National Taiwan University

Financial Technology

Project 3: Push Notification Click Prediction

Lecturer:
Che Lin (林澤)

Author: Tadeo Hepperle, Student ID: A11922105

November 22, 2022

Contents

1 Introduction

In this project we will attempt to use bidirectional LSTMs, GRUs and a Transformer Encoder model to predict if a push notification will be clicked. As input, we give to the model information about the past 10 push notifications that have been sent out to the user that receives the current push notification, including the day of the week and the topic of each notification. The models are trained on the binary response whether a push notification is clicked or not and are expected to give good binary predictions on unseen data.

1.1 Description of the Data

The data for the project was provided by the "AviviD Innovative Media" company, that excels in digital marketing. There are 150023 push notifications in the data, that were sent to 1427 different users. This amounts to about 100 push notifications per user that have been sent over a period of 10 days, so roughly 10 push notifications per day. Each push notification represents a news from one of 19 categories, that include a wide range of topics. Examples for such topics are: "Technology", "Games", "Politics" and "China". All of these topics are one-hot-encoded. The respective weekday has also been one-hot-encoded, because it might be an important predictor. Together, weekday and topic form a 27-element binary vector for each push notification. The data has been aggregated to include a 11x27 matrix for each notification, representing the features of this push notification and the 10 that have been most recently sent to the same user. This allows us to utilize sequential models such as LSTM and GRU for the predictions. For push notifications early on in a user's history, where less than 10 previous notifications are available, the first few rows of the 11x27 matrix have been padded with zero-values.

1.1.1 Splitting the Data

The data was split into 3 parts along the time axis:

- Train set: 2022/08/01-2022/08/07, 105320 items
- Validation set: 2022/08/08-2022/08/09, 34362 items
- Test set: 2022/08/10, 10341 items

The data is fairly imbalanced, with only 1519 of the 150023 items being clicks (label=1). This is just 1.01% and could lead models to ignore the minority class. The 148504 remaining cases were all either ignored by the user or blocked.

1.2 Description of Methods

To predict whether a push notification is clicked we use bi-directional LSTMs, GRUs and a Transformer Encoder Layer. All of these models follow the same principle: encode a k-dimensional input vector over t time steps into an e-dimensional encoding. A linear combination of all e elements of the e-dimensional encoding vector is then produced by a linear fully connected layer to produce a single scalar output. This scalar is then transformed by a sigmoid function to produce values between 0 and 1, which can be interpreted as predicted click-probabilities.

1.2.1 Loss Function and Optimizer

The binary cross entropy loss (BCE) has been used as a criterion for the gradient descent in training:

$$BCE = -(y \log 1 - p + (1 - y) \log 1 - p)$$

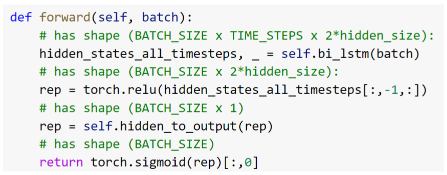
To optimize the models, the Adam optimizer with a learning rate of 0.001 has been utilized throughout the report. Other learning rates were tried but were either too slow to converge or were too high and therefore too "jumpy" and could not get the error low enough consistently. A batch size of 256 was used in training.

1.2.2 Bi-Directional LSTM

A unidirectional LSTM, as described in the last report, just looks at past data and updates the hidden state and cell state in each time step. In comparison, a bidirectional LSTM just consists of two unidirectional LSTMs, where one iterates over the time steps in a forward manner like the normal LSTM from 0 to t and the other one learns by looking at the date from the back to the front, from time step t to 0. Each of them has a $hidden\_size$-dimensional cell state and hidden state respectively. The final encoding of the input sequence after consuming all time steps in their respective directions is then just a concatenation of the final hidden states of forward and backward LSTM. This is a vector of size $2 * hidden\_size$. Even though in principle a BiLSTM with a hidden size of $hidden\_size$ is not more powerful than a single LSTM with a hidden size of $2 * hidden\_size$, in practice BiLSTM seem to learn a good representation of the data faster.

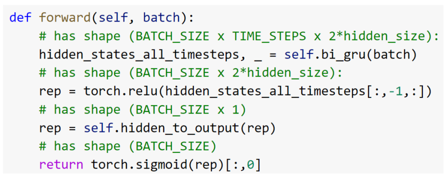
1.2.3 Bi-Directional GRU

A bidirectional general recurrent unit (BiGRU) relates to a unidirectional GRU in the same way a BiLSTM relates to an LSTM. only the implementation details in the two submodels (forward and backward) differ: the two GRUs used do not have a cell state like an LSTM cell and are less complicated to compute. How outputs are combined and predictions are made does not differ though. The BiGRU is like the BiLSTM parameterized by $hidden\_size$ For this paper we only trained single layer BiGRUs and BiLSTMs. Implementation of the forward pass of our BiLSTM and BiGRU in pytorch is displayed in Figure 1



```python
def forward(self, batch):
    # has shape (BATCH_SIZE x TIME_STEPS x 2*hidden_size):
    hidden_states_all_timesteps, _ = self.bi_lstm(batch)
    # has shape (BATCH_SIZE x 2*hidden_size):
    rep = torch.relu(hidden_states_all_timesteps[:,-1,:])
    # has shape (BATCH_SIZE x 1)
    rep = self.hidden_to_output(rep)
    # has shape (BATCH_SIZE)
    return torch.sigmoid(rep)[:,0]
```

(a) Forward Pass BiLSTM



```python
def forward(self, batch):
    # has shape (BATCH_SIZE x TIME_STEPS x 2*hidden_size):
    hidden_states_all_timesteps, _ = self.bi_gru(batch)
    # has shape (BATCH_SIZE x 2*hidden_size):
    rep = torch.relu(hidden_states_all_timesteps[:,-1,:])
    # has shape (BATCH_SIZE x 1)
    rep = self.hidden_to_output(rep)
    # has shape (BATCH_SIZE)
    return torch.sigmoid(rep)[:,0]
```

(b) Forward Pass BiGRU

Figure 1: Forward pass of BiLSTM and BiGRU are almost identical

1.2.4 Transformer Encoder Layer

A Transformer Encoder Layer (TEL) is one part of the popular Transformer-Architecture. It consists of a self-attention mechanism and a feed-forward layer. The output of the feed-forward layer can be mapped back to the input dimension to create an encoding that is hopefully better suited to be used in a linear prediction of a scalar value as the output. There are a number of attention heads that learn independently which parts of the input sequence are most important. In our implementation a feed-forward dimension of 64 has been chosen, we use 3 attention heads.

2 Results

We discovered that the loss of all models starts to show signs of overfitting after the first 3-10 epochs. This effect is shown in Figure 2 where we trained a BiGRU with a hidden size of 64 for 100 epochs.

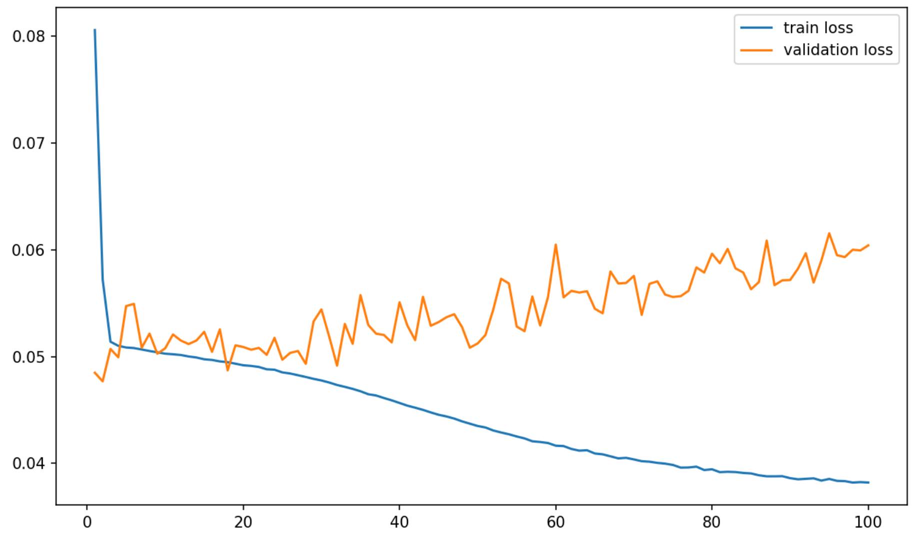
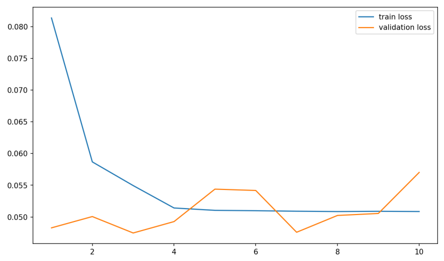
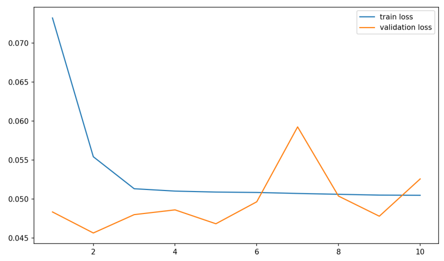


Figure 2: Overfitting behavior of a BiGRU with $hidden\_size = 64$

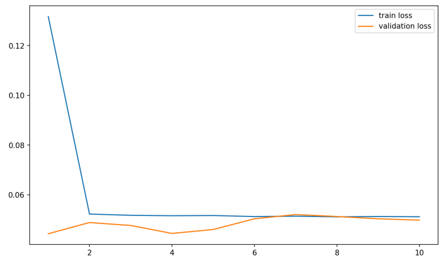
Therefore, we choose to train our models for 10 epochs only and all statistics below are subject to this value. By experimentation a hidden size of 64 was chosen for the BiLSTM and BiGRU models. Figure 3 shows training and validation losses after training the BiLSTM, BiGRU and TEL models for 10 epochs. As we can see all models reach a test and train error of around 0.05. Table 1 displays the accuracy of the 3 models on train, validation and test set each.



(a) Train and Validation Loss BiLSTM



(b) Train and Validation Loss GRU



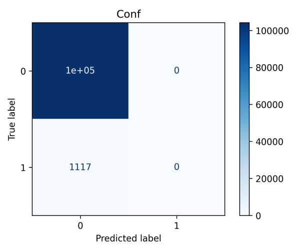
(c) Train and Validation Loss Transformer Encoder Layer

Figure 3: Train and Validation Losses for all 3 models

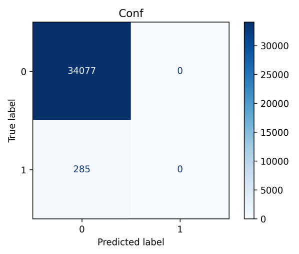
Table 1: Accuracy of the 3 models

Accuracy	Train	Validation	Test
BiLSTM	0.989	0.992	0.989
BiGRU	0.00103	0.992	0.989
TEL	0.989	0.992	0.989

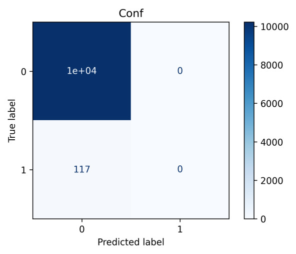
At a first glance the values in Table 1 seem very good. Such a high accuracy! But taking a closer look at the confusion matrices for all models, see Fig 4 reveals that all models just predict "no click" for every single instance in every single set. This is likely an artifact of the imbalance of the data and should probably be dealt with by choosing a different loss function or an under/oversampling approach.



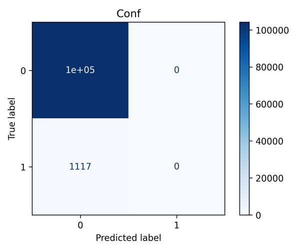
(a) BiLSTM on train



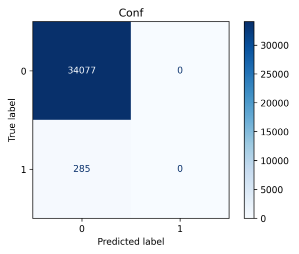
(b) BiLSTM on validation



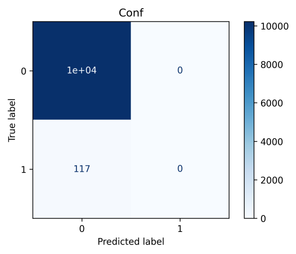
(c) BiLSTM on test



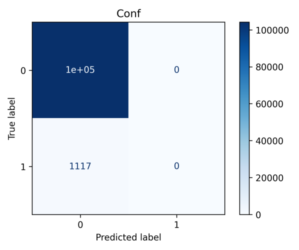
(d) BiGRU on train



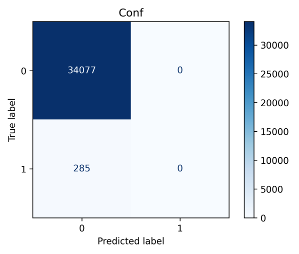
(e) BiGRU on validation



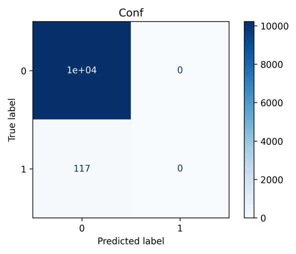
(f) BiGRU on test



(g) TEL on train



(h) TEL on validation



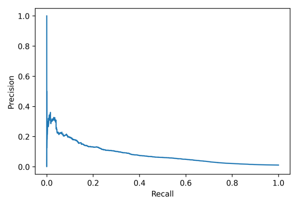
(i) TEL on test

Figure 4: Confusion Matrices

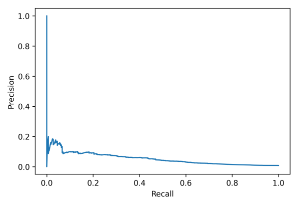
2.1 ROC and PRC

The ROC curve plots the true-positive rate against the false-positive rate, while the PRC plots precision against recall. In our case we should choose PRC because our data is very imbalanced. The ROC gives a false sense of quality because so many of the non-clicks receive a low probability. ROC curves and PRC curves are shown in Figure 6 and Figure 5 respectively.

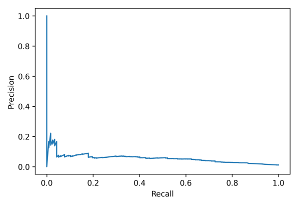
Table 2 shows the area under curve (AUC) values for PRC and ROC on train, test and validation sets for the 3 models. Comparing the 3 models we find that they are all very similar. However, the BiGRU shows the highest values in the PRC on the test set if that is what we care about. All in



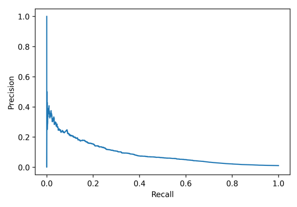
(a) BiLSTM on train



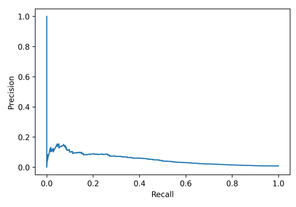
(b) BiLSTM on validation



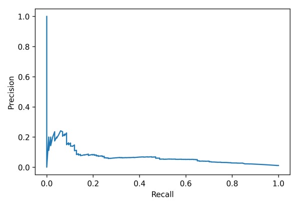
(c) BiLSTM on test



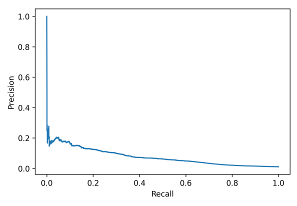
(d) BiGRU on train



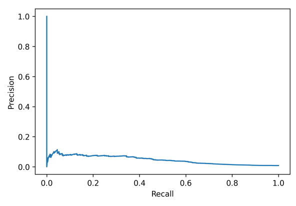
(e) BiGRU on validation



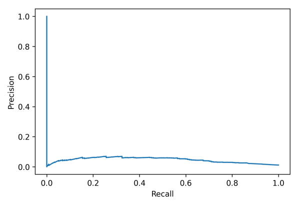
(f) BiGRU on test



(g) TEL on train



(h) TEL on validation



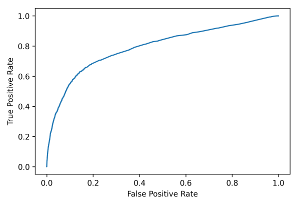
(i) TEL on test

Figure 5: Precision recall curves

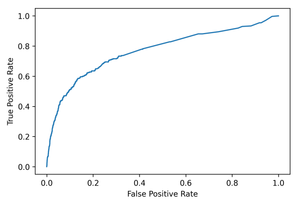
all the values are so low though, that these small differences might just be statistical anomalies and could be different in another run of training. We would have thought that the transformer performed better, but it was actually the worst on all datasets when using PRC as the metric.

2.2 Using a different time window

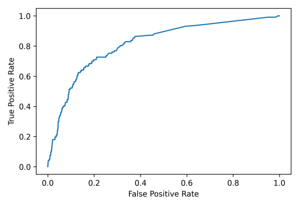
In the following we retrain all models with the same configurations but this time we just allow them to learn on a) the past 3 days or b) on the past 5 days in comparison to the 10 days that were used before. As a metric of performance we compare area under PRC and ROC curve on the test set in Table 3. The accuracy and confusion matrices have not changed at all, still all models predict all instances to be non-click. More about how this can be addressed in the next section. From the values in the PRC AUC values



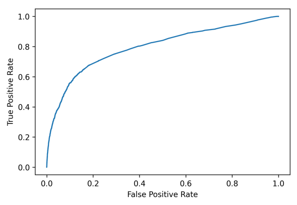
(a) BiLSTM on train



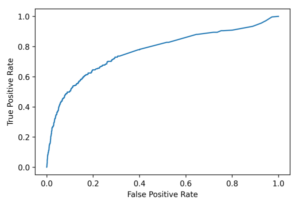
(b) BiLSTM on validation



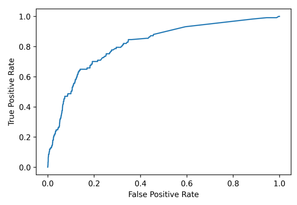
(c) BiLSTM on test



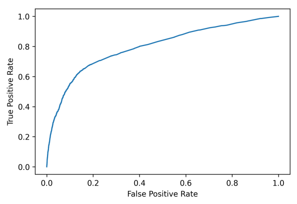
(d) BiGRU on train



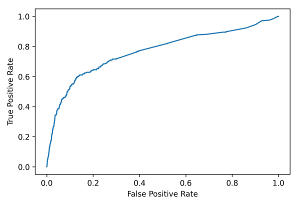
(e) BiGRU on validation



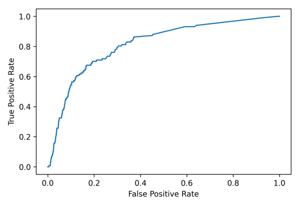
(f) BiGRU on test



(g) TEL on train



(h) TEL on validation



(i) TEL on test

Figure 6: receiver operating characteristic curves

in the table it seems like Transformer Encoder Layer and BiGRU have a slightly better test performance when only 3 days are used. For BiLSTM no real difference was visible.

2.3 Dealing with the Data Imbalance

Until now, we just predicted instances where a model gives a probability of ≤ 0.5 as "non-click" and otherwise as "click". However, this might not be the best cutoff value. If we look at the ROC curve, we can choose the point on the curve closest to the top-left corner as the best point. We want to achieve a recall (= true positive rate) like in this point. To do so, we can take all our predicted probabilities for instances that are clicks in reality and sort them according to their predicted probability in decreasing order. Let L be the length of that sequence. The element at index $L * recall$ is

Table 2: PRC and ROC AUC values of the 3 models

	ROC		
	Train	Validation	Test
BiLSTM	0.7927	0.7673	0.8143
BiGRU	0.7962	0.7687	0.8161
TEL	0.7963	0.7658	0.8135
	PRC		
	Train	Validation	Test
BiLSTM	0.0832	0.0529	0.0559
BiGRU	0.0902	0.0518	0.0675
TEL	0.0745	0.0457	0.0455

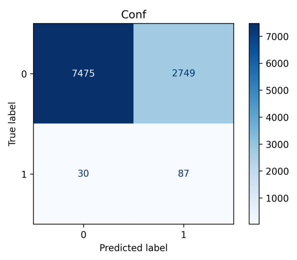
Table 3: Models trained on different time frames

	PRC AUC	ROC AUC
3 days BiLSTM	0.052	0.8047
5 days BiLSTM	0.0444	0.8098
10 days BiLSTM	0.0559	0.8143
3 days BiGRU	0.0761	0.8058
5 days BiGRU	0.0429	0.8111
10 days BiGRU	0.0675	0.8161
3 days TEL	0.0599	0.8081
5 days TEL	0.0475	0.8121
10 days TEL	0.0455	0.8135

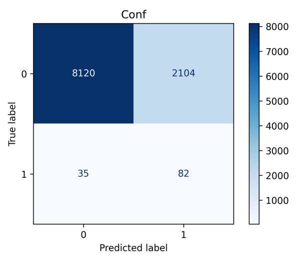
now the new cutoff probability we will pick for our model. On the training data the optimal cutoff value of around 0.002 to 0.04 was found for a recall of about 0.7 in the ROC curve. Applying this cutoff to the test data results in new confusion matrices for the models trained on the 10-day sequence as seen in Figure 7. As we see we can now detect more of the clicks but also get a huge amount of false positives.

3 Discussion

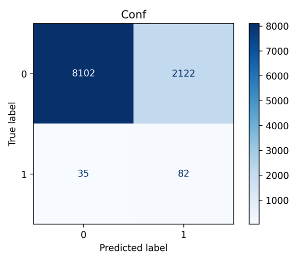
As the results show it is really hard to have the models produce good results on their own. Because of the imbalance in the data we cannot rely on the raw predictions obtained by minimizing the binary cross entropy during



(a) BiLSTM



(b) BiGRU



(c) TEL

Figure 7: Confusion matrices on test data after adjustment of cutoff value

training. If we did so, we would just predict every push notification as a "non-click" right from the get go, which is just not helpful and does not leverage the real power of machine learning. There are other ways however to deal with this imbalance, not shown in this report. For example, we could perform over- or undersampling. A real improvement could be found in the way we selected our features. For example one of the strongest predictors of future clicks might be past clicks. I suppose we could build a more powerful model just by utilizing the sequence of past clicks (click vs. non-click) and maybe the time of the day alone. This would however not be very useful in the general setting that we want to design interesting push notifications, but only in specific cases where the model would decide for a particular user if it makes sense to send the push notification or not. Another interesting feature would be time of the day. For example is it likely that people tend to click push notifications with different topics depending on if they are at work or in their leisure time. Also, comparisons with baseline models would be useful which was not done in this report.