K12 *Insight*

Template Recommendation Engines

By
Yanling Sun
Yichen Peng
Kaitong Hu
Xingchen Wang

Supervisor: Ming-Long Lam

A Capstone Project
Submitted to the University of Chicago in partial fulfillment
of the requirements for the degree of
Master of Science in Applied Data Science

Division of Physical Sciences

Dec 2023

Abstract

The Let's Talk platform by K12 Insight, designed to enhance communication between K-12 schools and stakeholders, faces challenges with its template feature, leading to inefficiencies and compromised response quality due to a cumbersome selection process and occasionally inaccurate suggestions. In response, an enhancement is proposed to efficiently surface relevant templates, aimed at improving response efficiency and fostering trust between schools and stakeholders. Our approach involves developing a template recommendation engine that utilizes collaborative and content-based filtering techniques. This engine evaluates the similarity between customer queries and templates and also between different queries. This solution has been implemented in a Streamlit web application, providing an interactive and user-friendly experience for clients.

Keywords: Recommendation System, Collaborative Filtering, Content-based Filtering, Sentence Transformer, Logistic Regression, Question Template Matching, Natural Language Processing

Executive Summary

K12 Insight has developed the “Let’s Talk” customer service platform, a centralized hub designed specifically for K-12 institutions, to manage concerns and inquiries from its various stakeholders. While this platform has enabled school district administrators to manage and respond to customer inquiries in a timely manner, there are inherent challenges in the current template selection process. Users often find the process of choosing a pre-written text template from the database tedious, time-consuming, and ineffective.

Considering these challenges and the importance of customer satisfaction, the primary goal of our work is to enhance the “Let’s Talk” platform by introducing an intelligent template recommendation engine. This engine, underpinned by both collaborative filtering and content-based filtering approaches, is designed to reduce the time and effort required for users to select appropriate response templates. We aim to recommend text templates based on the input queries from customers by leveraging advanced techniques, including Natural Language Processing (NLP) and Large Language Models (LLMs). Specifically, we will evaluate the semantic and Jaccard similarity between the query and the template as well as the similarities between different queries using data provided by K12 Insight.

The implementation of this recommendation engine is expected to transform the way users of the “Let’s Talk” platform interact with the customers. By providing high-quality and relevant template suggestions, we will be able to enhance the user experience, increase stakeholder engagement, and ultimately, foster trust between school district administrators and their stakeholders.

Table of Contents

List of Figures

List of Tables

Introduction

Problem Statement

Owing to the heightening focus on the academic and social-emotional well-being of students, K-12 educational institutions receive a growing number of concerns and inquiries from their students, parents, and community members every day. K12 Insight's Let's Talk customer service platform is designed specifically for K-12 institutions to facilitate communications between schools and their various stakeholders. This centralized communication hub aims to enable users, such as school district administrators, to effectively manage, track, and respond to customer inquiries, while providing a positive and responsive experience for all interested parties. As it stands now, users of this platform resolve customer inquiries by either manually selecting a pre-defined text template from the database for the corresponding district or spending extra time typing a new answer when no suitable template is available. While the template feature helps ensure consistency and increase productivity in making replies, several problems are worthy of attention:

1. The template selection process is time-consuming and inefficient because users need to go through every single template to find the most appropriate one.
2. Relevant templates can be overlooked by users.
3. Searching templates by keywords sometimes yields incorrect template suggestions.

The aforementioned issues lead to an inefficient communication channel. This can be resource-intensive for staff, resulting in delays in responding to concerns or feedback, especially when users are handling a high volume of inquiries. Moreover, the current business workflow fails to offer consistent quality in terms of the responses provided, which can potentially lead to customer frustration and negatively impact the credibility of educational institutions.

Acknowledging that meeting the customers' satisfaction is essential in building trust between schools and their stakeholders, as well as sustaining the company's revenue growth, K12 Insight is eager to introduce a new feature on its Let's Talk platform to streamline the template selection process. By intelligently surfacing relevant templates, the time and effort required to manually search and select the templates will be significantly reduced. Users will be empowered to provide high-quality service to their customers, therefore enhancing customer experience and fostering trust between all parties involved.

Goals of Analysis

The overall goal of this project is to automate the template selection process to improve efficiency. To help streamline the process, we propose designing a recommendation engine that can suggest pre-written responses to commonly asked questions and various feedback types because the Let's Talk customer service platform currently lacks one. This system should be able to analyze the prompts collected from various digital channels, understand the content and intent behind it, and recommend appropriate pre-written response templates. We will leverage advanced and state-of-the-art techniques and methods, including but not limited to Natural Language Processing (NLP) and Large Language Models (LLMs), to identify patterns, extract keywords, and recognize named entities from both customer inquiries and existing text templates.

The implementation of such a system is expected to enable users of this platform to respond quickly and effectively to customers, while reducing the burden on staff and improving overall communication and engagement. Further to this, the system can provide a much more reliable way to handle a large volume of saved templates. Utilizing a robust recommendation

engine can not only save time but also reduce the probability of choosing a wrong response template.

Scope

In spite of the benefits discussed in previous sections, it is important to note some limitations of our proposed solution.

1. Generalizability: Users within a specific school district are constrained to select candidates from a set of pre-written templates exclusive to that district, suggesting that a model that works best in one district might not achieve comparable performance when applied to another district.
2. Language Limitation: Because the dataset we used to train our recommendation system will be in English only, the final product may struggle to accurately interpret input received in languages other than English.
3. Obsolete Template Suggestions: Since the recommendation engine will be trained on past customer inquiries and user responses, it might fall short of understanding the most up-to-date demands as customers' needs and preferences evolve over time.
4. Mixed Data Type: This recommendation system cannot work with input that involves a mixed data type, such as inquiries that contain any images.

Having a clear understanding of the constraints of this text template recommendation engine can assist K12 Insight in making well-informed decisions on whether this approach is the best fit for their requirements. Being fully aware of these limitations can lead to a more successful implementation.

Background

K12 Insight is a customer intelligence solutions provider for schools and school districts. Founded in 2002, the company is headquartered in Herndon, Virginia, USA. K12 Insight partners with school districts to enhance their educational services, interaction, and involvement with their students, using cutting-edge technology such as a customer experience platform and chatbot, research, and professional development. As a trailblazer in customer intelligence solutions for schools, K12 Insight has supported more than 400 school districts nationwide. Our project mainly focuses on K12 Insight's Let's Talk solution, which is a cloud-based customer experience and intelligence platform that enables school districts to manage and respond to feedback and inquiries from customers. This system aims to cultivate a positive school culture, establish trust, and encourage family and community engagement.

In our project, it is important to understand the definitions of "customers" and "users." Customers are people K12 Insight collects inquiries or feedback from, such as parents and students. These inquiries are received via different channels, including email, chatbox, and text messages. The content of those inquiries may vary from topics like school registration dates to the cafeteria menu. Users, on the other side, are people who utilize the platform to respond to those inquiries or feedback, and they are mainly school administrators or district officers. The main objective of our project is to help users match inquiries with pre-existing response templates to enhance communication efficiency so that users do not need to type the same response again and again.

Past efforts at solving similar problems can be found in the literature review. This project aims to contribute to the existing body of knowledge by developing a template recommendation

engine specifically tailored for K12 Insight's Let's Talk solution. By doing so, we hope to enhance the customer experience for school districts and further improve education outcomes.

Literature Review

Question matching and question-answer matching are the two main techniques we will use to match inquiries with pre-existing response templates. Question matching involves the process of matching new inquiries with similar or related questions that have been previously asked. This allows for reusing the pre-existing response templates that contain the most appropriate answers for those questions. On the other hand, question-answer matching involves a more direct and focused approach to identify the most relevant answers to a specific question based on the context and intent of the question.

Back in 1965, Semantic networks and rule-based systems were two mainstream techniques used for question-answer matching (Simmons, 1965). A semantic network is a way of representing knowledge using a graph structure, where nodes represent concepts or entities and edges represent relationships between them. This is used to identify synonyms and antonyms and for disambiguating words with multiple meanings. For the rule-based system, it used a set of syntactic and semantic rules to generate responses to questions. One famous system at that time was the Base Realization System (BRS), however, BRS was limited by the explicit rules and knowledge to function effectively.

Around 2000, to resolve the limitations of traditional rule-based systems, including relying on human experts to manually create rules and patterns to match questions and answers, predictive annotation was introduced. This technique involves using statistical models to identify and annotate relevant portions of text that are likely to contain the answer to a given question

(Prager, Brown, Coden, & Radev, 2000). However, predictive annotation does not take into account the contextual dependencies between words in a sentence.

Nowadays, as natural language processing (NLP) technology advances, more sophisticated models are developed that can be used for question matching, leading to improved accuracy and performance. Regardless of which matching method we need to use, our main consideration lies in capturing the semantic meaning and long-term dependencies. Understanding the semantic meaning allows for better comprehension of the intent and content of a question and its corresponding answer. By considering semantic meaning, the matching algorithm can focus on capturing the intended semantics rather than relying solely on surface-level patterns or keywords. Long dependencies refer to the relationships between words or phrases that are distant from each other in a sentence but still connected grammatically or semantically. These dependencies can be crucial in understanding complex questions or statements that involve multiple subjects. Based on a more recent literature review, we deliberatively selected four algorithms as our candidate models:

1. BERT (Bidirectional Encoder Representations from Transformers): A transformer-based neural network architecture that can handle multi-faceted questions. BERT often excels in capturing the intricate nuances and complexities inherent in such queries.
2. LSTM (Long Short-Term Memory): LSTM is a Recurrent NN-based model renowned for its ability in capturing long-term dependencies. LSTM can effectively discern and incorporate contextual information spanning distances within a sequence.
3. QANet (Question Answering Network): This model adopts a convolutional neural network (CNN) based architecture. It is specifically engineered to handle long input sequences.

4. Siamese Neural Network: This algorithm places greater emphasis on capturing semantic information and handles flexible lengths of input.

Currently, as the era of deep learning emerges, it is more common to use convolutional neural networks (CNN) (Tan, Dos Santos, Xiang, & Zhou, 2016) and Long Short-Term Memory (LSTM) (Dhakal, Poudel, Pandey, Gaire, & Baral, 2018) to encode questions and answers or pairs of questions and match them based on their semantic similarity. The most common metric for semantic similarity is cosine similarity since it is computationally efficient and is better suited for high-dimensional data, like text (Tan, Santos, Xiang, & Zhou, 2015).

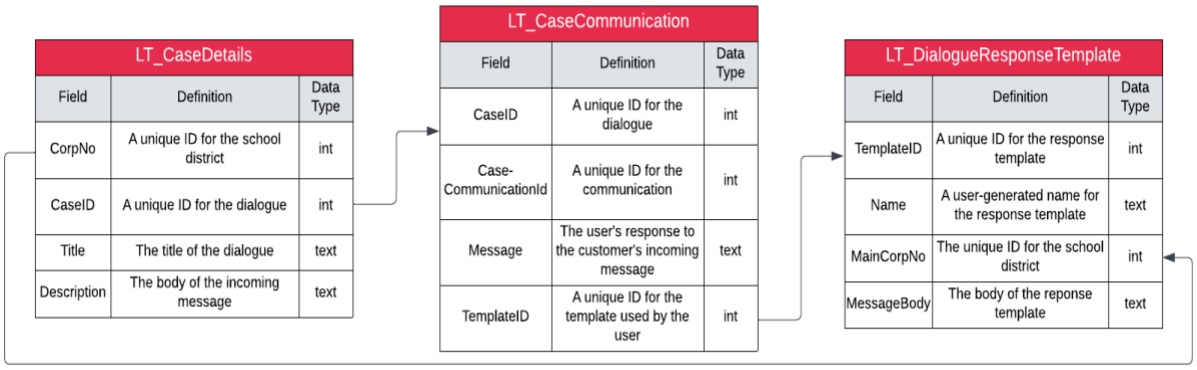
Those models can also be used to solve problems in industries other than education. For instance, the BERT-based financial sentiment index model can be used to analyze financial news articles and social media posts to extract financial sentiment scores. The LSTM-based stock return predictability model takes in historical stock price data along with the financial sentiment scores generated by the BERT-based model to predict future stock returns (Hiew et al., 2019). This highlights the potential of deep learning models such as BERT and LSTM in the field of finance and investment.

Data

Data Sources

The data was extracted from K12 Insight's internal database and was stored in an MS SQL Server. There are three main tables in the database:

Figure 1. *Table Schema*



1. LT_CaseDetails: This table captures the initial communication from the customer (i.e. parent/ student/ employee of the school district) directed to the school district. It serves as a repository of original inquiries. However, this table only captures the initial message and does not include subsequent interaction between the customer and the user of a school district.
2. LT_CaseCommunication: This table records the user's response to the customer's incoming message. There can be multiple interactions between a single customer and various district representatives.
3. LT_DialogueResponseTemplate: This table contains the pre-written response templates in HTML format, which might require text preprocessing techniques to extract pure text content for analysis.

For our project, we merged the abovementioned tables to identify customer inquiries that were answered with the pre-written templates. This shall shed light on the frequency of template usage and how these standardized templates were used to address specific types of customer inquiries.

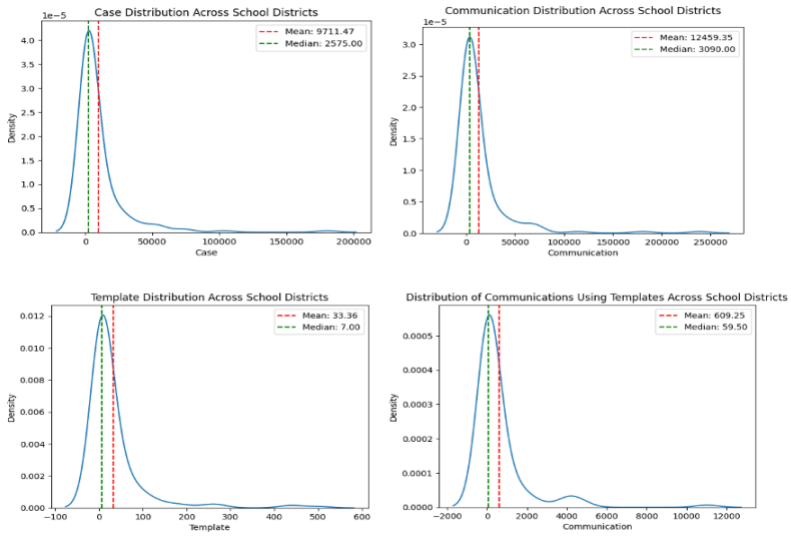
Descriptive Analysis

We used a wide range of analytical techniques to gain valuable insights and validate preliminary assumptions in our exploration of these datasets. The objective of this analysis was to understand the utilization and distribution of pre-written templates across different school districts.

The number of instances in the "LT_CaseDetails" and "LT_CaseCommunication" tables was reduced to 1.7M and 2.5M, respectively, after extensive data cleaning procedures, while the "LT_DialogueResponseTemplate" was left with 5971 relevant templates. Recognizing that one of the main problems in the current business process was the underutilization of the template feature, we suspected that a minuscule proportion of cases were solved using the pre-written templates. Our assumptions were supported by empirical evidence. A mere 4% of the total responses were found to be using these text templates. This imbalance in the data introduced considerable challenges in the development of effective models for our recommendation system.

In the context of our customer service platform, users within a specific school district are restricted to selecting templates available exclusively within their designated district. As such, a comprehensive understanding of the distribution of cases, communications, templates, and communications that were addressed with templates across all school districts becomes imperative. This information is crucial for us to identify the most representative districts upon which to base our model development and training. As illustrated in Figure 2, the distributions are notably skewed towards the right. This observation suggests that while a few districts were very active and had a multitude of pre-written templates, most school districts might have relatively fewer.

Figure 2. *Case&Templates Distribution Across School Districts*



Methodology

Feature Engineering

In this project, embeddings will be performed on templates and queries to help calculate similarities between queries and templates and similarities between queries and queries. Sentence Transformers will be utilized to derive embeddings.

Sentence Transformers

Sentence Transformers are a framework for computing dense vector representations of sentences. While traditional word embeddings, like Word2Vec or GloVe, represent each word by a fixed vector, sentence embeddings represent the entire sentence as a vector. This is especially useful for tasks where the meaning of the entire sentence or paragraph is more important than individual words.

The idea behind Sentence Transformers is to adapt the powerful Transformer models to efficiently generate semantically meaningful embeddings for sentences or paragraphs. This typically involves training transformer-based models (like BERT, RoBERTa, etc.) on various tasks such as sentence similarity comparison, paraphrasing, or other NLP tasks. Some key features and details about the Sentence Transformers are discussed as follows:

1. Architecture: Most Sentence Transformers are based on the Transformer architecture, especially the BERT (Bidirectional Encoder Representations from Transformers) model or its variants. These models are initially trained on a large corpus of text using tasks like masked language modeling (for BERT) or casual language modeling (for GPT).
2. Training: Sentence Transformers are typically fine-tuned for tasks, such as Semantic Textual Similarity (STS) or Natural Language Inference (NLI).
3. Usage: Once trained, you can pass a sentence (or even a paragraph) through the model to get a fixed-size vector representation. This representation can then be used for various downstream tasks like semantic similarity, clustering, classification, etc.
4. Efficiency: Sentence transformers are generally faster than using large transformer models in a pairwise comparison setting. For instance, to find the similarity between two sets of sentences, instead of doing pairwise comparisons with BERT, you can compute embeddings for each sentence with Sentence Transformers and then compute the similarity between the embeddings.
5. Libraries: There is a popular Python library named 'sentence-transformers' that simplifies the process of obtaining sentence embeddings. It provides pre-trained models, easy-to-use interfaces, and tools to fine-tune models on custom data.

In summary, Sentence Transformers offer an efficient way to capture the semantic meaning of sentences in a dense vector form, which can be beneficial for various NLP tasks.

Since both similarities between queries and queries and similarities between queries and templates will be calculated, two different pre-trained models will be imported for these two tasks. All-mpnet-base-v2 is used for performing embeddings for Q&Q tasks while multi-qa-mpnet-base-dot-v1 is for Q&A tasks.

Fine-Tuning Sentence Transformers

In the realm of customer service, the nature and structure of customer inquiries present a unique challenge for general Q&A tasks. The inquiries received on the Let's Talk platform exhibit greater length and complexity compared to the more concise and direct questions posted on forums, Quora, or other internet platforms. In addressing these characteristics of customer dialogues, we decided to fine-tune the multi-qa-mpnet-base-dot-v1 model, which was originally optimized for general Q&A tasks. Our objective was to generate more precise and contextually relevant embeddings that aligned closely with the nuances of our customer inquiries.

The process of fine-tuning involved using two distinct datasets: one comprising matched inquiry-response pairs and the other consisting of unmatched pairs. The matched set contained all customer inquiries that were successfully addressed using pre-written text templates in the past. Meanwhile, for the unmatched set, we strategically paired each inquiry in the matched set with two different text templates that were clearly unsuitable for the given context. This approach allowed us to generate a relatively balanced dataset with a positive-to-negative class ratio of approximately 1:2, effectively reducing the typical biases associated with training on imbalanced datasets.

The fine-tuned model was employed to extract embeddings from both customer inquiries and user responses. A subsequent evaluation of our recommendation system revealed that these fine-tuned embeddings have demonstrated a marked improvement in performance, evidenced by a slight increase in the recall rate. This improvement underscored the effectiveness of our approach in refining the model to better meet the specific demands of our customer service platform.

Scoring Mechanism

For each pair of query and template, three kinds of scores will be derived, and they will be used as the input of the model to classify whether this query and this template are a good match. The three kinds are listed below:

1. Query-template similarity score: The embeddings from multi-qa-mpnet-base-dot-v1 model will be used with dot product to calculate the similarities between query and template.
2. Collaborative Filtering Score: The similarities between this query and other queries from the same district will be calculated. The top 10 similar queries will be selected. If this query is a good match for template X, the collaborative filtering score between the query and template X will be added by the similarity between this query and the selected query. In this circumstance, embedding from All-mpnet-base-v2 will be used along with the dot product.
3. Jaccard Similarity Scores: The Jaccard similarity score between each query and template will be calculated. It is defined as the size of the intersection divided by the size of the

union of the sample sets, which measures how closely an inquiry aligns with a predefined text template.

Modeling Frameworks

Following the feature engineering process, a query-template matrix is obtained. Given the considerable number of templates and in pursuit of enhanced model performance, a distinct strategy is adopted. Instead of treating a single query as one data point for a multi-classification problem, we approached it from a binary classification perspective, considering each query-template pair as a unique data point to determine if it is a "match" or not. This method ensures that the model isn't constrained by the notion that a query can only correspond to a single template. Instead, it learns the pattern that a single query can have multiple matching templates, offering a more flexible and comprehensive recommendation system.

When addressing binary classification problems, our choices gravitate towards logistic regression, MLP, XGBoost, and other algorithms that offer good interpretability and speed. This decision is also grounded in our deployment strategy for the future: for every new query, due to our approach, there will be predictions equivalent to the number of templates. This makes the efficiency and interpretability of the model crucial, as it will be subjected to a vast number of predictions for each query in real-time.

Logistic Regression

Functional Form of the Model

Logistic Regression is a linear model for binary classification problems. The functional form involves predicting the log-odds of the probability of a particular class. The logistic (or sigmoid) function is used to transform linear combinations of predictors into values between 0 and 1, representing probabilities.

$$p = \frac{1}{1 + e^{-(a+bX)}}$$

Where p is the probability of the positive class.

e is the base of natural logarithms.

a is the bias or intercept term.

b is the coefficient of the predictor variable X.

Predicted Outcome

The model predicts the probability of an instance belonging to a particular class. A threshold (commonly set at 0.5) is then used to classify this probability into one of the two binary classes. However, in this case, the threshold will be adjusted to achieve a higher recall. In other words, by lowering the threshold, the model tends to predict more matches and therefore the recommendation system can recommend more than one candidate template.

Training Methodology

Training a logistic regression model involves estimating the parameters a and b that maximize the likelihood of observing the training data. Optimization techniques such as gradient descent are commonly used to find these parameters.

Model Validation Methodology

Data is typically split into training, validation, and testing. The training set is used to train the model, the validation set to tune hyperparameters and prevent overfitting, and the test set to evaluate the model's final performance. However, in this case, there are two types of test sets, one contains the queries that have a matched template used to test the model performance while another contains the queries that do not have a matched template. These queries do not have a matched template may be because there is indeed no matching template, or the user does not use the template function. Therefore, these queries could be a good simulation for unseen data.

Parameter Tuning and Methods

In the context of logistic regression, regularization techniques might be introduced to prevent overfitting. Two common regularization methods include L1 (Lasso) and L2 (Ridge) regularization. Hyperparameters related to regularization strength (C) are tuned using techniques like grid search and k-fold cross-validation on the validation set.

Evaluation Metrics

The primary evaluation metric for logistic regression in binary classification problems is accuracy. However, in imbalanced datasets, precision, recall, and the F1-score might be more indicative of the model's performance. In this case, attention should be paid to lowering the false negative as much as possible for two reasons. On one hand, it's more likely for a user not to be used to taking advantage of the template than not finding a matched template. On the other hand, the client expects to see the recommendation system recommending more than one candidate template. Therefore, recall would be a more suitable evaluation metric for the model.

Multi-layer Perceptron

Functional Form of the Model

Multilayer Perceptron (MLP) is a feedforward artificial neural network that consists of multiple layers of interconnected nodes or neurons. Each neuron in a layer is connected to every neuron in the subsequent layer. Each connection has an associated weight. An activation function, such as a non-linear function like ReLU, sigmoid, or tanh, is applied to the output of each neuron.

$$y = f(\Sigma(w * x + b))$$

Where:

y is the output after the activation function.

w is the weight of the connection.

x is the input from the previous layer.

b is the bias term.

f is the activation function.

Predicted Outcome

MLP can be used for both regression and classification tasks. In a classification problem, the final layer often uses a softmax activation function, which produces a probability distribution over the classes. The class with the highest probability is chosen as the prediction.

Training Methodology

MLP is trained using the backpropagation algorithm. During training, the model adjusts its weights and biases to minimize the difference between the predicted output and the actual target values, typically using a method such as gradient descent. The loss function, like cross-entropy for classification, quantifies this difference.

Model Validation Methodology

Same as the introduction above. Data is split into training, validation, and test sets. The training set is used to adjust the weights and biases of the model, the validation set helps in hyperparameter tuning and early stopping to prevent overfitting, and the test set provides an unbiased evaluation of the model's final performance.

Parameter Tuning and Methods

MLP has several hyperparameters that need to be tuned for optimal performance. These include the number of hidden layers, number of neurons in each layer, type of activation functions, learning rate, regularization parameters, and more. Techniques such as grid search, random search, and Bayesian optimization can be employed to find the best hyperparameters.

Evaluation Metrics

The primary evaluation metric for MLP in binary classification problems is accuracy. imbalanced datasets, metrics such as precision, recall, and the F1-score may offer a more accurate reflection of the model's performance. For our case, emphasis should be placed on minimizing false negatives due to two main considerations. Firstly, users are more prone to underutilizing available templates than encountering unmatched ones. Secondly, the client anticipates a recommendation system that suggests multiple candidate templates. Thus, recall emerges as a more appropriate metric for evaluating the model.

XGBoost

Functional Form of the Model

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting algorithms. The functional form of XGBoost can be described in terms of how it builds the model by sequentially adding decision trees, where each new tree attempts to correct the errors made by the previous ensemble of trees.

The objective function in XGBoost is a combination of a loss function and a regularization term. It is given by:

$$Obj(\theta) = L(\theta) + \Omega(\theta)\ \hat{y}$$

Where L is the loss function that measures how predictive the model is with respect to the training data and Ω is the regularization term that penalizes model complexity to prevent overfitting. θ represents the parameter of the model.

For binary classification problems, XGBoost typically uses the binary logistic loss, also known as log loss or binary cross-entropy. The formula for binary logistic loss is:

$$L(y, \hat{y}) = -\sum [y_i(1 + exp(-\hat{y}_i)) + (1 - y_i)\ log(1 + exp(\hat{y}_i))]$$

Here y_i is the actual label (0 or 1), and $\hat{y}_i$ is the predicted probability for the ith instance.

The regularization term helps to smooth the final learned weights to avoid overfitting. It is typically defined as

$$\Omega(\theta) = \gamma T + \frac{1}{2}\lambda \sum w_j^2$$

Where T is the number of leaves in the tree, w_j are the scores on the leaves, γ is the complexity control for the number of leaves, and λ is the L2 regularization term on the leaf weights.

Predicted Outcome

XGBoost can perform both classification and regression tasks. In binary classification, where the target variable has two classes, often labeled as 0 and 1, XGBoost models predict a probability score between 0 and 1. This score represents the likelihood of the instance belonging to the positive class, usually denoted as 1, in our case, the score measures the likelihood of a case that can be matched with a certain template. We apply a threshold of 0.5 to these probabilities, which means if the predicted probability is greater than or equal to 0.5, the instance is classified as the positive class, which is matched; otherwise, it's classified as the negative class, which is unmatched.

Training Methodology

The training methodology of XGBoost for classification tasks involves building an ensemble of decision trees sequentially. Each tree in the sequence focuses on correcting the errors made by the previous ones, which is log loss for binary classification. Regularization techniques are integrated to prevent overfitting, and the model is carefully fine-tuned through various hyperparameters. Additionally, advanced optimizations such as tree pruning and handling imbalanced data enhance the model's accuracy and efficiency.

Model Validation Methodology

Same as the introduction above. Data is split into training, validation, and test sets. The training set is used to build the ensemble of decision trees, with each tree learning to correct the mistakes of its predecessors. The validation set plays a crucial role in hyperparameter tuning, allowing for the adjustment of XGBoost's numerous parameters to optimize model performance. The test set serves to impartially assess the final performance of the model, confirming its efficacy and ability to generalize to new and unobserved data.

Parameter Tuning and Methods

XGBoost provides an extensive array of tunable parameters, encompassing the learning rate and various tree-specific parameters such as the maximum depth of the trees and the minimum loss reduction required for further partitioning on a leaf node. Additionally, it includes several regularization parameters, especially the L1 and L2 regularization terms applied to the weights. Methods including grid search, random search, and Bayesian optimization are utilized for hyperparameter tuning.

Evaluation Metrics

Same as the introduction above, the primary evaluation metric for XGBoost in binary classification problems is accuracy. Nonetheless, in the context of imbalanced datasets, metrics such as precision, recall, and the F1 score provide a more accurate reflection of the model's performance. In our specific scenario, it is imperative to minimize the occurrence of false negatives for two primary reasons. Firstly, it is more probable that a user may overlook utilizing an available template rather than encountering a scenario where no suitable template is found. Secondly, the client anticipates that the recommendation system should suggest multiple candidate templates. Consequently, in this context, recall is more important than precision. But

we will keep precision larger than a certain threshold so that the recommendation system won't recommend too many irrelevant templates.

General Model

Considering our client's requirement for a universal model applicable across all districts and our computational resources, we selected the top 10% most frequently used templates from each district. This approach yielded a total of 266 templates and their corresponding 35,744 cases. These were utilized to train our XGBoost model, which has demonstrated the best performance in our evaluations. This strategy aims to develop a generalized model for district-wide application, thereby reducing the need for multiple models across different districts and enhancing overall efficiency.

Findings

We use the train-validation split to validate our model, which means we divide our queries with matched template data into three subsets: a training set, a validation set, and a testing set. The model is trained on the training set, tuned on the validation set, and evaluated on the testing set. There is another test set which includes the cases with unmatched templates, and this is the unseen data to test our recommendation engine.

Regardless of the selection of models, all models will be tuned to solve a binary classification problem since the task of template matching essentially tells whether the semantic meaning of inquiries matches that of templates. Therefore, the following regular classification metrics will be used for the whole queries with matched template data:

1. Accuracy: It is the ratio of correctly predicted observations to the total observations. In the template matching task, it is the ratio of the number of correct matches to the total number of matches. It measures how well the system performs in terms of correctly

matching the templates to the inquiries. A higher accuracy value indicates a higher proportion of correct matches.

2. Precision: It is the ratio of correctly predicted positive observations to the total predicted positives. In the template matching task, it is the ratio of the number of relevant templates retrieved to the total number of retrieved templates. It focuses on the quality of the retrieved templates. It measures how many of the retrieved templates are actually relevant. A higher precision value is considered better.
3. Recall: It is the ratio of correctly predicted positive observations to the total actual positives. In the template matching task, it is the ratio of the number of relevant templates retrieved to the total number of relevant templates. It is a measure of the completeness of the retrieved answers. It quantifies the proportion of relevant templates that are successfully retrieved. A higher recall value is considered better.
4. F1-score: It is the weighted average of Precision and Recall, which tries to find the balance between precision and recall. A higher F1 score is considered better.
5. Precision and Recall Curve: It is used to evaluate the performance of binary classification models, particularly when the dataset is imbalanced. It illustrates the trade-off between precision and recall as the classification threshold varies. The curve provides insights into how well the model balances the need to correctly identify positive instances while minimizing false positives. A higher area under the Precision and Recall Curve indicates better performance.

Among the above five classification metrics, recall and precision are the two most important metrics and recall holds a higher priority than precision. This preference aligns with our primary objective, which is to accelerate user responses to customer queries and elevate the

overall user experience. By minimizing the requirement for users to repeatedly create responses, we aim to deliver a comprehensive collection of pertinent templates. In this specific context, recall's importance takes precedence over precision. We will maintain precision at a level that ensures all recommended templates are pertinent to the queries while avoiding unrelated suggestions.

Table 1. *Classification Report of District 44070 using Logistic Regression*

	Precision	Recall	F-1 Score	Support
0	1.00	0.96	0.98	100548
1	0.17	0.90	0.29	882
Accuracy			0.96	101430
Macro Avg	0.58	0.93	0.63	101430
Weighted Avg	0.99	0.96	0.97	101430

Table 2. *Classification Report of District 44070 using MLP*

	Precision	Recall	F-1 Score	Support
0	1.00	1.00	1.00	100548
1	0.75	0.44	0.56	882
Accuracy			0.99	101430
Macro Avg	0.87	0.72	0.78	101430
Weighted Avg	0.99	0.99	0.99	101430

Table 3. *Classification Report of District 44070 using XGBoost*

	Precision	Recall	F-1 Score	Support
0	1.00	0.97	0.98	100548
1	0.20	0.92	0.33	882
Accuracy			0.97	101430
Macro Avg	0.60	0.94	0.66	101430
Weighted Avg	0.99	0.97	0.98	101430

The three tables above show the classification report for the test data of District 44070, which is the district that has the second-largest number of templates. The optimal result is achieved by employing XGBoost, as it yields the highest recall score and a marginally superior precision compared to logistic regression, whereas MLP's significantly lower recall score suggests it may not recommend an adequate number of relevant templates. It is clear to see that from Table 3, which is the classification report using XGBoost, the precision reaches 0.94 for this district and the accuracy reaches 0.97. Also, we have evaluated our model across various districts, and the recall score consistently reaches 0.9, which indicates that this recommendation engine successfully identifies and recommends 90% of relevant templates within each district. This shows its effectiveness in providing users with options closely aligned with their needs or queries.

We also applied these metrics to another test dataset, consisting of queries with unmatched templates. In this scenario, as we are uncertain about the presence of matches, we manually labeled 20 test cases to assess the accuracy of these matches. The result of the test case pass rate is in the below table:

Table 4. *Test Case Pass Rate for Different Models*

Embeddings	Model	Test Case Pass Rate
Fine - Tuned	XGBoost	80%
Raw	XGBoost	45%
Fine - Tuned	Logistic Regression	30%
Raw	Logistic Regression	55%
Fine - Tuned	MLP	30%
Raw	MLP	35%

From the above table, it is obvious to see that using the XGBoost model with embeddings extracted from the fine-tuned sentence transformer has the best test case pass rate.

We also test our model on district 45671 and the below table shows all the recall scores using different models:

Table 5. *Recall Rate for Different Models for District 45671*

Recall Rate		
	No Fine-tuning	Fine-tuning
XGBoost	0.71	0.80
Logistic Regression (LR)	0.64	0.69
Neural Network (NN)	0.64	0.60

The result in the above Table 5 further illustrates the point that XGBoost with embeddings from fine-tined sentence transformers has the best result, which further shows the

effectiveness of fine-tuning the sentence transformer model since there is obvious improvement in model performance after using embeddings from fine-tuned sentence transformer.

Our client is also prepared to conduct A/B tests to evaluate the performance of our recommendation engine, and these metrics can serve as key evaluation criteria for the A/B tests.

Discussion

The main objective of this template recommendation engine is to increase the usage of the Let's Talk template feature to facilitate the usage of existing templates to accelerate user responses to customer queries. Our template recommendation engine is able to recommend the most relevant templates with a recall score of 0.9, which means that it can successfully identify and recommend 90% of relevant templates within each district. This number shows that it definitely helps to increase the efficiency of entering user responses.

The final model we used is XGBoost with embeddings using fine-tined sentence transformers. The reason why XGBoost worked better than Logistic Regression is that Logistic Regression is a simpler and linear model and may not capture the complex patterns in data as effectively as XGBoost. Compared with Neural Networks, XGBoost has fewer hyperparameters to tune which leads to better results since it simplifies the model selection process, making it more accessible and less prone to overfitting. Furthermore, XGBoost incorporates built-in regularization, which can contribute to better generalization of unseen data.

Our recommendation engine has several limitations. First, given that each district possesses its own distinct set of templates, and users are restricted to choosing templates within their designated district, the model may require individualized adjustments for each district. This limitation may lead to differences in the relevance and quality of template recommendations for users in different districts. If new cases and templates from additional districts are introduced in

the future, it may be necessary to retrain the model using the updated data or the users may miss out on the opportunity of using templates in new districts. The second constraint is that our recommendation engine exclusively functions with English text, making it unable to provide template recommendations for case descriptions written in languages other than English. This limitation restricts the engine's reach and hinders its effectiveness for a global or multilingual user base. In addition, our recommendation engine is incompatible with input that combines different data types, for instance, queries that include images. This limitation narrows the scope of cases the engine can effectively handle, potentially limiting its utility for users with image-centric queries.

Conclusion

Overall, the accomplishment that we hope to achieve at the end of the capstone project is to deliver a comprehensive and efficient template recommendation engine that can assist school administrators in choosing response templates more quickly and accurately.

We are expected to develop a template recommendation engine, which takes text, an incoming message from customers, as the model input. The model output will be one or several pre-existing Let's Talk templates that are highly relevant to the customer inquiry if they exist. If there are no relevant templates, we will not make any recommendations.

Our deliverables consisted of three parts, the first component includes the Python code used to train those models, the second part is the streamlit web app built on our best model to test and show the performance more clearly, and the third part is the written technical documentation of the techniques we used along with the evaluation of the model based on our evaluation metrics.

Streamlit Web App

In our project, we have developed our query-template matching model into a Streamlit web application. This transformation serves dual purposes: enhancing the visualization of test results for clearer analysis, and providing an interactive platform for our clients to conduct further A/B testing. The latter is especially crucial for validating the real-world impact of our recommendation system.

Introduction to Streamlit

Streamlit is an open-source Python library that simplifies the process of creating and sharing beautiful, custom web applications for machine learning and data science projects. It enables developers to quickly turn data scripts into shareable web apps with minimal coding effort. Streamlit's intuitive design and powerful functionalities make it an ideal choice for showcasing machine learning models and engaging users in interactive data exploration.

Implementation and Key Features of the Application

The Streamlit application includes several features to ensure a smooth user experience. These key features are shown as follows:

1. Web App Introduction: The application begins with a brief introduction, explaining its purpose and how users can interact with it. This introduction sets the stage for the user experience, providing context about the underlying query-template matching system.
2. Regional Selection Dropdown: We incorporated a dropdown box enabling users to select different regions. This feature is crucial as our pre-created templates are region-specific, allowing for a tailored experience based on the user's geographical preference.
3. User Query Input: A dedicated input box is provided for users to enter their queries. This user-friendly feature is designed for ease of use, encouraging users to actively engage with the system.

4. Processing and Model Interaction: Upon submitting a query, the application takes in and preprocesses the input information. It then loads the model and utilizes it to find the most suitable answer template for the given query and selected region.
5. Display of Results: The application dynamically displays the identified answer template on the web app, providing users with immediate feedback on how the model responds to their specific queries.

Purpose and Benefits

The development of the Streamlit web application is a significant deliverable of our project. It not only showcases our model's capabilities in an engaging and comprehensible manner but also facilitates practical evaluation and testing by our clients. This application aids in demonstrating the effectiveness and adaptability of our query-template matching system in real-world scenarios, providing a clear avenue for assessment and feedback.

Our solution delivers more than just efficiency. First of all, our recommendation engine delivers 90% relevant pre-written response templates for 70% of all school districts, which reduces the response time by 60%. Secondly, it significantly reduces the annual operational costs by 0.7 million dollars due to the increase in response efficiency. Thirdly, since achieving enhanced efficiency in addressing inquiries, we have attracted more school districts and are targeting a 60% expansion in our customer base.

Next Steps

Our next step involves consistently fine-tuning and enhancing our recommendation engines. This includes exploring various models to address language limitations, optimizing the algorithm to improve training speed without compromising recall and precision, and adding new scoring components. It is also important to collect feedback from users when utilizing this

template recommendation engine, including explicit ratings, comments, as well as implicit feedback drawn from user interactions and behavior within the system.

For instance, A/B tests can be used to evaluate the effectiveness of this recommendation engine. This involves conducting experiments with users who either have the recommendation engine implemented or do not have it implemented when providing their responses. The goal is to observe changes in metrics like user engagement, response time, or click-through rates.

Reinforcement Learning can also be employed to gather implicit feedback from user interactions within a system by defining the environment, agent, states, actions, and rewards. The agent observes user behaviors, such as clicks and navigation patterns, and receives rewards based on positive or negative user responses to its actions. Over time, the agent learns to take actions that enhance user engagement or satisfaction by continuously updating its policy to maximize cumulative rewards. This approach requires ongoing training and adjustment of the model to ensure it remains effective and aligns with the desired outcomes, such as improved user experience or increased conversions.

Expanding beyond our current work, it is valuable to consider integrating our recommendation engine into a variety of platforms, including mobile applications and streamlit web applications. Such integration serves to enhance the overall user experience. For instance, leveraging mobile device features such as location data, our engine can provide users with templates that are not only context-aware but also location-aware so that users are not required to manually input their district information.

Furthermore, the adaptability of our recommendation engine allows for its customization in e-commerce industries, where a substantial volume of customer service cases is a constant challenge. While our recommendation engine's primary function is to match queries and

templates for online education platforms, it can be efficiently adapted to match cases and templates used by e-commerce websites.

In addition to the above, it is also important to develop models to create templates for frequently used answers that haven't yet been templated. For instance, Retrieval-Augmented Generation model can be a suitable approach. The retrieval part of RAG can be used to search through the database of existing answers. This component helps the model find the most relevant existing answers or information related to a given query or topic. Once relevant answers are retrieved, the generative part of the model can be employed to synthesize and generalize this information into a template format.

References

Chicco, D. (2021). Siamese neural networks: An overview. *Artificial neural networks*, 73–94.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

Dhakal, A., Poudel, A., Pandey, S., Gaire, S., & Baral, H. P. (2018). Exploring deep learning in semantic question matching. In *2018 ieee 3rd international conference on computing, communication and security (icccs)* (pp. 86–91).

Hiew, J. Z. G., Huang, X., Mou, H., Li, D., Wu, Q., & Xu, Y. (2019). Bert-based financial sentiment index and lstm-based stock return predictability. *arXiv preprint arXiv:1906.09024.*

Prager, J., Brown, E., Coden, A., & Radev, D. (2000). Question-answering by predictive annotation. *In Proceedings of the 23rd annual international acm sigir conference on research and development in information retrieval* (pp. 184–191).

Simmons, R. F. (1965). Answering english questions by computer: a survey. *Communications of the ACM*, *8*(1), 53–70.

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586.*

Tan, M., Dos Santos, C., Xiang, B., & Zhou, B. (2016). Improved representation learning for 24 question answer matching. *In Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 464–473).

Tan, M., Santos, C. d., Xiang, B., & Zhou, B. (2015). Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30.*

Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., & Le, Q. V. (2018). Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541.*

Appendix

Sentence Level Similarity Calculation

Since there may be some irrelevant information or words in both queries and templates, sentence level similarity calculation is applied to calculate the similarity between queries and templates to extract the most relevant pair of sentences. In sentence-level similarity calculation, both queries and templates will be divided into different sentences. Embeddings of these sentences will be derived from the fine-tuned Sentence Transformer model. The similarities for each pair of sentences between queries and templates will be calculated by the dot product of respective embeddings. For example, if there are m sentences in the query and n sentences in the templates, m*n similarities will be calculated, and the highest will be used to represent the similarity between the query and the template.

Threshold-Based Recommendations

Threshold-based recommendations have been tried to recommend templates, but the performance is quite unsatisfactory. Therefore, it will not be applied in our model pipeline. In threshold recommendation, the thresholds of similarities for each district will be determined. If the similarity between the query and the template is higher than the threshold, this template will be recommended. The threshold is calculated by the following steps:

1. The sentence level similarities of all matched queries and templates are calculated.
2. Q1(25%quantile), Q3(75%quantile) and IQR(Q3-Q1) of the list of similarities are derived. The lower bound = Q1- 1.5*IQR is then determined to remove the outliers.
3. The threshold will be the higher one of the lower bound and the minimum value of all similarities.