Automated Ticketing System
Author: Dominik Macko

Introduction

Every company needs a good system to handle customer problems and questions. This system is called a ticketing system. In many companies, people do a lot of the work in this system, like deciding which team should handle a ticket or guessing how long it will take to solve a problem. But what if an automated system could help with these tasks?

In this project, I wanted to see if machine learning could help make the ticketing system better. I tried to make the computer:

1. **Choose the Right Team for a Ticket**: When someone has a problem, it's important to send their ticket to the right team.
2. **Guess How Long to Fix a Ticket**: It's useful to know how long a problem might take to fix.
3. **Find Old Tickets that are Similar**: Sometimes, old problems can help solve new ones.

This report will talk about how I did these things as a proof of concept.

Methodology

These three tasks can be mapped into three separate tasks of natural language processing:

- Team assignment — multi-class classification of a short text data
- Speed prediction — regression of a a short text data
- Ticket similarity — similarity of two short texts, sometimes called sentence similarity, also bit related to recommender systems and text retrieval

Overall all of these tasks concern natural language processing, which is expected as our data was unstructured text. Now, to give a brief overview of the methods and tools

used in NLP:

1. **Classical Methods for NLP**: Earlier methods revolved around statistical techniques and rule-based approaches. Techniques such as term frequency-inverse document frequency (TF-IDF) and the Bag-of-Words model were widely used. They convert text data into a numerical form by representing each document as a vector.
2. **Word Embeddings**: With the evolution of deep learning, word embeddings became the cornerstone of NLP. Tools like Word2Vec, GloVe, and FastText offer pre-trained models that represent words in a dense vector form, capturing semantic meaning based on context. These embeddings allow models to understand relationships between words, like "king" is to "queen" as "man" is to "woman".
3. **Transformers**: Models such as BERT, GPT, and T5 are built on the self-attention mechanism, allowing for enhanced understanding of context. They've set records across various NLP benchmarks. Unlike traditional methods which look at words in isolation, transformers consider the entire context, making them incredibly effective for complex tasks. Additionally they are often used pretrained and further finetuned for our downstream tasks.

Overall the goal was to experiment with all of these and compare them.

Dataset

For dataset we used a “dirty” anonymized dataset. Overall we had 24141 records where each record contained ticket problem abstract. Out of these records, we had labels, specifically team it was assigned to and open/close dates for 4741 records. The rest 19402 records were unlabeled, thus they only contained the problem abstract.

We performed simple preprocessing of the dataset, specifically:

1. Basic data cleaning — remove record when a column is missing, calculate hours to resolve ticket, drop some noisy teams and unify others which are obviously supposed to be the same.
2. Merge labeled and unlabeled data while removing duplicates — some records appeared in both (this was determined by ticket id and equality of the texts)
3. Split labeled data with ratio 6:2:2 into train, validation, and test sets

Additionally, we analyzed the texts for stopwords and determined classical english stopwords shold be sufficient. We also analyzed lengths of texts in chars and tokens. The maximum of character size was 250 characters with mean close to 50. This meant that there would be no issues with long texts for Transformer models.

Team Assignment

For team assignment we decided to compare baselines, classical algorithms that require structured data, word embeddings, and Transformer model. Overall, we chose to tune the metric of f1-score micro-averaged as we were dealing with multi-class classification which was imbalanced.

Classical Structured Algorithms

Here we used old school approach of preprocessing the text by splitting into lowercased tokens, removing stopwords, converting it into tf-idf matrix and then using algorithms that can use with such data. Specifically, we used:

- Logistic Regression
- Linear Support Vector Machines
- Complement Naive Bayes — as it can work well with imbalanced classes
- Random Forest

First three algorithms were picked as they can allow us to linearly model the data. Random Forest was picked for its ability to model complex relationships. The optimal parameters of models were found using grid seach with 5-fold cross validation with goal of minimizing our metric.

Word Embeddings

Here we used pretrained word2vec and GloVe word embeddings. We did very simple preprocessing of text by splitting it into lowercased tokens. Then we averaged word embeddings for all the tokens for each text. Subsequently, we used the same algorithms as previously with tf-idf matrix. We also tuned hyperparameters of models as in previous task.

Transformer

Here, we experimented with few Transformers but in the end used Roberta base size. But more specifically because of the unlabeled data we leveraged also masked language modeling to use this data. Thus, we first split unlabeled dataset with ratio 9:1 into train and validation and finetuned the model here using masked language modeling with early stopping. After that, we finetuned the model for classification task using our previous train and validation sets. We also leveraged early stopping here.

Evaluation

For team assignment we decided to compare baselines, classical algorithms that require structured data, word embeddings, and Transformer model. We compare metric of f1-score micro averaged. For baselines we use model predicting majority class and other model giving stratified prediction. Overall for each word embedding model we only provide the better one out of word2vec and GloVe

Model	F1 micro averaged
Naive Bayes	0.75
Transformer	0.75
Linear SVM	0.73
Word2Vec Linear SVM	0.61
GloVe Random Forest	0.58
GloVe Logistic Regression	0.56
Random forest	0.56
Logistic regression	0.56
GloVe Naive Bayes	0.54
Baseline majority class	0.33
Baseline stratified	0.18

In this evaluation we can see that all the models beat the baselines. However, surprisingly the old school models performed the best with Transformer. While Transformer had same score as simple Naive Bayes, it was much harder to train the model and in the future mantain. These results can most likely be explained by not having much data.

Speed Prediction

For speed prediciton we decided to compare baselines, classical algorithms that require structured data, word embeddings, and Transformer model. Overall, we chose to tune the metric of root mean squared error as it is standard metric for regression tasks.

Classical Structured Algorithms

The preprocessing is the same as in task of team assignment. However, we used models:

- Linear Regression with L1 and L2 regularization (Elastic Net)
- Linear Support Vector Regression
- Random Forest Regression

The first 2 models were chosen to see if we can model the data based on linear relationships only. Random Forest was chosen for more complex modeling. We used 5-fold cross validation with grid search to find optimal hyperparameters.

Word Embeddings

We used the same approach here as in team assignment with same algorithms as in previous section.

Transformer

Here, we experimented with few Transformers but in the end used Roberta base size. We leveraged the model that was already finetuned on task of masked language modeling from team assignment. After that, we finetuned the model for regression task using our previous train and validation sets. We also leveraged early stopping here.

Evaluation

We evaluated based on root mean squared error. We additionally used two dummy models, where one predicted always mean and other predicting median. Table below shows the results. Overall, word2vec embeddings were worse than GloVe so we do not include them.

Model	RMSE
linear regression	2125
linear svm	2193

random forest	2204
glove linear regression	2252
glove linear svm	2323
glove random forest	2311
dummy mean	2375
Transformer	2411
dummy median	2474

The results show that simple models performed the best. This can be also explained by lack of much data. However, surprisingly. Transformer model performed here even worse than baseline. Additionally, this is way worse compared to performance of the model on classification. We could not really find some reason why this is happening and just accepted that Transformer might not be suited for this specific task, even tho it is suited for classification.

Ticket Similarity

In this task we aimed to find similar tickets. However, we lacked labels here and thus we used unlabeled, train, and validation datasets as our dataset. And we leveraged simple 10 tickets from test set for evaluation. A more rigorous evaluation would require annotators labeling whole test set and possibly even using cross validation for evaluation.

Classical Approach

We leveraged tf-idf vectorization followed by cosine similarity and BM25 (also called “tfidf on steroids”) which is state of art in lexical text retrieval algorithm.

Word Embeddings

We leveraged both GloVe and Word2Vec embeddings averaged over tokens followed by cosine similarity.

Sentence Transformers

We leveraged GTE Sentence Transformer Bi-Encoder. We chose this model based on Massive Text Embedding Benchmark Leaderboard. We wanted one of the best

performing model for STS (semantic textual similarity) in base size. We used the model to embed the whole ticket problem abstracts and then used cosine similarity.

Evaluation

For proper evaluation we would use metrics such as MAP@10 (Mean Average Precision at 10). But this would require labeling data. Since our evaluation sample only consisted of 10 tickets, we resorted to doing just qualitative evaluation and counting how many similarities made 0 sense, and how similar algorithms were. The results of the algorithms were often very similar with just different order.

Overall results from bm25 and tfidf were very similar as was expected. For example, recommending ticket with credentials for ticket which is asking for credentials for siemens v3 energy was not very useful, see image.

```
('Credentials for new customer Siemens Energy',
          ticket                                    problem_abstract
 8375   311658856                                      Siemens Energy
 10931  309825371                          New Security credentials
 8948   312612394                                SNMPv3 Credentials
 105    306487733  Please cancel OL 7862952 / Advanced Energy Ind...
 8862   308106279                     Security Credentials Creation
 11440  314373286  Additional Credentials for Sysco Corp Customer...
 9146   308052313                              Credentials for Sally
 8307   313576459                   SNMP credentials for Eli Lilly
 3629   310027961        ?*OGE ENERGY FLD<WFA/LEC# KI000618>TST TX-1
 8660   310493396                        Credentials for Micro Focus)
```

But very similar recommendations were from Transformer but also embeddings. Overall while evaluating this, I found out that I am not sure how to evaluate this. The ticket dataset is not very clean and meaningful and labeling by some experts (and or with more metadata) would be actually needed. However, based on quality inspection it seemed like the Sentence Transformer model was performing the best. This could be for example seen when it wouldn’t get confused by words config and configuration meaning the same thing as can be seen below.

```
('Kyndryl config backup is missing from the directories',
          ticket                                    problem_abstract
 10626  308097953  Configuration backup is not working for specif...
 7183   312861812             Configuration is not getting backed up
 7902   308829455             don't see backup config in action-tool
 11021  315300651        Missing configuration backup file on poller
 7542   309686335         Missing backup config for multiple devices
 8491   307412874     Running configuration not backed up in action.
 8009   308524680  Backup configuration file of device not in ACTION
 10593  314298897  unable to pull older backup configs from the p...
 7516   309799931  URGENT!!! - Action config backup not working p...
 7517   309799776  URGENT!!! - Action config backup not working p...)
```

Conclusions

In this project we experimented with proof of concept for simple automated ticketing system. We investiaged ways to assign ticket to a team, predict the hours it will take to resolve the ticket, and lastly finding similar already solved tickets. For all of these tasks we experimented with basic approaches that rely on exact matching of tokens using tf-idf vectorization, word embeddings, but also Transformer architectures.

Future Improvements

As is expected, there are many more improvements that could be done in this project:

- clean the data more,
- obtain bigger dataset,
- try out more algorithms,
- try some expert rules to improve models or create rule-based models,
- experiment with Cross-Encoders and supervised finetuning for ticket similarity,
- do proper evaluation with annotators for ticket similarity,
- create proof of concept application using Gradio.

Future Work (Tasks)

It is also important future work that could be done to expand the automated ticketing system. For example:

- sentiment of ticket problems could be analyzed,

- clustering could be used to find meaningful clusters,
- topic modeling could be used to find topics occurring typically,
- LLM such as GPT or LLama could be used to interpret and summarize the clusters and topics,
- LLM could further enhance ticket similarity and straight out based on some additional knowledge base answer question what to do with this new ticket.