Automatically Quantifying Customer Need Tweets: Towards a Supervised Machine Learning Approach

Niklas Kühl
Karlsruhe Institute of Technology
kuehl@kit.edu

Marius Mühlthaler
Karlsruhe Institute of Technology
marius.muehlthaler@student.kit.edu

Marc Goutier
Karlsruhe Institute of Technology
marc.goutier@student.kit.edu

Abstract

The elicitation of customer needs is an important task for businesses in order to design customer-centric products and services. While there are different approaches available, most lack automation, scalability and monitoring capabilities. In this work, we demonstrate the feasibility to automatically identify and quantify customer needs by training and evaluating on previously-labeled Twitter data. To achieve that, we utilize a supervised machine learning approach. Our results show that the classification performances are statistically superior—but can be further improved in the future.

1. Introduction

The identification and prioritization of customer needs is crucial for businesses in order to succeed in the market [1]. By knowing what the needs, wants and demands of (potential) customers are, commercially successful products and services can be designed. Commonly used methods to identify customer needs are interviews, surveys or observations [2]. While these methods have proven to be successful, they lack automation capabilities and, thus, scalability and continuous monitoring capabilities. Depending on the scope, they can be very time- and cost-intensive. On the customer side, it is nowadays common to share personal information on social media like Twitter, Facebook or Instagram. As Perrin shows, 65% of all Americans and 76% of all American Internet users draw on social media networking services—with a remarkable growth within the last ten years [3]. In the group of young American adults (aged 18 to 29), already 90% use social media. A share of these social media instances contains valuable insights about the needs of customers [4]. With a high volume of social media, e.g. 500 million tweets [5] and 55 million Facebook status updates [6] per day, these platforms present a promising data source to gain knowledge about customer needs in order to design new products and services.

In previous work, an artifact has been presented capable of classifying tweets as to whether they contain customer needs [7]. Such an artifact could be used in a more comprehensive approach, which automatically identifies and quantifies customer needs from micro blog data in general and Twitter data in particular. While the successful design of a machine learning based classifier artifact shows that the automatic identification of "need tweets" is generally feasible, one major limitation remains. The artifact so far is only able to identify tweets containing needs, but not the needs themselves. In order to address this limitation, this work depicts a Design Science Research (DSR) cycle [8], in which we apply a supervised machine learning approach to allow automatic need quantification. The contribution of this work is threefold. First, it provides an overview of the different needs expressed in Twitter for our evaluation domain of e-mobility. Second, it provides a classification model, which can automatically assign new incoming tweets to previously identified categories. Third, the artifact is deployed as a publicly available web service for further usage and integration into other analytical applications.

The remaining paper is structured as follows: After regarding related work, we present our DSR-based methodology, which determines the remaining sections. We elaborate the awareness of the problem, suggest a three-phase iteration based on a machine learning process model, implement it and finally deploy the corresponding web service. We then evaluate the results and finish with a conclusion.

2. Related Work

Gaining insight from data which is voluntarily shared by people on the internet has been of soaring interest in the past years. Social media arrived in most people's everyday life and the amount of available information correspondingly increased [3]. Not only has it become critical for many businesses to be aware

URI: http://hdl.handle.net/10125/50145
ISBN: 978-0-9981331-1-9

of what their costumers expose on social media [9], but also has the analysis of information been able to create value in many other areas. For instance, information systems have been developed to predict the outcome of political elections [10] or to forecast movements in the stock market [11]. A distinct aspect research has drawn significant attention to is the elicitation of opinions of customers out of customer reviews or social media data. Although a common use case of this so-called "opinion mining" [12] is the examination of costumer critiques of products they have purchased on e-commerce platforms, opinion mining has also recently investigated the potential of accumulating the opinions of users of social media platforms [13]. However, opinion mining does not necessarily include machine learning methods. One established method is to directly search for keywords which, for instance, determine the opinion of the writer of a product review [14]. Such a whitelist- based approach for tweets, i.e. retrieving costumer needs out of tweets, can be found in Kühl & Goutier [15]. On the other hand, machine learning systems for automated text classification have been developed immensely in the last years [16]. Hence, a broad variety of different techniques and algorithms exist. A major distinction can be made between supervised and unsupervised machine learning. Whereas supervised learning can allocate new instances to previously defined target values, unsupervised learning refers to models built to discover new patterns and relationships [17]. The two approaches have different advantages and disadvantages, but both have been investigated in previous research. For instance, Pang et al. [18] successfully implemented a supervised machine learning artifact to determine the sentiment of movie reviews. Turney [19] conducted an unsupervised approach to predict whether reviews of different domains (e.g. movies, cars) recommend the products they are about. In contrast, the work at hand focuses on a supervised approach to allocate costumer needs from tweets, which—to the best of our knowledge—does not exist so far.

3. Research Design

In order to design a need classification artifact, we follow the DSR process methodology and its individual phases according to Vaishnavi & Kuechler [20]. In terms of knowledge contribution, the presented work is an *exaptation* according to Gregor & Hevner [8], since we apply a mature solution (supervised machine learning) to the new challenge of automatic need classification. To evaluate the artifact, we use a technical experiment as proposed by Peffers et al. [21]. We evaluate the statistical classification performances of the models identified. The DSR research structure according to Vaishnavi & Kuechler [20] is separated into the steps of problem awareness, suggestion, development, evaluation and conclusion—which determines the remainder of this paper.

4. Prerequisites & Awareness of Problem

While previous work shows the feasibility of identifying whether or not a tweet contains a need, the need itself remains undetected. Aiming for a social information system, which is able to automatically identify and quantify customer needs from Twitter data, it is crucial to not only identify the pure existence of a customer need—but more precisely be able to display the expressed needs in an aggregated version for an innovation manager. Two possibilities on how to tackle this problem arise: supervised and unsupervised approaches. While unsupervised approaches have the advantage to not require any information beforehand, they rely on large amounts of data—which are not always available. Therefore, we explore the option to utilize supervised approaches for a specific evaluation domain as part of this work. A resulting artifact, once trained on manually labeled data, would then be able to automatically classify new incoming tweets regarding their need—and, therefore, allow to both monitor and quantify specific needs automatically—over any period of time. Before going into detail about our chosen approach, we define the terms *customer need* as well as our evaluation domain of *e-mobility* to lay the foundations of the remaining work.

There are three main scientific fields which do research on needs of customers: Psychology, marketing and information systems. In psychology, the focus of need research focusses strongly on fundamental human needs—but does not take economic aspects into account [22]. However, in the field of marketing, one of the most important challenges is to understand customer needs and ways to satisfy them. Kotler & Armstrong [23] separate customer needs into three distinct categories (*needs*, *wants* and *demands*). Needs are often intangible—for example, the needs for mobility or financial security can be interpreted and satisfied in many ways. Therefore, individuals concertize them implicitly by transforming them into wants and demands. Additionally, Harding et al. [24] outline that a customer need can also be expressed as a *requirement* of a product or service. In the field of information systems, research on "requirements engineering" states that a need is considered as a high-level requirement which has to be transformed into low-

level requirements to find ways of fulfilling the need [25]. For the purpose of this work, there is little to be gained differentiating between marketing-oriented customer need definitions (needs, wants, demands) or need definitions in the context of requirement engineering (high-level and low-level requirements): In a first step, any information about needs, regardless of the level of granularity, is valuable information. For simplicity, we, therefore, stick with the term *customer need*—taking all mentioned types into account.

For testing our approach in an application domain, we require candidate domains to be both dependent on fast and ongoing monitoring of arising needs and rich Twitter data traffic. The domain of electric mobility (*e-mobility*) as defined in Scheurenbrand et al. [26] fulfills both our requirements. We further have to narrow down the domain to a geographical area with a coherent set of laws and regulations, markets as well as socio-economic conditions. In addition, we require the Twitter data in a unique language, as we need consistent semantics to analyze. As a result of these requirements on the domain, languages like English and Spanish—which cannot be related to one region—are not suitable. Because of our familiarity with German, we focus on the German-speaking region. While there is also plenty of research on the analysis of micro blog data for English, there is only few research on German instances—on Twitter in particular [27] and social media in general [28].

5. Suggestion

First, we have to select an overall process model for supervised machine learning. As we regard the special case of multiple classification and the need of deploying a fully-working prediction model, we choose the process model of Hirt et al. [29], as it particularly addresses classification model initiation, its error estimation and deployment.

5.1. Model initiation

The model initiation starts with the acquisition of appropriate data and its labeling, followed by the selection of performance measurements to evaluate the machine learning process. Additionally, we decide how we preprocess our acquired data and choose a machine learning algorithm.

5.1.1. Data acquisition & Labeling. The first step in the supervised machine learning classification process is to gather relevant data which we later use for our model training and testing. These manual steps are necessary in supervised machine learning before the automatic need allocation can be implemented. The data should fulfill our requirements for source (Twitter), domain (e-mobility) and target value (need). Historic data of Twitter is not fully receivable [30]. However, it is possible to receive the unfiltered stream of real-time tweets via the Twitter Streaming API [31]. The only feasible way to acquire all instances (tweets) from Twitter is therefore to collect the data from the live stream and continuously store it. Since we are only interested in tweets in the field of e- mobility, we limit the collection of tweets from the data stream based on keywords representing our domain of e-mobility and, in our case, additionally restricted to a geological region and language. Next, we remove non-user generated tweets, e.g. from bots or news media. After filtering, the remaining tweets need to be classified according to whether the message itself contains a customer need. To obtain this piece of information in an objective way, independent participants take part in multiple lab labeling sessions. In these sessions, we instruct the participants to classify a set of tweets. They are incentivized and paid as described by [32], receiving a fixed payment. All participants are given the same definition of *customer need*. Different participants classify each tweet three times. The outcomes are aggregated for the generation of the final data set, regarding only tweets where at least two participants agree it contains a need. The remaining instances are fulfilling our requirements for source and domain—and we know that the instances contain needs.

However, our target value is not the binary decision if one instance contains a need or not, as this is addressed in previous work [7]. We are rather interested in which precise need or need category is mentioned in a tweet. Therefore, we have to label the remaining data again. We choose the method of descriptive coding as described by Saldaña [33]. Descriptive coding consists of multiple iterations in which researchers with domain knowledge analyze the tweet data, instance for instance, and assign codes to every tweet independently. These codes represent customer needs or customer need categories. The first iteration is focused on gaining an understanding about the data. We start to assign codes to an instance every time the instance contains a need. The number of codes equals the number of mentioned needs in the tweets. In case the need of one tweet is similar to a need of another tweet, we use the exact same code. If there is uncertainty about the meaning of a need or which code we should use, we tag the tweet with the *other* code. After finishing the first iteration, the codes are usually very broad and represent rather abstract need categories. We compare our need categories, discuss and merge them for a common categorization. Based

on the initial iteration we focus on three key features during later iterations: First, if we find any consistency based on disjoint of categories or needs, we will rework our codes from the previous iteration. Second, we aim at finding sub categories of our last codes to reveal all needs mentioned. However, we also aim at finding major need categories as the highest level of abstraction. In the end we get labels for the needs and major need categories. Every need label belongs only to one major need category, whereas a major need category contains multiple need codes. Third, the *other* category contains tweets after the initial iteration, so we try to find similar tweets in the *other* category to build new categories. We continue the process until all researchers agree on the last coding as well as all key features are fully executed and fulfilled. In the end every need in every instance is labeled by a code which represents the need. Moreover, every code belongs to one major need category. We can now train a machine learning algorithm either on the need labels or on the major need category labels of the tweets. The only requirement for both options is to have a sufficient [34] amount of instances for every label in our dataset.

5.1.2. Performance measure. Before starting with the machine learning process, we have to choose an adequate performance measure. We weigh different possibilities. If we look from an overall statistical point of view, the area under the ROC curve (AUC) [35] is generally accepted as a meaningful classification indicator—and highly preferred over accuracy [36]. Accuracy does not take class imbalances of the target class into account—but since we want to predict minority classes (5-20% share), accuracy is not specific enough, as it might reveal good results without actually learning the minority class well enough. While AUC is statistically meaningful, it is still worth regarding precision and recall—as both can vary significantly with similar AUC results [37]. If aiming for the highest possible precision (and thus lowest fall-out rate), or aiming for the highest possible recall, both measures on their own are not helpful, but need to be regarded in combination. The balanced compromise between precision and recall is measured by the F_1-score [38]. As we expect innovation managers to neither miss out on relevant instances, nor get presented with wrong predictions, we choose the F_1-score as our core performance measure for this work.

5.1.3. Choosing preprocessing & algorithm. The previous data acquisition and labeling performed result in a dataset of tweets labeled with their needs, and every need belongs to one major category. One general question is therefore whether to build the models upon the discrete needs or only upon the major need categories. This mainly depends on the actual size of the dataset and the amount of needs or need categories found during the development part. We therefore discuss this question in the development. For readability reasons, we continue referring to *needs*, yet every step in the suggestion part can generally be applied to both single needs as well as broader need categories.

Since we need a binary labeling basis in order to train a binary classifier, we first assign binary labels to each tweet, determining whether or not a tweet contains a need, for each need respectively. Every tweet can be assigned to more than one need, but we do not want our machine learning artifact to be bound to combinations of needs, but rather to be able to identify every need independently. We hence conduct every of the following steps for each need separately. We build a classifier using the tweets with the binary labels to determine whether a tweet contains this need. Prior to training a classifier, we first need to find the well-suited preprocessing steps, sampling techniques and classification algorithms for this problem instance. Although the amount of feasible kinds of preprocessing methods is illimitable, we aim to systematically choose and evaluate a broad range of preprocessing steps. We consider the removal of words which do not contain any useful information ("stop words") [39], the removal of words that appear very rarely or too frequently to be significant (frequency removal), combining *n* words into one feature (n-gram) [40], downcasing or linguistic transformations such as stemming [41] or lemmatizing [42]. We use a simple bag-of-words concept to build a feature vector [14], where one tweet is represented as one feature vector. Essentially, this vector contains, for all words in the dataset, the number of occurrences of words in this tweet. In order to extract the words from a tweet, different kinds of tokenization are taken into account. The tokenizers are distinguished in the manner of how they treat punctuation (e.g. emoticons) and in the amount of characters a token must at least have to not get removed. We include over-, under- and no sampling [43] into our collection of possible pre-treatments. We also consider whether to use a term frequency and inverse document frequency (tf-idf) transformer [14], which weights the words in the feature vector according to their relative frequency distribution. Table 1 shows an overview of the preprocessing. Another important aspect clearly is the choice of the classification algorithm. We examine both a Support Vector Machine (SVM) [44] and a Random Forest (RF) Classifier [45]. Following the objective to find the most suitable combination of all

of these elements, it would be best to try out every single permutation with regard to the final model performance. Therefore, we combine all these elements into a factorial design [46]. However, as the amount of different combinations and various values for each element is too high and would therefore be too computationally expensive, we split this step into two runs. In the first run, we gain an overview of the influence of the preprocessing steps on the F_1-score. We are then able to determine the ones that either have very little influence or the ones that always lead to the best scores across all needs. We keep them fixed for the second run, so that we can again try some new steps, together with the remaining variables in the first run that were not consistent in their influence on the F_1-score. We finish with comparing the results of each need. For consistency reasons, we ultimately determine one set of general preprocessing steps that are applied to all need classifiers.

Table 1. Overview of preprocessing options

Preprocessing step	Short description
stop word removal	remove irrelevant words, such as ”are”, ”the”, ”get”
frequency removal	remove words that have a lower or higher frequency of occurrences than the specified threshold
n-gram	combine n words into one sequence
stemming	reduce words to their stem (root form), e.g. ”apples” becomes ”apple”
lemmatizing	replace words by their lemma, e.g. ”gone” becomes ”go”
tokenization	divide text into units, e.g. single words
oversampling	replicate samples of the minority class towards an equal distribution of both classes
undersampling	ignore samples of the majority class towards an equal distribution of both classes
tf-idf transformation	weight the words in the feature vector according to their relative frequency distribution

5.2. Model error estimation

As an intermediate step between *model initiation* and *model deployment*, we conduct a *model error estimation* for each need separately, which later allows us to use the whole dataset for tuning the hyperparameters and training the classifiers we deploy, while still having a statistically safe model evaluation of every classifier. Tuning the hyperparameters means to find the optimal values for the parameters of a classification algorithm (such as the C and the γ for a SVM) that are not directly learned from the data during the model training. The classical approach we use to accomplish this is a grid search, meaning to exhaustively search through all permutations of manually specified sets of possible values [47], [48]. The concept we choose for the model error estimation is a nested cross-validation (CV) [49], which we initiate by defining a parameter search space for the hyperparameters of the algorithm selected in the previous step. For the CV itself, we split the dataset into equal sized folds while maintaining the overall class distribution across all folds. In an inner CV, we tune these hyperparameters with a grid search, and use the hold-out set of the outer CV to gain information about the performance measures on unseen data of a classifier trained with the optimal parameters from the inner CV. This later enables us to perform a grid search with the same parameter space, while being positive that the model performance will be between the minimum and maximum value of the scores on every hold-out set of the outer CV. In the model deployment step, we can therefore omit a global test set and exploit the whole data set to perform a grid search (with the same parameter space as in the nested CV), and simply use the best performing parameters while still having an unbiased model error estimation.

5.3. Model deployment

As the previous step already results in an overall model error estimation, we can now use the whole dataset to perform a final grid search with the same parameter space as for the nested CV. We identify the best combination out of the parameter space using a grid search with a CV and train the classifier on all the data with this exact combination of parameters. This classifier is now able to elicit needs out of a new tweet. Note that we perform these steps separately for each need, resulting in distinct classifiers for each need. For further usage, we aim our classifiers to be available for other users and for integration into other analytical tools. We therefore choose to implement a web service which can use the persistently stored and trained classifiers. Different classifiers of different domains can be plugged into the web service, too. The web service provides one endpoint with two parameters, one for a new tweet and another one for the domain (e.g. e-mobility). Other services can then access this (remote) endpoint and the web service returns all the previously defined needs in the specified domain and their corresponding probability. Being a lightweight web service, it is also practicable to integrate it into other more sophisticated tools. Imaginable solutions could include other web services, for example one that can connect to the Twitter Streaming API [31], pass the tweet onto a service capable of determining whether a tweet contains a need or not, and again pass all the need tweets onto the web service developed in this work. A database could store this information over a period of time. A user interface would then allow users (e.g. innovation managers) to analyze different aspects and, having stored information over a period

of time, analyze the development of needs or the impact of a marketing campaign.

6. Development

After we suggest an approach to gain labeled data, initiate a model, estimate its error and deploy it in a web service, we implement each of the individual steps in the following subsections.

6.1. Model initiation

We start with collecting tweets from Twitter and label the needs in the tweets by descriptive coding. Furthermore, we evaluate different preprocessing techniques and algorithms by a grid search to select well-suited preprocessing steps for our dataset.

6.1.1. Data acquisition & Labeling. The technical retrieval of relevant tweets is not part of this work and is only explained briefly. A more detailed description can be found in Kühl et al. [7], who illustrate an approach to automatically detect tweets containing customer needs. We conduct the retrieval of relevant tweets by using the Twitter Streaming API [31]. We collect every instance (tweet) which contains at least one word of a predefined keyword list. The list is reasoned on the opinion of professionals as part of a workshop and on popular electric vehicles in Germany. From March 2015 to May 2016 and from November to February 2017, over 2 million tweets are collected. Based on the language information of Twitter, all non-German tweets are sorted out—which reduces our dataset to 107,441 instances. After the identification of user-generated content [7], the dataset amounts to 6,996 possibly relevant tweets. After labeling, we finally end up with 1,093 remaining instances containing needs, which are only identified as such if at least 2 out of 3 labelers agree on the tweet containing a need. This resembles the dataset of the work at hand. The labeling of the tweets regarding their needs and major need categories starts with our first look at the 1,093 instances. Researchers with knowledge about e-mobility, who were not part of the previous labeling process, conduct the descriptive coding [33]. We perform five iterations of clustering until we reach saturation. In the end, we reveal seven major need categories, the category *other* and 28 different needs which are depicted in table 2. To decide if we train our machine algorithm on the need labels or the major need category labels, we have to take the amount of instances for every label into account. Whereas some needs like car price and politics have a large amount of instances, six needs have only a single-digit amount of instances. The lowest number of instances for a major need category is 71, which represent slightly over 5% of all found needs. Therefore, we decide to continue to train our machine learning algorithm on major need categories.

Table 2. Quantitative share of the major need categories and the needs for the regarded tweets, n=1,093

Major Need Category	Amount (Share)	Need	Amount (Share)
price	202 (14.8%)	car price	154 (11.2%)
		electrical price	22 (1.6%)
		price (other)	22 (1.6%)
		oil/gas price	4 (0.3%)
car characteristics	145 (10.6%)	car characteristics (other)	66 (4.8%)
		car design	28 (2.0%)
		car sound	19 (1.4%)
		driving experience	15 (1.1%)
		car comfort	7 (0.5%)
		car performance	6 (0.4%)
		car smell	4 (0.3%)
charging infrastructure	305 (22.3%)	charging infrastructure existence	191 (14.0%)
		charging infrastructure availability (technical)	45 (3.3%)
		charging infrastructure(general)	43 (3.1%)
		charging infrastructure availability (physical)	26 (1.9%)
range	135 (9.9%)	range	135 (9.9%)
charging technology	119 (8.7%)	charging interfaces and technologies	55 (4.0%)
		charging speed	30 (2.2%)
		battery (other)	29 (2.1%)
		range extender	5 (0.4%)
environment & health	71 (5.2%)	environmentally friendly car usage	39 (2.8%)
		environment & health (other)	29 (2.1%)
		environmentally friendly car production	3 (0.2%)
society	283 (20.7%)	politics	171 (12.5%)
		desire for e-mobility	112 (8.2%)
other	109 (8.0%)	other (miscellaneous)	60 (4.4%)
		definable	39 (2.8%)
		joke	10 (0.7%)

6.1.2. Choosing preprocessing & algorithm. In order to implement the models suggested in section 5, we use the Python programming language and the well-established machine learning package *scikit-learn* [50]. Once we have gathered and labeled the tweets, we develop a program to systematically evaluate and choose the best suited kind of preprocessing steps. We propose several kinds of possible language processing methods: For word stemming, we use the *SnowballStemmer* [51] for German of the package NLTK [54], a natural language toolkit for Python. Lemmatizing is done using *TextBlob* [53], a Python package built on top of *NLTK*. We also develop own methods to replace emoticons, URLs and usernames with the words “emoji”, “url”, and “name” using regular expressions. Apart from the language processing, we take different tokenization methods into account.

We use the standard *scikit-learn* tokenizer, which by default removes all tokens that consist of one character only and considers any kind of punctuation characters as word separators. This is the reason why we replace emoticons, URLs and usernames, instead of just removing them—as they would get removed by the tokenizer anyway. In addition to the standard tokenizer, we implement two variations of this tokenizer: One which does not remove tokens with the length 1 and another one which always removes tokens with less than 3 characters. Moreover, a *TweetTokenizer* from *NLTK* is used. When it comes to the removal of words containing very little

information, we work with the German *NLTK* stopwords. Other variables we take into consideration are the lowest and highest threshold of frequency removal, or the value for *n* in n-grams. Over– and undersampling is done using the Python package *imbalanced-learn* [54]. For the tf-idf transformation of the feature vector, we use the built-in *tfidfTransformer* of *scikit-learn*.

Our goal now is to determine which of these methods promise the best overall model performance. We therefore perform a 3-fold CV (with equal class distributions for each fold) with every reasonable combination of all of these methods, including the possibility to not use a step, such as no stop word removal. Since we cannot yet determine which classification algorithm to use, we do it for both a SVM and a RF Classifier—without tuning their hyperparameters at this point. Generally working with separate binary classification models, we run this process for each need category individually. As described in section 5, we split this part into two runs. We use random seeds whenever it is necessary, in order to be able to shuffle or randomly split the data, while maintaining comparability among the two runs.

In a first run, we consider whether to remove stop words, downcasing, removing words that appear in less than 1% or 5%, and in more than 50% or 75% of the documents, uni-grams and bi-grams, the *NLTK TweetTokenizer* against the *scikit-learn* default tokenizer, stemming, lemmatizing and different combinations of emoticon-, url- and name-replacement. For each need category and for each classification algorithm, we make a list of the ten best performing combinations in regard to the F_1-score. Based on a majority of occurrences in the top ten lists of the first run, we decide to always use stop words, lowercase, stemming, no lemmatizing, uni-gram, as well as emoticon- url- and name- replacement. The results are not consistent in terms of the classification algorithm, tokenizer and frequency removal, which we therefore keep in the set of possible methods for the second run. In the second run, we additionally try over- and undersampling as well as tf-idf / no tf-idf transformation of the feature vector. Again, for every need category, we calculate the performance (F_1-score) of all possible combinations and rank them. As we consider each need category as one binary classification problem, we have to perform this task for each need category separately.

Interestingly, the rankings are slightly different for each need category, meaning that each need category has its own set of preprocessing steps that leads to the best scores. Nevertheless, we are able to single out the best preprocessing combinations they have in common, in order to have a general preprocessing that we can apply to every tweet, no matter which need category it is assigned to. Clearly, as we generalize the preprocessing across all need categories, we cannot use the individual preprocessing that would be best for each need category on its own. In addition to the steps selected based on the first run, our pre-treatment in the end includes a tf-idf transformation and oversampling. We do not perform any frequency removal, however, the best tokenizer removes all tokens with a single character. With this preprocessing, a SVM in terms of algorithm choice yields to better scores.

6.2. Model error estimation

To gain evaluation scores of the upcoming grid search, we define a parameter search space and perform a nested CV. This allows us to estimate the model error of the following grid search performed to tune hyperparameters of the SVM. Concretely, we optimize the C-value, the gamma, the kernel and the degree (in case of a polynomial kernel). We consider the preprocessing elements from the previous step as given and use the same implementations. In an outer loop, we split the data into ten outer folds and perform a grid search on nine folds with three inner folds. This results in a *best parameter set*, which we use to train a new classifier. This classifier is then tested on the hold-out set of the outer loop. Ten outer folds result in ten iterations through the outer loop, each iteration with its own results on the hold-out data set. Lastly, we determine the mean, the standard deviation and the minimum and maximum values of these ten scores.

6.3. Model deployment

After the model error estimation step, we perform a final grid search with the same parameter grid as in the previous step using the built-in *scikit-learn* grid search, and train the final classifier on the whole data set. As we use a SVM as classification algorithm, this classifier would only be able to determine whether a tweet contains a need category or not. We hence take advantage of *scikit-learn*'s option to additionally carry out the Platt scaling algorithm [55] in order to be able to calculate the *probability* of a tweet containing a need category later. As we want to make the classifiers publicly available for integration into other tools, we store the classifiers on a web server. To make them accessible, we implement a RESTful web service, using the Python *Flask* web framework.

7. Evaluation

The evaluation is divided into two parts: First, we focus on the performance measures of the model error estimation and the final grid search. Second, we point out the real-world usability of the web service.

7.1. Model error estimation

As stated before, we conduct a nested CV for each need category to obtain a model error estimation of the possible classifiers. The ten outer folds of this nested CV result in ten different F_1-scores of different classifiers on unseen data. Table 3 shows the minimum, maximum, mean and standard deviation of these ten scores for each need category. The minimum and maximum values compose the confidence interval for the F_1-score of the finally deployed classifier. Indeed, as in the final training, we use the same parameter space as in the nested cross validation, the scores for each need category lie within the confidence intervals of the nested CV.

Table 3. F_1-scores of the nested CV for the model error estimation

Need	Min	Max	Mean	Standard Deviation	Base-line	Impro-vement
price	0.524	0.737	0.642	0.059	0.264	+143.18%
car charac.	0.308	0.600	0.471	0.089	0.199	+136.68%
charging infras.	0.719	0.871	0.783	0.043	0.353	+128.13%
range	0.538	0.917	0.721	0.122	0.199	+262.31%
charging enviro.	0.222	0.444	0.360	0.078	0.174	+106.90%
enviro. & health	0.125	0.800	0.543	0.203	0.107	+407.48%
society	0.452	0.746	0.543	0.080	0.333	+63.01%
other	0.171	0.457	0.278	0.079	0.167	+66.47%

The scores for *charging infrastructure, range, price and society* show that it is possible to automatically allocate major need categories from tweets. Being the harmonic mean of precision and recall, a high F_1-score means that the classifiers are in most cases able to correctly identify whether a tweet contains a need category or not. This can certainly add substantial value to companies and help innovation managers in their decision making. The results differ notably between the individual need categories. One influencing factor is that the class distributions for each need category vary: For the need category *environment & health* for example, only 66 tweets have been labeled as containing a need category, in contrast to 298 for *charging infrastructure*. Different class distributions among the need categories are also one reason why they have (slightly) different *best parameter sets*, since the nested CV and the grid search in the model deployment phase are conducted for each need category separately. Indeed, the parameters are optimal for each need category, but—in addition to differing class distributions—different classifiers with different (optimal) parameters might behave differently and are therefore another reason why the scores for each need category are heterogeneous. Nevertheless, further investigation on the influencing factors responsible for the varying results is required. The low score for the category *other* is consistent with the fact that this category is of very diverse content—making it difficult for the classifier to find the right patterns which determine whether a tweet belongs to this category or not.

7.2. Model deployment

After manually performing the necessary steps like data gathering, labeling and deciding upon the preprocessing, we successfully implement a stand-alone web service that is capable of automatically allocating a tweet to the probabilities of each need category in the specified domain. The REST API makes it in general convenient to embed this service into more sophisticated analytical tools. A user interface for innovation managers, allowing them to analyze their costumer needs, could be developed. Interesting aspects, such as the distribution of costumer needs over time, could then be visualized. As the web service calculates the probabilities of containing a need category, an element empowering the user to define a "probability threshold" could be placed on the user interface. The managers could then decide by themselves, "how certain" the estimates must be in order to be taken into account for the visualization. For them, this might be much more meaningful than the F_1-scores discussed in the previous section.

8. Conclusion

In the work at hand we explore the option to automatically identify and quantify customer needs from a predefined set. To achieve that, we first code over 1,000 German tweets containing customer needs in the field of e-mobility with a descriptive coding approach. With this labeled data at hand, we choose the preprocessing steps, estimate the model error and train different supervised machine learning models for predicting the need category of incoming, unseen tweets. We encapsulate this functionality into a web service, which allows an automatic prediction of eight need categories for the e-mobility domain. If

implemented into a larger analytical social information system, this web service can assist innovation managers and alike in their daily operations of researching and possibly prioritizing different innovation ventures.

The work has several limitations. We only explore a supervised approach and only do so for the field of e-mobility. Consequently, the customer needs need to be identified once before the automatic identification can be enabled and non-identified categories cannot be revealed. Additionally, new classifiers need to be trained if other domains are regarded outside of e-mobility. Also, the data set is rather small and for single categories the results of the error estimation show a high variance and therefore the possibility of model over-fitting. Especially for unstructured text data from social media, the choice of the appropriate preprocessing steps (including feature selection and dimensionality reduction) is critical but also challenging. Although we evaluate a broad range of preprocessing methods, a more detailed evaluation on how to tailor the preprocessing of unstructured Twitter data to the particular use case is needed. In any case, this requires manual evaluation and decision-making. On the technical side, we utilize the Twitter Streaming API, which allows us to receive all tweets containing the mentioned keywords as long as the fetched data does not exceed more that 1% of all tweets [30]. However, when receiving the tweets, our results are limited to the selected keywords as well as the ability of Twitter to identify the language—since our language identification relies on the received meta data. By doing so, we cannot capture tweets which are written in mixed language (e.g. English and German) as well as tweets from Germans tweeting in a different language (e.g. English). Furthermore, as we only examine tweets in German, future work needs to explore the adaptability to other languages.

Nonetheless, we are able to show the feasibility of supervised machine learning to provide a solution for the challenge of automatable need identification and quantification, which can provide businesses additional insight into the needs of their customer in the future. By providing a social information system which utilizes the proposed web service, innovation managers do not need to manually scan possibly relevant tweets anymore, as the system would automatically assign single tweets to a broader category. By doing so, we would allow them to continuously monitor the needs expressed on Twitter for a certain domain—without any manual effort. This could be a step change for the field of innovation management, since most traditional approaches of customer need identification and quantification lack in scalability and automation capabilities. The exploration of further options, including unsupervised approaches, yields an interesting and promising field of research.

9. References

[1] D. Limehouse, “Know your customer,” *Work Study*, vol. 48, no. 3, pp. 100–102, 1999.

[2] J. R. Hauser and A. Griffin, “The Voice of the Customer,” *Marketing Science*, vol. 12, pp. 1–27, 1993.

[3] A. Perrin, “Social Media Usage: 2005-2015: 65% of Adults Now Use Social Networking Sites – a Nearly Tenfold Jump in the Past Decade,” *Pew Research Center*, no. October, pp. 2005–2015, 2015.

[4] F. Misopoulos, M. Mitic, A. Kapoulas, and C. Karapiperis, “Uncovering customer service experiences with Twitter: the case of airline industry,” *Management Decision*, vol. 52, no. 4, pp. 705–723, 2014.

[5] Twitter, “About Twitter,” 2016, https://about.twitter.com/de/company/ press/milestones, last accessed 2016-08-02.

[6] R. Cuthbertson, P. I. Furseth, and S. J. Ezell, *Innovating in a Service- Driven Economy: Insights, Application, and Practice*. Palgrave Macmillan UK, 2015

[7] N. Kühl, J. Scheurenbrand, and G. Satzger, “Needmining: Identifying Micro Blog Data Containing Customer Needs,” *Proceedings of the 24th European Conference of Information Systems*, 2016.

[8] S. Gregor and A.R. Hevner, “Positioning and Presenting Design Science Research for Maximum Impact,” *MIS Quarterly*, vol. 37, no. 2, pp. 337– 355, 2013.

[9] J. H. Kietzmann, K. Hermkens, I. P. McCarthy, and B. S. Silvestre, “Social media? get serious! understanding the functional building blocks of social media,” *Business horizons*, vol. 54, no. 3, pp. 241–251, 2011.

[10] A. Bermingham and A. F. Smeaton, “On using twitter to monitor political sentiment and predict election results,” 2011.

[11] R. Chen and M. Lazer, “Sentiment analysis of twitter feeds for the prediction of stock market movement,” *stanford.edu. Retrieved January*, vol. 25, p. 2013, 2013.

[12] D. Lee, O.-R. Jeong, and S.-g. Lee, “Opinion mining of customer feedback data on the web,” in *Proceedings of the 2nd international conference on Ubiquitous information management and communication*. ACM, 2008, pp. 230–235.

[13] H. Chen and D. Zimbra, “Ai and opinion mining,” *IEEE Intelligent Systems*, vol. 25, no. 3, pp. 74–80, 2010.

[14] A. N. Srivastava and M. Sahami, *Text mining: Classification, clustering, and applications*. CRC Press, 2009.

[15] N. Kühl and M. Goutier, “Needmining: Evaluating a whitelist-based assignment method to quantify customer needs from micro blog data,” *Operations Research Proceedings 2016—Selected Papers of the Annual International Conference of the German Operations Research Society (GOR)*, August 30 - September 2, 2016 2016.

[16] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[17] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 6.

[18] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment classification using machine learning techniques,” in

Proceedings of the ACL- 02 conference on Empirical methods in natural language processing- Volume 10. Association for Computational Linguistics, 2002, pp. 79– 86.

[19] P. D. Turney, "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*. Association for Computational Linguistics, 2002, pp. 417–424.

[20] V. Vaishnavi and B. Kuechler, "Design Science Research in Information Systems Overview of Design Science Research," *Ais*, p. 45, 2004.

[21] K. Peffers, M. Rothenberger, T. Tuunanen, and R. Vaezi, "Design Science Research Evaluation," *Design Science Research in Information Systems. Advances in Theory and Practice*, pp. 398–410, 2012.

[22] K. M. Sheldon, A. J. Elliot, Y. Kim, and T. Kasser, "What is satisfying about satisfying events? testing 10 candidate psychological needs." Journal of personality and social psychology, vol. 80, no. 2, p. 325, 2001.

[23] P. Kotler and G. Armstrong, *Principles of Marketing*. Prentice Hall, 2001, vol. 9.

[24] J. A. Harding, K. Popplewell, R. Y. Fung, and A. R. Omar, "An intelligent information framework relating customer requirements and product characteristics," *Computers in Industry*, vol. 44, no. 1, pp. 51– 65, 2001.

[25] E. Hull, K. Jackson, and J. Dick, *Requirements engineering*. Science & Business Media, 2010. Springer

[26] J. Scheurenbrand, C. Engel, F. Peters, and N. Kuehl, "Holistically Defining E-Mobility: A Modern Approach to Systematic Literature Reviews," in *Proceedings of the First Karlsruhe Service Summit Workshop*, Karlsruhe, Germany, 2015, pp. 17–27.

[27] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment." *ICWSM*, vol. 10, no. 1, pp. 178–185, 2010.

[28] D. Maynard, K. Bontcheva, and D. Rout, "Challenges in developing opinion mining tools for social media," *Proceedings of the@ NLP can u tag# usergeneratedcontent*, pp. 15–22, 2012.

[29] R. Hirt, N. Kühl, and G. Satzger, "An end-to-end process model for supervised machine learning: From problem to deployment in information systems," in *12th International Conference on Design Science Research in Information Systems and Technology (DESRIST)*. Springer, 2017.

[30] Twitter, "Twitter streaming api," 2015, https://twittercommunity.com/t/best-solution-for-fetching-tweets-mentioning-hundreds-of-terms/28294, received on 12-18-2015.

[31] A. Bifet and E. Frank, "Sentiment knowledge discovery in twitter streaming data," in *International Conference on Discovery Science*. Springer, 2010, pp. 1–15.

[32] O. Kvaløy, P. Nieken, and A. Schöttner, "Hidden benefits of reward: A field experiment on motivation and monetary incentives," *European Economic Review*, vol. 76, pp. 188–199, 2015.

[33] J. Saldaña, *The coding manual for qualitative researchers*. SAGE Publications, 2015.

[34] G. I. Webb, M. J. Pazzani, and D. Billsus, "Machine learning for user modeling," *User modeling and user-adapted interaction*, vol. 11, no. 1, pp. 19–29, 2001.

[35] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[36] C. X. Ling, J. Huang, and H. Zhang, "Auc: a statistically consistent and more discriminating measure than accuracy," in *IJCAI*, vol. 3, 2003, pp. 519–524.

[37] M. Kuhn and K. Johnson, *Applied predictive modeling*. Springer, 2013, vol. 26.

[38] D. M. Powers, "Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[39] C. Silva and B. Ribeiro, "The importance of stopword removal on recall values in text categorization," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 3. IEEE, 2003, pp. 1661– 1666.

[40] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," *Ann Arbor MI*, vol. 48113, no. 2, pp. 161–175, 1994.

[41] N. O. Andrews and E. A. Fox, "Recent developments in document clustering," Technical report, Computer Science, Virginia Tech, Tech. Rep., 2007.

[42] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: a comparison of retrieval performances," *Lecture Notes on Software Engineering*, vol. 2, no. 3, p. 262, 2014.

[43] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 853–867.

[44] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.

[45] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[46] D. C. Montgomery, *Design and Analysis of Experiments*, eighth edi ed. Hoboken, Arizona: Wiley, 2013.

[47] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, 2011, pp. 2546–2554.

[48] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," 2003.

[49] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *Journal of Machine Learning Research*, vol. 11, no. Jul, pp. 2079–2107, 2010.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, and others, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[51] M. Porter, "Snowball: A Language for Stemming Algorithms." 2001, http://www.snowball.tartarus.org/texts/introduction.html, last accessed 2017-06-01.

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

[53] S. Loria, "TextBlob," 2017, http://textblob.readthedocs.io/en/dev/index.html, last accessed 2017-06-02.

[54] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[55] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.