TDI Capstone Project Description

May 28, 2021

Sentiment Analysis of Text Messages

Imagine a platform for finding the perfect match concerning both a room and the roommates. Find a good place to live in and build strong communities. Messaging features facilitate the interaction between residents and the applicant to get to know each other.

The main focus of this project is to explore whether, and if yes, how messaging feature helps to shape the applicants and the residents' experience and their decision; for applicants deciding to sign a lease and for residents to approve or refuse an applicant.

To navigate this question, I use Natural Language Processing technique and perform sentiment analysis of the chat messages. My results allow flagging of signaling messages that the customer service can further process and attend to the clients' concerns more effectively.

Below I explain the steps I took into navigating the project.

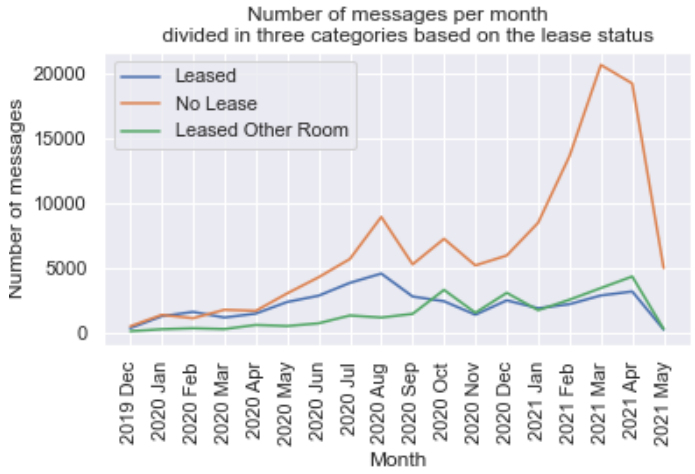
Exploratory Analysis

The data includes 180K observation and the following information:

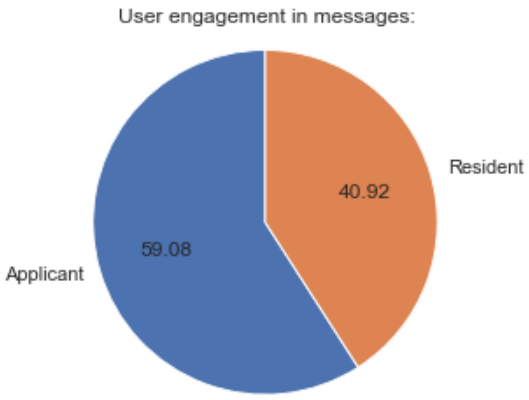
- **Channel ID**, assigned to each chat message initiated corresponding to a listing.
- **Creation timestamp** corresponding to each sent message.
- **Users ID** (the applicants and residents) sending a message.
- The role of each **user** (applicant or resident).
- The **lease status**, corresponding to the message channel initiated for each listing.
- The **market** as in the city where the listing is located.
- And finally, the **messages**.

Through initial exploration of the dataset, the following observations are made:

- Overall, the number of messages exchanged every month is drastically increasing. Meaning more clients are using the application, looking up rooms, and interact via the messaging feature. However, the number of successful leases shows a minor increase, which could mean that despite more significant engagement and usage, the number of successful leases is not growing at the same speed.



- More applicants engage and send messages compared with the residents. This could be particularly interesting considering that there is usually more than one resident in each household. This could indicate that not all the residents within a household are engaging in the chat messages, while they all should vote on accepting or refusing an applicant. In other words, it could be an indication of lower engagement of residents in the process.



These two main observations point us in exploring the content of the chat messages and the behavior of applicants and the residents.

The illustration are produced via exploratorty and explanatory notebook.

Preprocessing and Feature Engineering

The data set is preprocessed to prepare the data for sentiment analysis and set up features used for the predictor model. In particular:

- The response time for each message is quantified using the timestamps provided for each message.

- The text of messages is cleaned up to remove HTTP addresses and unnecessary notations.
- The length of conversations is quantified as the number of messages sent within each channel.
- Messages are numbered based on their order in the conversation.

See preprocessing notebook for details.

Sentiment Analysis

Using natural language processing techniques, the sentiment around each text message is evaluated.

In particular, Stanford CoreNLP API sentiment tool is used to extract the sentiment score of each sentence in a message. The Sentiment Annotator of CoreNLP implements Socher et al.'s sentiment model by attaching a binary tree in the sentence level. The Node of the tree contains the predicted class and scores for that subtree. The current version of the sentiment annotator of CoreNLP includes five score classes: very negative, negative, neutral, positive, and very positive.

I extract the probability distribution associated with the five score classes (very negative to very positive) for each sentence within a message to record the sentiment scores of text messages. A score of -2, -1, 0, 1, 2 is assigned to the five classes, and with that, the expected score for each sentence is calculated as:

$$E_i = \sum_{j=1}^{5} P_{i,j}(s) \times s_{i,j}$$

where E_i is expected score corresponding to sentence i, and P_j is the probability associated with each score class, s_j in (-2,-1,0,1,2).

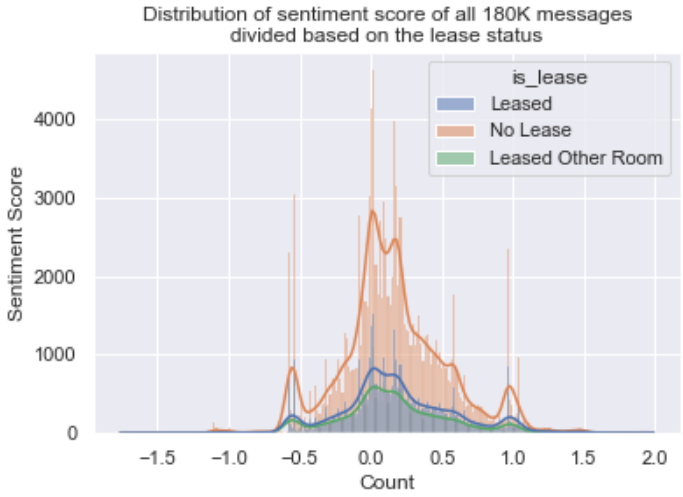
Ultimately, the frequency of score classes for a text message is calculated by taking the histogram of expected scores of all sentences. Furthermore, The average of expected scores of sentences is calculated to indicate the overall score of a whole text message.

The number of sentences and words for each message is recorded through this process.

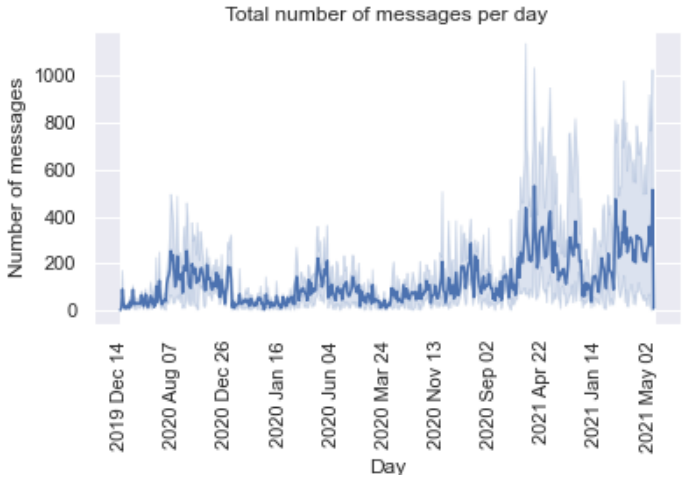
See the sentiment analysis notebook for further information.

Implications

The distribution of sentiment score of all messages shows a multimodal distribution (figure below), with a pretty dominant peak of neutral messages as expected. Most messages intuitively include unbiased content introducing oneself, exchanging contacts to meet in person, etc. There are two smaller peaks observed at negative and positive scores. These messages could include more signaling content. I took a closer look into the messages that contain a higher ratio of negatively scored sentences. Interestingly, most of these messages had more conflicting content, and a pipeline to identify such messages can be an excellent tool for the company to improve the clients' experience.



Filtering the thousands of messages sent daily (figure below) and flagging only the highly signaling messages enables the customer service unit to (better) identify clients' concerns and therefore be able to address them more effectively.

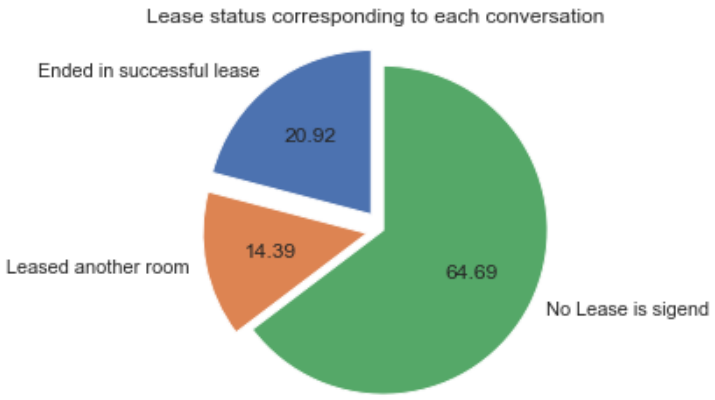


Supervised Machine Learning Model

Features, observations, and labels. Following features are provided in the data set: - User role (applicant or resident), - Response time, - Conversation length, - Message length, - Average sentiment score for a text, - Frequency of sentiment score of all sentences within each message in five classes: very negative, negative, neutral, positive, very positive.

The data set includes a total of 180K observations and 10 features. A binary classification is performed to predict the lease status. The 35.31% of conversations correspond to a sucessful lease (same room or another) versus not leasing a room at all (see figure below), and therefore the classes are assumed to be balanced.

The data set is divided into training and testing (90%:10%) sets, manually, in order to assure that



messages corresponding to the same channel are kept together within one data set.

Preprocessing. The categorical feature, i.e., the user status (applicant or residents), is encoded via One Hot Encoder algorithm.

Training a classifier. I trained Scikit Learn Random Forest Classifier to predict the lease status corresponding to each channel.

Since I had a vague idea of the best hyperparameters and I decided to narrow my search and first evaluate a wide range of values for each hyperparameter using Randomized Search Cross Validation.

Based on the best parameters identified by the random search, I concentrated my search on a smaller range of hyperparameters and used Grid Search explicitly try every specified combination.Both of these algorithms are provided in the python Scikit Learn library.

> See ML model notebook for details.

Implications

The performance of the ML model is limited by features that are used in training. Other than the chat impression, there are many other components, which can affect (1) the resident's vote on accepting or rejecting an applicant and (2) the applicant's decision on signing a lease for a room.

In the future outlook, other features such as: (1) the number of maintenance tickets submitted via each household, (2) unit price compared to market prices, and (3) the number of available rooms versus total rooms, could be included in order to improve the accuracy of the predictor models.