National Taiwan University

Financial Technology

Final Project: Detection of Amazon Fake Reviews

Lecturer:
Che Lin (林澤)

Author: Tadeo Hepperle, Student ID: A11922105

December 21, 2022

Contents

1 Introduction

In this porject we took a look at a dataset consisting of amazon reviews that were either real or fake. Real meaning that the review was genuine and reflects the customers "real" opinion. Fake meaning that the review is probably forged with other motivations in mind, for example to boost or damage a products reputation. Because the majority of online shop customers rely on reviews when making their purchase decisions it is important for a platform like amazon to ensure the reviews reflect the real qualities of a product and are not misused as a marketing instrument by unethical businesses. To detect fake reviews we propose two different neural network models that both make heavy use of a BERT architecture to generate text embeddings that are useful in the binary classification. To see how our best model performs in action you can check out this little online game, where you can compete against our model yourself in realtime.

1.1 Description of the Data and Preprocessing

We used a dataset uploaded by Github user aayush210789 for our report, published here on Github. The data is from a corpus of reviews published by amazon and was internally labeled as fake or real. That means the labeling was done by Amazon and we sadly do not have advanced insight into the proceedings how exactly the labels were created. We do know however that the user behavior (for example if one user writes 100 reviews in one day) and the time of writing the review (e.g. 100 days after the product was bought) played a big role in the labeling. The dataset consists of a csv file with the following columns:

1. review_title: title of the review
2. review_text: content/text of the review, mostly moderately long text
3. rating: 1-5 star rating for the product
4. verified: 1 if there was a verified purchase before the review was submitted, else 0

5. label: 0 for a real review, 1 for a fake review
6. product_id: every Amazon product has a unique alphanumerical id
7. product_title: title of the product
8. category: the category of the product, in total 30 categories were present, such as PC, Baby, Books, ...

The CSV file contained exactly 10500 fake and 10500 real reviews such that imbalance was not an issue. The reviews encompassed 18857 Amazon products with unique IDs.

1.1.1 Scraping additional Data

We discovered that most of the products were still listed on Amazon under the same product_id. So we wrote a script that made 18857 requests to the corresponding amazon websites, that are available under "https://www.amazon.com/dp/product_id". For more than 86% of the products (16309 products) the requests went through and we were able to save the entire HTML content. With these 26.4 GB of data we were able to extract much more information. For each of the products we could extract from the HTML:

1. rating_count: how many ratings a product had in total
2. rating_avg: the average rating on a continuous scale from 1.0 to 5.0
3. rating1: percentage of ratings that were 1 star
4. ...
5. rating5: percentage of ratings that were 5 stars
6. reviews: for each review and the front page of the product:
    - review_title: title of the review
    - review_text: content/text of the review, mostly moderately long text
    - rating: 1-5 star rating for the product

- verified: 1 if there was a verified purchase before the review was submitted, else 0
- helpfulness: how many people found this review helpful

In total we were able to scrape 99995 additional reviews from the product front pages. These reviews can be helpful in fake review detection because they help us to see a certain suspicious review in context: If a review is overly positive and has 5 stars it might be fake, but if we see that the most helpful reviews for this product are also all very positive it might be more likely that the review in question is in fact genuine.

Using only products, for which we were able to scrape additional data, 18107 labeled reviews remained. There was a slight bias towards real reviews (52% real vs 48% fake), but that is not a big deal.

1.1.2 Splitting the Data

The data was randomly split into 3 parts:

- Train set: 70% of the reviews (12674)
- Validation set: 20% of the reviews (3641)
- Test set: 10% of the reviews (1792)

1.2 Description of Methods

We propose two different models for the detection of fake reviews, the firs one is called 3-BERT the second one is called Context Embedding Model. The 3-BERT model ignored the scraped data and just used the columns from the initial dataset. The Context Embedding Model tried to use the context information of the review (e.g. what do other reviews say about the product) to produce better predictions. In theory the Context Embedding Model should be more powerful and achieve better performance, because it has access to a superset of the data that the 3-BERT model has access to.

We use the Binary Cross Entropy Loss to train our models with gradient descent utilizing the Adam optimizer.

1.2.1 3-BERT Model

1.2.2 Context Embedding Model

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 **??** where we trained a BiGRU with a hidden size of 64 for 100 epochs.

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 **??** 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 **??** displays the accuracy of the 3 models on train, validation and test set each.

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 **??** seem very good. Such a high accuracy! But taking a closer look at the confusion matrices for all models, see Fig **??** 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.

../images/BiGRU100_trainlosses.png

Figure 1: Overfitting behavior of a BiGRU with $hidden\_size = 64$

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 **??** and Figure **??** respectively.

Table **??** 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 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.

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

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 **??**. 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 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

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

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 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 **??**. 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 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.

(a) Train and Validation Loss BiLSTM

(b) Train and Validation Loss GRU

(c) Train and Validation Loss Transformer Encoder Layer

Figure 2: Train and Validation Losses for all 3 models

../images/BiLSTM_train_conf.png ../images/BiLSTM_val_conf.png ../images/BiLSTM_test_conf.png

(a) BiLSTM on train (b) BiLSTM on validation (c) BiLSTM on test

../images/BiGRU_train_conf.png ../images/BiGRU_val_conf.png ../images/BiGRU_test_conf.png

(d) BiGRU on train (e) BiGRU on validation (f) BiGRU on test

../images/Transformer_train_conf.png ../images/Transformer_val_conf.png ../images/Transformer_test_conf.png

(g) TEL on train (h) TEL on validation (i) TEL on test

Figure 3: Confusion Matrices

../images/BiLSTM_train_prc.png

(a) BiLSTM on train

../images/BiLSTM_val_prc.png

(b) BiLSTM on validation

../images/BiLSTM_test_prc.png

(c) BiLSTM on test

../images/BiGRU_train_prc.png

(d) BiGRU on train

../images/BiGRU_val_prc.png

(e) BiGRU on validation

../images/BiGRU_test_prc.png

(f) BiGRU on test

../images/Transformer_train_prc.png

(g) TEL on train

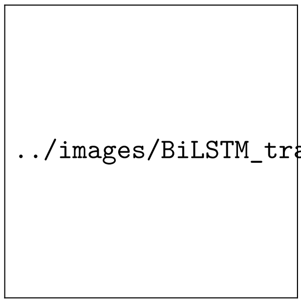
../images/Transformer_val_prc.png

(h) TEL on validation

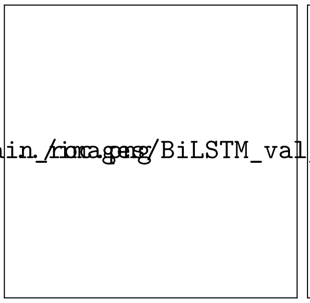
../images/Transformer_test_prc.png

(i) TEL on test

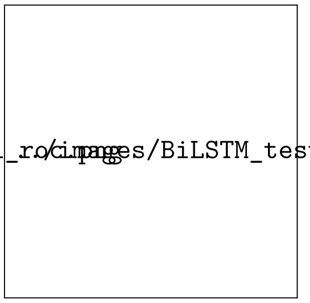
Figure 4: Precision recall curves



(a) BiLSTM on train

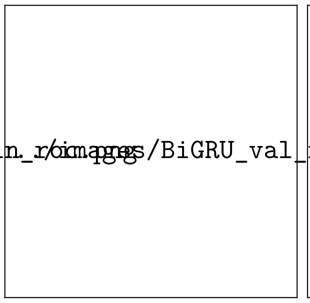


(b) BiLSTM on validation



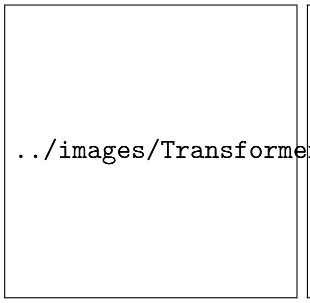
(c) BiLSTM on test

(d) BiGRU on train

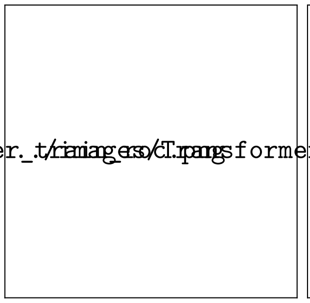


(e) BiGRU on validation

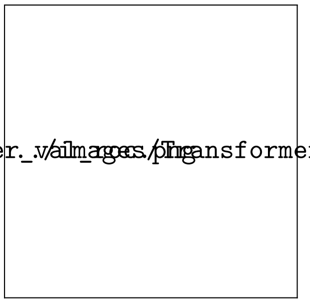
(f) BiGRU on test



(g) TEL on train



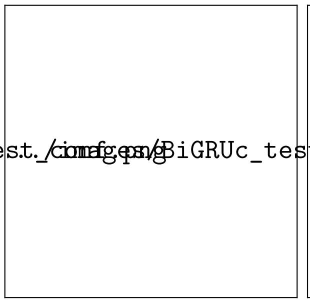
(h) TEL on validation



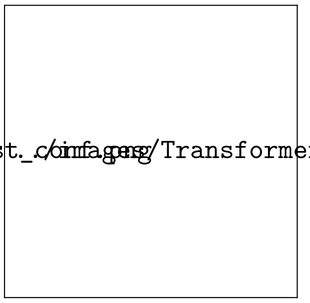
(i) TEL on test

Figure 5: receiver operating characteristic curves

(a) BiLSTM



(b) BiGRU



(c) TEL

Figure 6: Confusion matrices on test data after adjustment of cutoff value