

Technical Write Up - Group 1

Agnes Pilch, Bavly Shenouda, Maximilian Pfleger, Janik Schnellbach

Abstract *We, a group of TUM students, addressed the issue of optimizing an SAP Support Ticket System in the context of a practical course in Machine Learning (ML). This was attempted by conducting a full ML pipeline on the given dataset with multiple ML approaches such as various types of regression, random-forest classifiers, linear SVC, auto-encoded clustering, and the classification of NLP embeddings. Out of which the classification based on NLP embeddings delivered promising accuracy of 85.2% for the support level classification of German and English tickets, while the rest could not live up to our hopes. The main goal was to increase the processing efficiency of an incoming ticket. This included the pre-classification of tickets into their respective support ticket level and the prediction of time. Pre-classification means that we tried to classify a ticket with only the information given upon the creation of one. We relied heavily on Natural Language Processing Transformers like BERT, which are pre-trained neural networks since most of the information on a ticket is condensed in text format. We also took it upon us to discover new insights on the data through data exploration and were able to aggregate and visualize our results in the form of a Tableau dashboard.*

Project Topic Clients using an SAP system hosted by TUM can request support via the SAP Ticketing Support System. In this context, the client seeking help is often overwhelmed by the number of options and categories in the ticket system such that they don't specify the problem sufficiently to allow efficient support. Sometimes, it also happens that customers do not create a ticket at all but only send an email. Since the problems of the clients range from legal issues and technical problems to financial issues and acquisition requests, a proper categorization and suitable delegation of the problem is crucial to handle the tickets effectively. The support is divided into two levels and multiple fields of specialty. The first-level support serves as an entry point and delegates more complex and subject-specific requests towards the second-level support.

Motivation Clients often don't bother to read Frequently Asked Questions (FAQs): they still create tickets and write emails. This causes a large waste of time for both the client and the support team. Beyond that, the first-level support often has difficulties relaying a ticket that may or may not be an urgent issue to the higher-level support if the necessary information is lacking, which is often caused by poor client description and categorization. Thus, the ticket keeps going back and forth between the first-level support and the customer to obtain the necessary information that is required for further processing. The non-uniform structure of accepting either tickets or emails for customer support is also a major problem, that leads to inconsistency in data and increased overall complexity for the support team.

Data The most significant and usable data comes from the history of completed support tickets. They may be exported in a structured format that contains information about complexity and problem categories for each ticket to support the foundation of the project. It could be challenging to extract useful information from the tickets properly, as a lot of information lies within the corresponding text messages between support staff and clients. However, this problem can be faced with the proper administration of the tickets. Emails sent by the clients, on the other hand, could also be a source of data but don't fit the basic structure of the tickets, making it hard to automate the processing. In this case, it might be smarter to forgo the emails, since the effort spent on manual preprocessing would be far too great. The support team may also have valuable expertise to share.

Data Transformation

Imputation While we dropped five columns ('Geändert Am', 'Geändert Von', 'Support Team', 'Erste Antwort', 'Letzte Antwort'), we were required to impute missing values in six cases. For the features 'Meldender' and 'Auftraggeber', we replaced empty values with 'EMPTY'. In the case of the remaining four features, we were able to identify that a missing value signifies that the respective ticket has been raised automatically. Hence, we substituted it with 'AUTOMATED'.

Feature Extraction For further analysis and problem modelling, we derived the following features:

initial_message	All messages in text form of the client until the first reply from the support side
embeddings	BERT embeddings of the initial_message
similarity	Cosine similarity of the initial_message compared to FAQ questions
difference	Euclidean distance of the initial_message compared to FAQ questions
num_editors	The total number of support side editors that worked on a ticket
time_taken	The total time it took to process a ticket in hours
faq_index	The index pointing to the most similar faq question
language_flag	A flag indicating the language of the ticket, which is either German or English

Table 1: A list of the features we have created

NLP Embeddings and Similarity Comparison Since most of the information in the dataset was condensed into the text attribute, we decided to use natural language processing to deal with it, as models need numerical values to work properly.

We decided to use BERT's pre-trained Sentence Transformer to embed the initial_message and the FAQ questions into numerical arrays of the same length. Missing shorter/longer sentences were padded/cropped to achieve the same array length. These numerical arrays are called embeddings. With the embeddings as a vantage point, we could now use similarity metrics such as the cosine similarity or the Euclidean-distance to calculate the similarity of the initial_message to each of the frequently asked questions. The evaluated cosine similarity yields a continuous number between 0 and 1, with a higher value indicating a higher similarity as well.

Data Visualization & Exploration We initially relied on Pandas and Matplotlib to gain deeper insights into the data and the relationship between several features. While 47.8% of the tickets have been raised automatically, the top 50 requesters ('Auftraggeber') and ticket raiser ('Meldender') account for 69% and 47% of all tickets respectively. A Pearson Correlation Matrix refuted initially assumed linear relationships between the number of editors and the number of messages. Furthermore, an increase of the average ticket time (excl. alert tickets) could be observed. To provide a more dynamic possibility to gain insights, we created a Tableau dashboard containing the most relevant charts.

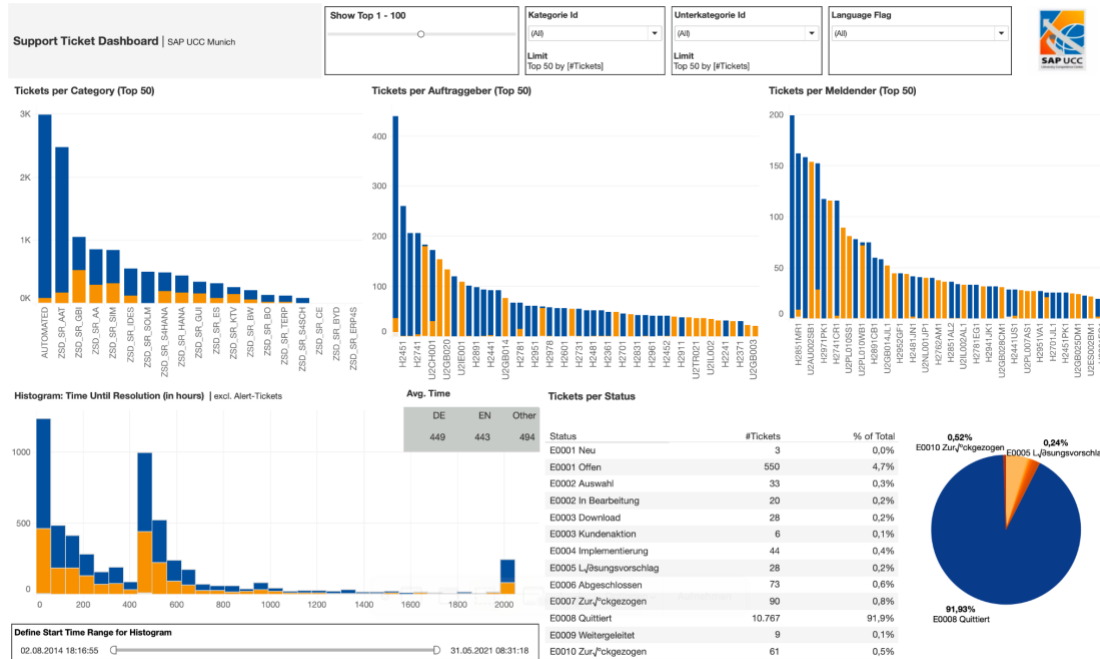


Figure 1: Support Ticket Dashboard

Machine Learning

Modeling was a long process of trial and error, trying to find meaningful relationships within the data, for every different use case a different set of assumptions had to be introduced and building upon them we have tried to use different model types to explain certain features.

1. Predicting the processing time of an incoming ticket

We thought that if we can manage to predict the time a ticket will take once it is issued that will help the team distribute the workload better and our first intuition is that there might be an underlying relation between certain users/companies and the level of complexity of the ticket. We included all the features that are available on ticket creation plus the description text embeddings. In addition, we used the similarities and differences indicators extracted from the text and compared them to the FAQ as it may indicate difficulty. In total, we had 778 features. Our approach was finding a model that can over-fit the data, and then from there try to use regularization and see how it performs on test data. We started by training a small neural network and a linear regression model and the performance was very underwhelming and we had a very low R2 score on the training data. From there, we decided to explore polynomial regression, but knowing the polynomial will scale the number of features exponentially, it was hard to run such a model with the 778 features on a consumer level computer, so we had to first use PCA and reduced the variance to 95% only to minimize the loss of information, yet that already scaled down the number of features to 144 features only which is within the range that a normal computer was able to run polynomial regression on. Polynomial regression easily succeeded in overfitting the data with an R2 score of more than 0.99, but as expected the test score was very low. In order to fight back over-fitting and use regularization, we experimented with Lasso Polynomial Regression with multiple alpha (regularization factor). The higher the alpha the lower the training score and the higher the test score got. Nonetheless, we shortly arrived at a point where the training score was decreasing with

no gain in the testing score. After extensive trial and errors with the hyper-parameters, namely the PCA variance, degree of polynomial, and the alpha factor, we arrived at the conclusion that the time taken cannot be explained by the given features and there were no underlying relations.

2. Predicting the support level based on the number of editors

In order to distinguish between first and second level support tickets, we had to define the support level based on other features. As complex issues often require different support agents to work on a ticket, we used the newly created feature 'num_editors' for the definition. If one or less support agents appeared in the correspondence of a ticket, the ticket is classified as first level. In case more support agents were involved, it would be flagged as second level. As the FAQs are only available in German, we only accepted the similarity measures to be meaningful for German tickets. That's why we only fed German tickets into our model in the first place. Our 80/20 split for training and test data set yielded to a training input of 7651 tickets with 15 features. For testing purposes, 1913 tickets remained. All categorical features have been encoded using One Hot Encoding. Thereupon, we used Linear Support Vector Classification and Random Forest Classification to predict the 'num_editors' (range: [0,6]). Only features that are available upon ticket creation have been used within this modelling approach. This is required from a business perspective as the support level classification should happen directly after a ticket has been created. Both approaches did not lead to the expected results (see table below). We additionally followed the above approach for English tickets as BERT was initially trained on English Wikipedia articles. This change led to the training-/test dataset shrinking to 2012/503 tickets. These results are also contained in the below table.

Metric / Model	Linear Support Vector Classifier	Random Forest Classifier
German Tickets (Training: 7651; Test: 1913 tickets)		
Test Accuracy	68.17%	68.11%
Test Error (MSE)	0.4637	0.4705
English Tickets (Training: 2012; Test: 503 tickets)		
Test Accuracy	50.89%	49.11%
Test Error (MSE)	0.8350	0.8370

Table 2: Results of Linear SVC and Random Forest Classifier to predict number of editors

3. Predicting the support level based on categories

In contrast to the above-described approach, we have also attempted an alternative labeling, this time in a different fashion. Instead of looking at the number of employees, we made a distinction based on the "Kategorietext" as well as the "Unterkategorietext", which are attached to every ticket. Using the first level tasks file provided on Moodle, we were able to determine a general setup for the labeling. For example, we have set developer key or password topics as first level tasks whereas system availability or product realized questions have been placed as second level task. A more detailed breakdown can be found in our repository.

Building on these labels, we have then used a transformer once again. However, unlike before, we used the transformer as part of a larger neural network and included it into a training process, thus gradually adjusted its pre-trained weights according to our needs. We complemented the output of the transformer with a dropout layer and a subsequent linear layer with two output neurons

matching the two classes of first and second level support. The dropout helped us in making the network more stable and increasing the dispersion of significant words or tokens. The setup of our classifier is depicted in Figure 3.

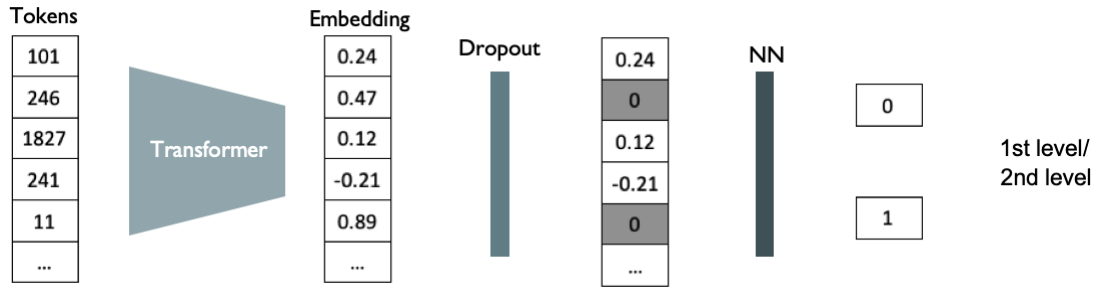


Figure 2: Topology of our support level classifier




Language		Accuracy	1 st level Accuracy	2 nd level Accuracy
Multilingual		85.2%	85.1%	85.2%
German		87.2%	86.8%	87.5%
English		89.7%	87.9%	91.1%

Table 3: Topology of our support level classifier

Based on the different ticket languages, we have set up three different models: multilingual, German and English. The corresponding results are included in Table 3. With almost 90% accuracy for the English tickets, and at least 85% for the entire nearly 4000 labeled tickets, it can be seen that the labeling approach as well as the classifier work well together.

4. Summary

Overall, we are quite satisfied with our work, especially considering the short period of time. We experimented with various ML models and were somewhat desperate at times. In the end, however, we found a successful model that even surpassed our expectations and also holds further potential for the future.

Challenges The available dataset on the tickets had a lot of missing values and was not labeled according to our needs, which led to us investing a lot of effort into the data preprocessing. Most of the crucial information of a ticket was condensed into text form, which we then replaced with numerical BERT embeddings. It was difficult to work with an unlabeled and patchy dataset, since we did not know if we would be able to gain any insights from the ML models or recognize any correlations and reach a certain threshold of accuracy.

Outlook Looking forward, we would like to encourage the UCC to collect data over a longer period of time with clear classification into first and second level support. While we were able to achieve promising results with our approach, this form of labeling/classifier is liable to get “trapped” in certain topics. Even very simple questions on second level support topics (which could be answered easily) unfortunately do not end up at first level support. Therefore, we need a refined labeling process with reliable and comprehensive data to exploit the enormous power of the language transformers.