomial Naïve Bayes from 89,79% to 90,24%. In summary, our experiments confirmed an insight that has been reported for other text mining applications: the simple lexical features bag-of-words and bag-of-bigrams have already a very strong predictive power. The classification performance can be improved by adding domain-specific features, but the improvement is not large. This research focused on the evaluation of lexical features and shallow syntactical features like parts-of-speech. Our future research will use deeper syntactical analysis such as complete parse trees and will try to understand the semantic components of the individual instructions. In any case, the results are already so satisfying, that our industry partner has decided to use the solution on a regular basis.

Even if our solution was designed and evaluated for a specific application - detecting repair instructions generated by a German automotive web community - there is nothing language-, domain-, or repair-specific to the artifact itself. This suggests that the solution can be generalized to a much wider range of applications. The proof of this is subject to future research.

7. References

Agarwal, B. and Mittal, N., 2015. *Prominent feature extraction for sentiment analysis* (p. 13). Springer.

Anderka, M., Stein, B. and Lipka, N., 2012, August. "Predicting quality flaws in user-generated content: the case of wikipedia," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* (pp. 981-990). ACM.

Aouladomar, F., 2005. "A preliminary analysis of the discursive and rhetorical structure of procedural texts," in *Symposium on the Exploration and Modeling of Meaning*.h

Bergstra, J.S., Bardenet, R., Bengio, Y. and Kégl, B., 2011. "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems* (pp. 2546-2554).

Bird, S., Klein, E. and Loper, E., 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J. and Layton, R., 2013. "API design for machine learning software: experiences from the scikit-learn project," *arXiv preprint arXiv:1309.0238*.

Byrum, J. and Bingham, A., 2016. "Improving analytics capabilities through crowdsourcing," *MIT Sloan Management Review*, *57*(4), p.43.

Canuto, S., Salles, T., Gonçalves, M.A., Rocha, L., Ramos, G., Gonçalves, L., Rosa, T. and Martins, W., 2014, November. "On efficient meta-level features for effective text classification," In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (pp. 1709-1718). ACM.

Chawla, N.V., 2009. "Data mining for imbalanced datasets: An overview," in *Data mining and knowledge discovery handbook* (pp. 875-886). Springer, Boston, MA.

Cossu, J.V., Labatut, V. and Dugué, N., 2016. "A review of features for the discrimination of twitter users: Application to the prediction of offline influence," *Social Network Analysis and Mining*, *6*(1), p.25.

Delpech, E., Saint-Dizier, P., 2008. "Investigating the structure of procedural texts for answering how-to questions," in *Language Resources and Evaluation Conference (LREC 2008)* (pp. p-544).

Godbole, S., 2006. "Inter-class relationships in text classification," Indian Institute of Technology, Bombay.

Goldberg, Y., 2017. "Neural network methods for natural language processing," *Synthesis Lectures on Human Language Technologies*, *10*(1), pp.1-309.

Gregor, S. and Hevner, A.R., 2013. "Positioning and presenting design science research for maximum impact," *MIS quarterly*, *37*(2).

Han, J., Pei, J. and Kamber, M., 2006. *Data mining: concepts and techniques*. Elsevier.

Hatzivassiloglou, V. and McKeown, K.R., 1997. "Predicting the semantic orientation of adjectives," in *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the European chapter of the association for computational linguistics* (pp. 174-181). Association for Computational Linguistics.

Hevner, A.R., March, S.T., Park, J. and Ram, S., 2004. "Design science in information systems research," *Management Information Systems Quarterly*, *28*(1), p.6

Jijkoun, V., de Rijke, M., Weerkamp, W., Ackermans, P. and Geleijnse, G., 2010. "Mining user experiences from online forums: an exploration," In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media* (pp. 17-18). Association for Computational Linguistics.

Kincaid, J.P., Fishburne Jr, R.P., Rogers, R.L. and Chissom, B.S., 1975. *Derivation of new readability formulas (automated readability index, fog count and Flesch reading ease formula) for navy enlisted personnel* (No. RBR-8-75). Naval Technical Training Command Millington TN Research Branch

Krumm, J., Davies, N. and Narayanaswami, C., 2008. "User-generated content," *IEEE Pervasive Computing*, *7*(4), pp.10-11.

Kuechler, B. and Vaishnavi, V., 2008. "On theory development in design science research: anatomy of a research project," *European Journal of Information Systems*, *17*(5), pp.489-504.

Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G., 2004. "Multinomial naive Bayes for text categorization revisited," In *Australasian Joint Conference on Artificial Intelligence* (pp. 488-499). Springer, Berlin, Heidelberg.

Lioma, C. and van Rijsbergen, C.K., 2008. "Part of speech n-grams and information retrieval," *Revue française de linguistique appliquée*, *13*(1), pp.9-22.

Maeta, H., Sasada, T. and Mori, S., 2015. "A framework for procedural text understanding," in *Proceedings of the 14th International Conference on Parsing Technologies* (pp. 50-60).

Milhacea, R., Lieberman, H. and Liu, H., 2006. "NLP for NLP: Natural Language Processing for Natural Language Programming," in *International Conference on Computational Linguistics and Intelligent Text Processing, Mexico City, Springer Lecture Notes in Computer Science*.

Mishne, G., 2005. "Experiments with mood classification in blog posts," in *Proceedings of ACM SIGIR 2005 workshop on stylistic analysis of text for information access* (Vol. 19, pp. 321-327).

Mori, S., Maeta, H., Yamakata, Y. and Sasada, T., 2014. "Flow Graph Corpus from Recipe Texts," in *LREC* (pp. 2370-2377).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, *12*(Oct), pp.2825-2830.

Rahman, M.M. and Davis, D., 2013, July. "Cluster based under-sampling for unbalanced cardiovascular data," in *Proceedings of the World Congress on Engineering* (Vol. 3, pp. 3-5).

Raschka, S., 2015. *Python machine learning*. Packt Publishing Ltd.

Refaeilzadeh, P., Tang, L. and Liu, H., 2009. "Cross-validation," in *Encyclopedia of database systems* (pp. 532-538). Springer US.

Salton, G. and McGill, M., 1983. *Modern information retrieval*. New York NY: McGraw-Hill.

Schumacher, P., Minor, M., Walter, K. and Bergmann, R., 2012. "Extraction of procedural knowledge from the web: A comparison of two workflow extraction approaches," in *Proceedings of the 21st International Conference on World Wide Web* (pp. 739-747). ACM.

Statnikov, A., Aliferis, C.F., Tsamardinos, I., Hardin, D. and Levy, S., 2005. "A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis," *Bioinformatics*, *21*(5), pp.631-643.

Sullivan, D., 2001. *Document warehousing and text mining: techniques for improving business operations, marketing, and sales*. John Wiley & Sons, Inc.

Swamynathan, M., 2017. *Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python*. Apress.

Takechi, M., Tokunaga, T., Matsumoto, Y. and Tanaka, H., 2003. "Feature selection in categorizing procedural expressions," in *Proceedings of the sixth international workshop on Information retrieval with Asian languages -Volume 11* (pp. 49-56). Association for Computational Linguistics.

Talib, R., Hanif, M.K., Ayesha, S. and Fatima, F., 2016. "Text Mining: Techniques, Applications and Issues," *International Journal of Advanced Computer Science & Applications*, *1*(7), pp.414-418

Turney, P.D., 2002. "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews.," in *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 417-424). Association for Computational Linguistics.

Uysal, A.K. and Gunal, S., 2014. "The impact of preprocessing on text classification," *Information Processing & Management*, 50(1), pp.104-112.

Walter, K., Minor, M. and Bergmann, R., 2011. "Workflow extraction from cooking recipes," in *Proceedings of the ICCBR 2011 Workshops* (pp. 207-216).

Weiss, S.M., Indurkhya, N., Zhang, T. and Damerau, F., 2010. *Text mining: predictive methods for analyzing unstructured information*. Springer Science & Business Media.

Yamakata, Y., Imahori, S., Sugiyama, Y., Mori, S. and Tanaka, K., 2013. "Feature extraction and summarization of recipes using flow graph," in *International Conference on Social Informatics* (pp. 241-254). Springer, Cham.

Yang, Y. and Pedersen, J.O., 1997. "A comparative study on feature selection in text categorization," in *ICML* (Vol. 97, pp. 412-420).

Yin, L. and Power, R., 2006. "Adapting the naive Bayes classifier to rank procedural texts," in *European Conference on Information Retrieval* (pp. 179-190). Springer, Berlin, Heidelberg.

Zhang, Z., Webster, P., Uren, V.S., Varga, A. and Ciravegna, F., 2012. "Automatically Extracting Procedural Knowledge from Instructional Texts using Natural Language Processing," in *LREC* (Vol. 2012, No. 520-527, pp. 520-527).