Vuolt

Documentation

Mobile cloud distributed machine learning

Jakub Bujas, Dawid Dworak, Bartłomiej Grochal

Jakub Kudzia, Mateusz Owczarek, Michał Wolny

1. Introduction

In the age of mobile internet and cloud development an increasing need of exploiting full potential of those two concepts has been observed. It is clear that solutions combining those two are highly likely to happen in the following years to solve the problems such as resource scarcity, mobility and frequent disconnections.

Mobile cloud computing is a combination of the following: cloud and mobile computing. It can be addressed to solve numerous problems with too much time for the information round trip - sending data from mobile to the cloud, cloud computation and returning result to mobile.

There are four types of cloud-based resources in mobile cloud computing, namely distant immobile clouds, proximate non-mobile computing entities, mobile computing entities, and hybrid (combination of the other three model).

Giant clouds such as Amazon EC2 or Google Cloud Engine are in the distant immobile groups whereas cloudlet or surrogates are member of proximate immobile computing entities. Smartphones, tablets, handheld devices, and wearable computing devices are part of the third group of cloud-based resources which is proximate mobile computing entities.

In the text below there will be presented a project combining not only concepts of mobile and cloud but also multiple machine learning techniques to achieve most accurate results.

2. Aim of the project

Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source to help understand the social sentiment of the brand, product, service or conversation. With the recent advances in deep learning, the ability of algorithms to analyse text has improved considerably. Creative use of advanced artificial intelligence techniques can be an effective tool for doing in-depth research.

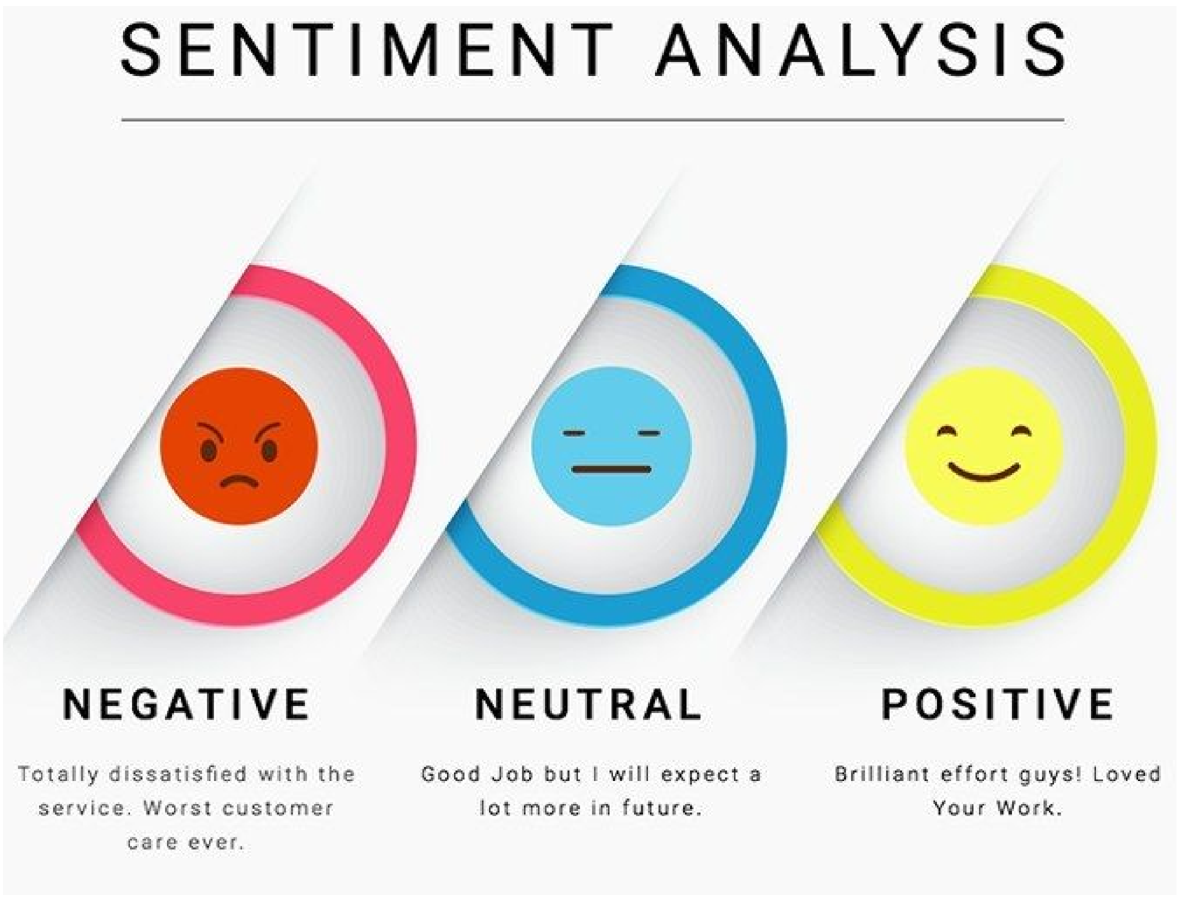
SENTIMENT ANALYSIS
NEGATIVE
Totally dissatisfied with the service. Worst customer care ever.
NEUTRAL
Good Job but I will expect a lot more in future.
POSITIVE
Brilliant effort guys! Loved Your Work.

Fig. 1: Sentiment analysis basic example

Sentiment analysis problem has been chosen as a theme of this project since there are numerous machine learning approaches to solve it as well as it requires datasets which are widely accessible for public.

That said at the very highest level, our application should take a set of words as an input and transform them into a sentiment - decide if the sentence was positive or negative.

That said, the first goal of the project is to ensure division of the pretrained classification models - possibly one in cloud and one per each mobile user. This would allow constantly up-to-date personal local model offline as well as single global one with data gathered from all of the users. Without the need of having internet connection user can choose to use a possibly less precise classifier, however personally tuned and always ready to run.

Secondly an investigation if the model distribution has impact on classifiers correctness should be performed to validate our introductory predictions.

A fine addition would be to compare two different machine learning approaches performance and correctness and use both to create an ultimate classifier taking best of both worlds.

3. Architecture

To provide the best possible solution we have decided to release the workload from mobile devices and pass the responsibility for heavy calculations to the cloud. The main server contains the global model, which is a source containing a combined knowledge from all connected devices. There we put the pretrained LSTM model basing on the large tweets dataset. A single mobile machine holds a model with characteristic data per user.

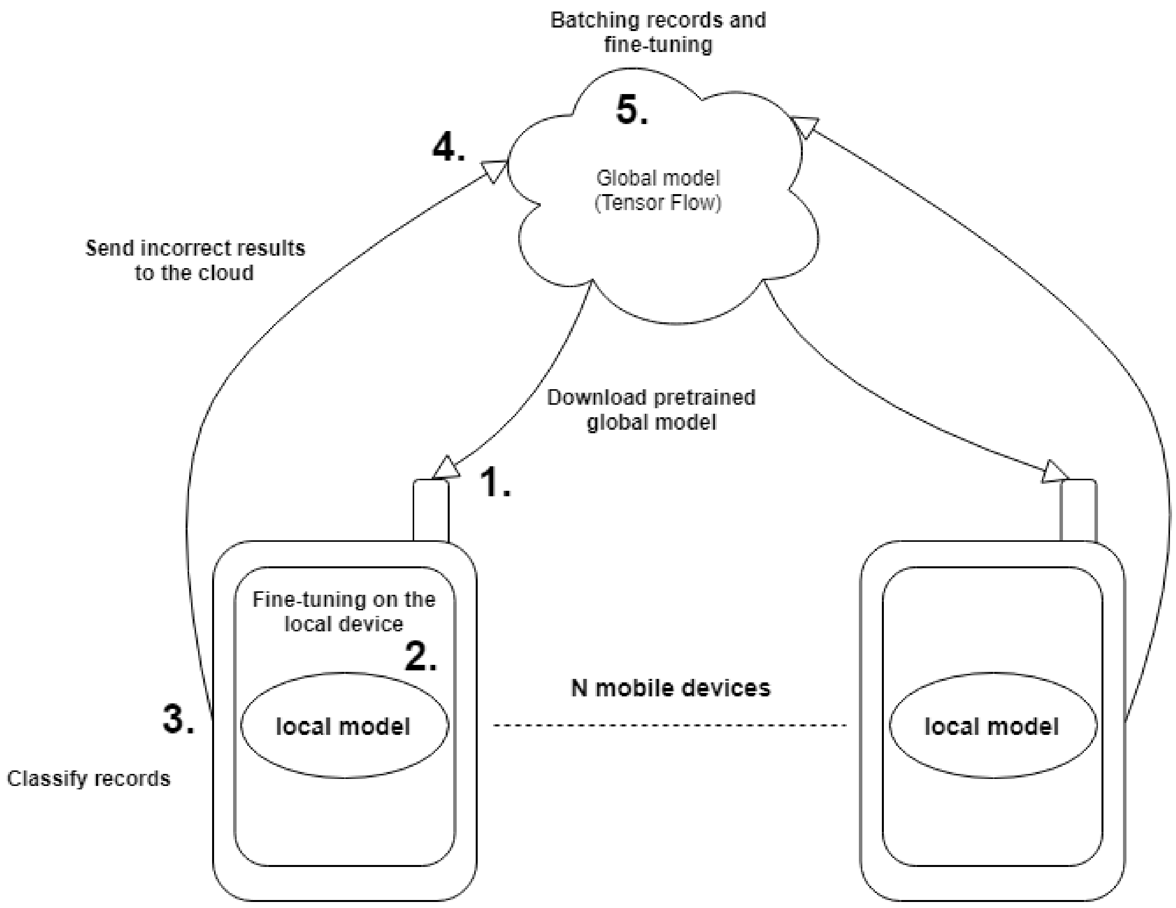
Batching records and fine-tuning
5.
4.
Global model (Tensor Flow)
Send incorrect results to the cloud
Download pretrained global model
1.
Fine-tuning on the local device
2.
N mobile devices
3.
local model
local model
Classify records

Fig. 2: Information flow between particular entities.

A Figure 2 illustrates the information flow between particular entities. The scenario of the single message loop may look like this:

1. The global model pretrained on the server in the cloud is downloaded by mobile device.
2. The gathered model is fine-tuned using the local data on the device.
3. When new record arrives to the device, the user classifies it using both local and global models.
4. Basing on the current strategy the decision which verdict is better is made. If the result is incorrect it is sent to the cloud.
5. The global server batches incorrectly classified records and once per a period of time it makes fine-tuning on its own model.

4. Solution

For the purpose of this project we have created two different machine learning models. It is crucial to have a lightweight model on the mobile device and a complex one in the cloud, which can be dedicated for long term calculations. Two models have been picked: Naive Bayes and LSTM RNN.

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It assumes that the value of a particular feature is independent of the value of any other feature, given the class variable. A naive Bayes classifier considers each of these features to contribute independently to all of the probabilities, regardless of any possible correlations between the features. For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification. That said we have chosen to use naive Bayes for local training model.

As for the global one we have decided to follow existing classic solutions of the problem and use LSTM neural networks.
These are Long Short Term Memory networks – special kind of RNN (Recursive Neural Networks), capable of learning long-term dependencies. They are now widely

used on a large variety of problems giving exceptionally good results, however they are explicitly designed to avoid the long-term dependency problem.
A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for "remembering" values over arbitrary time intervals, hence the word "memory" in LSTM. Each of the three gates can be thought of as a "conventional" artificial neuron, as in a multi-layer (or feedforward) neural network: that is, they compute an activation (using an activation function) of a weighted sum. Intuitively, they can be thought as regulators of the flow of values that goes through the connections of the LSTM, hence the denotation "gate". There are connections between these gates and the cell.
An LSTM is well-suited to classify, process and predict time series given time lags of unknown size and duration between important events. LSTMs were developed to deal with the exploding and vanishing gradient problem when training traditional RNNs. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods in numerous applications.
Besides implementing those two models an android application has been introduced with simple GUI to cover need for feeding user data with proper label which illustrates Figure 3.

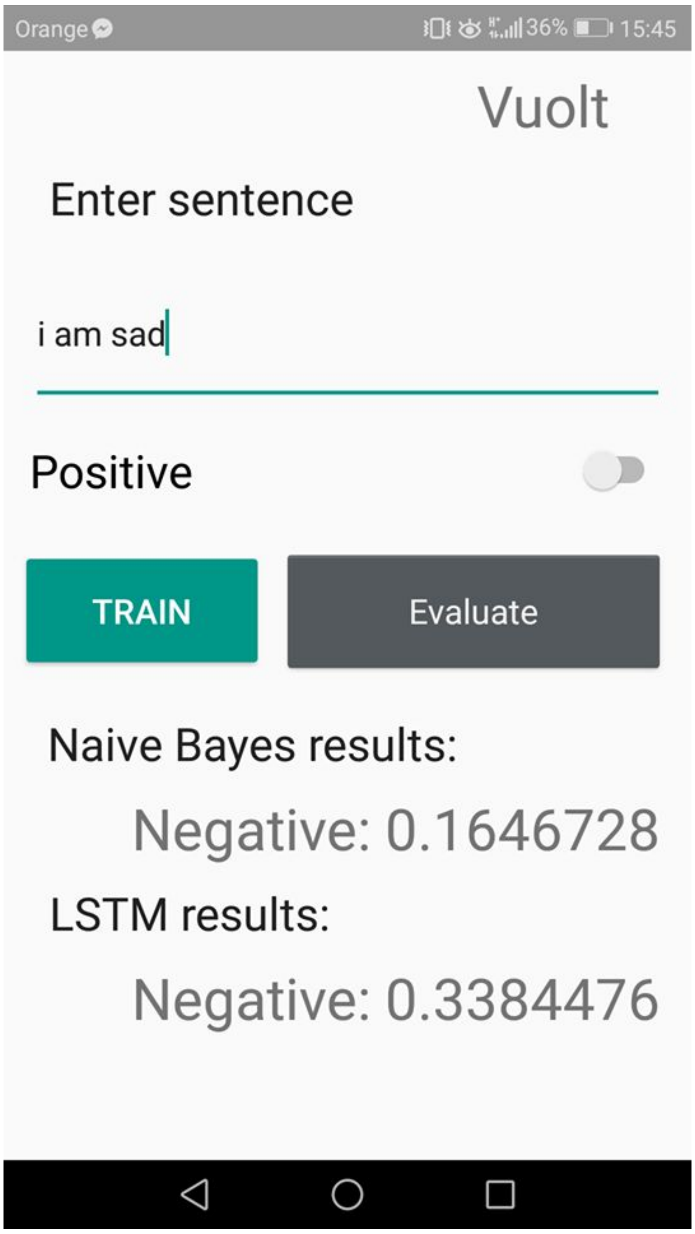

Fig.3: Vuolt mobile user interface.

The front-end side of the application is designed to provide the simplest and affordable way of message labeling. The user can enter the sentence to the application and click the 'Evaluate' button. It results with the classification from naive Bayes and LSTM models. If the user decides that the evaluation is incorrect, he can label the sentence using toggle button and fine-tune the model. Figure 4 presents basic schema of LSTM Fine-tuning:

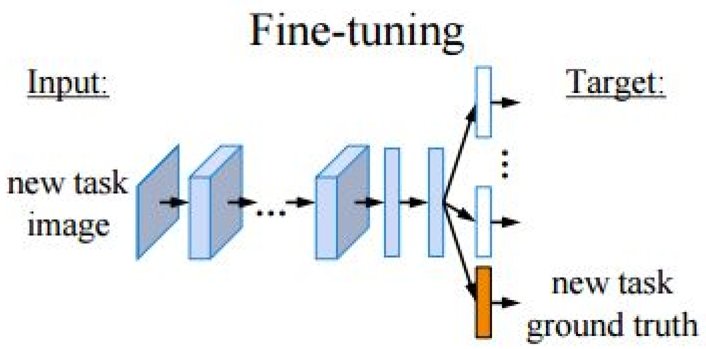

Fig.4: Fine-tuning method.

When it comes to the model hybridization, there are many ways to do it. We have picked three the most common approaches and compared them:

1. Arbitrarily choose one of the results - when results are compatible there is no problem but when they are inconsistent we have to designate one model that has higher priority.
2. Pick a function of resulting probabilities - we can achieve the result by mixing outputs from both models.
3. Decide randomly - sometimes it can be the best possible solution. We determine the probability percentages with which we take the result of a given model in a case of conflict.

5. Results evaluation

To correctly evaluate results, three different metrics has been introduced.
First one is Precision.
It attempts to answer the following question:
What proportion of positive identifications was actually correct?

$$Precision = TP / (TP + FP)$$

Formula 1: Precision equation

Precision is defined as described in Formula 1, where TP is number of true positives identifications and FP: false positive identifications.

Recall attempts to answer the following question:
What proportion of actual positives was identified correctly?

$$Recall = TP / (TP + FN)$$

Formula 2: Recall equation

Mathematically, recall is defined as in Formula 2, where TP is number of true positives identifications and FN: false negative identifications.

Third metric is the precision-recall curve. It shows the tradeoff between precision and recall for different threshold.

Last metric is the F1 score presented in Formula 3 (also F-score or F-measure) being a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score.
The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0:

$$F_1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Formula 3: F1 score equation

5.1. LSTM Pretraining results

For LSTM pretraining we obtained following accuracy and loss presented in Figures 5 and 6. As we can see, accuracy has reached level 70-75% after 10 000 iterations and stayed on that level:

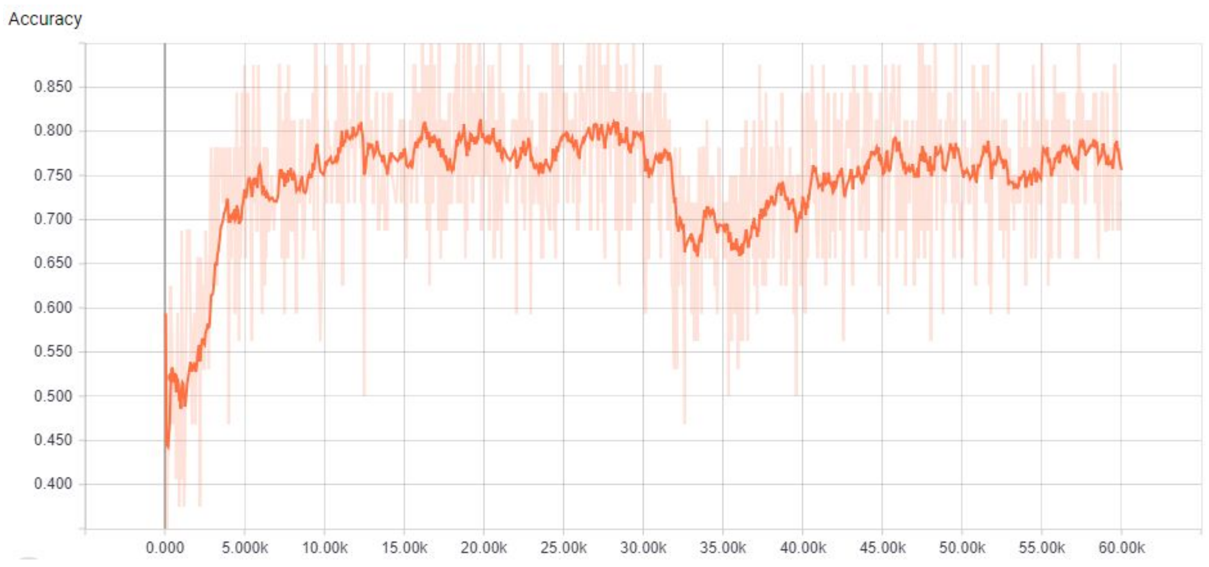

Fig.5: LSTM method accuracy (X axis presents number of iterations)

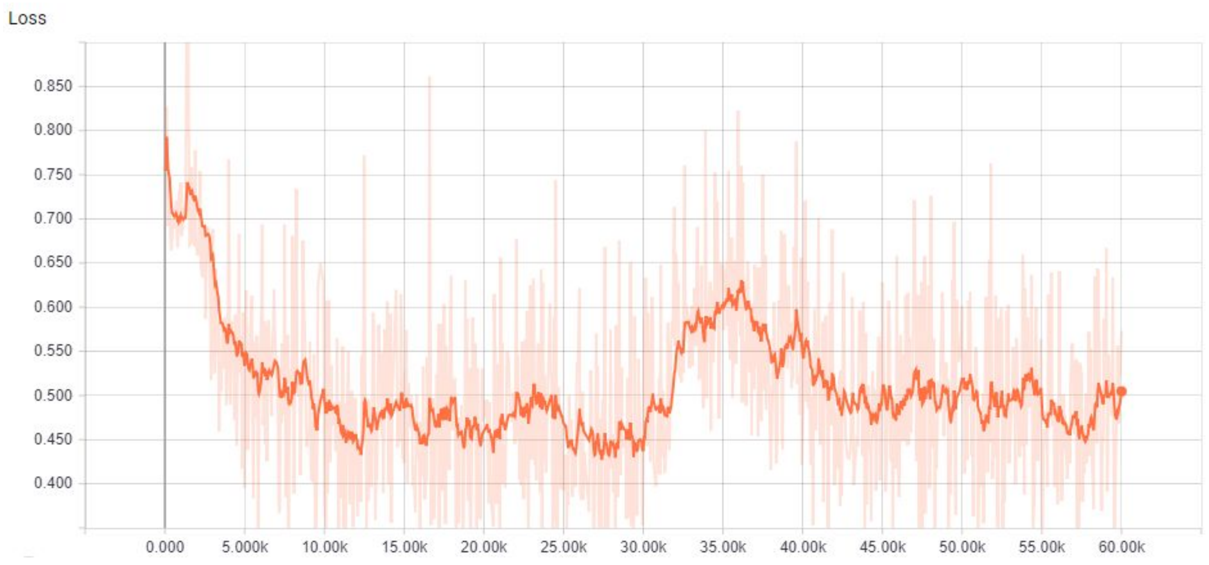

Fig.6: LSTM method loss (X axis presents number of iterations)

Our pretrained global model achieved 76.46% correct predictions ratio, according to the results presented in Table 1.

```
Incorrect predictions: 4646 (23.54%)

             precision    recall  f1-score   support

          0       0.77      0.76      0.76      9856
          1       0.76      0.77      0.77      9888

avg / total       0.76      0.76      0.76     19744
```

Table 1: Global model results

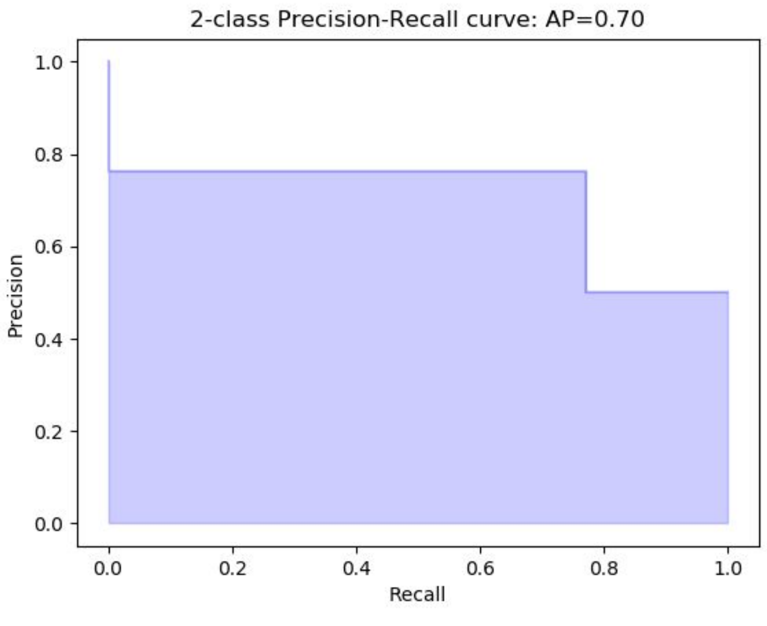

Fig.7: Precision-recall curve for global model

High area under the curve in Figure 7 can be interpreted with high recall and high precision of the model, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

5.2. Local model results

As for the local model, predictions ratio was slightly lower, about 1.5% (Table 2) with similar precision-recall curve (Figure 8):

```
Incorrect predictions: 4924 (24.95%)

             precision    recall  f1-score   support

          0       0.72      0.82      0.77      9856
          1       0.79      0.69      0.73      9878

avg / total       0.75      0.75      0.75     19734
```

Table 2: Local model results

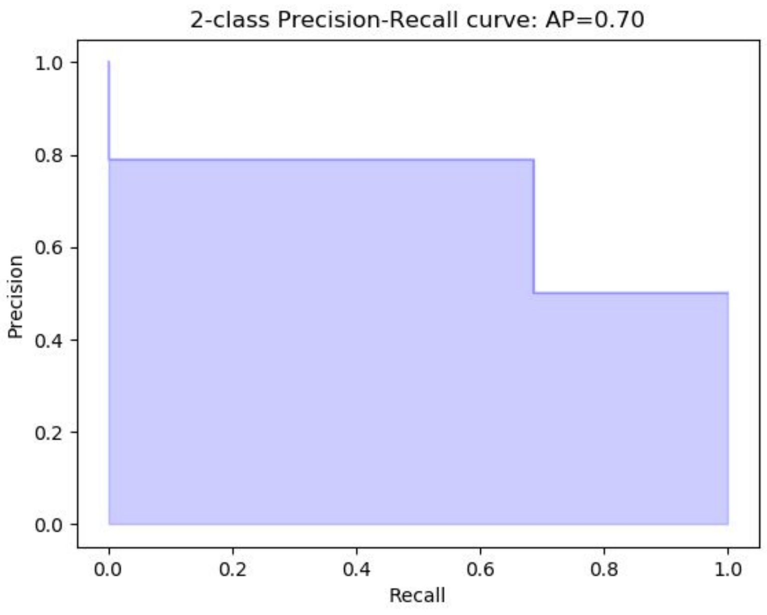

Fig.8: Precision-recall curve for local model

5.3. Hybrid model results

For this problem we achieved best results (almost 89% prediction ratio) with solving conflicts randomly with 65% chance of using LSTM and 35% of using Naive Bayes method. Table 3 and Figure 9 present results for this approach:

```
Incorrect predictions: 2182 (11.06%)

             precision    recall  f1-score   support

          0       0.75      0.78      0.76      9856
          1       0.77      0.74      0.75      9878

avg / total       0.76      0.76      0.76     19734
```

Table 3: Results for our hybrid model (LSTM: 65%, Naive Bayes: 35%)

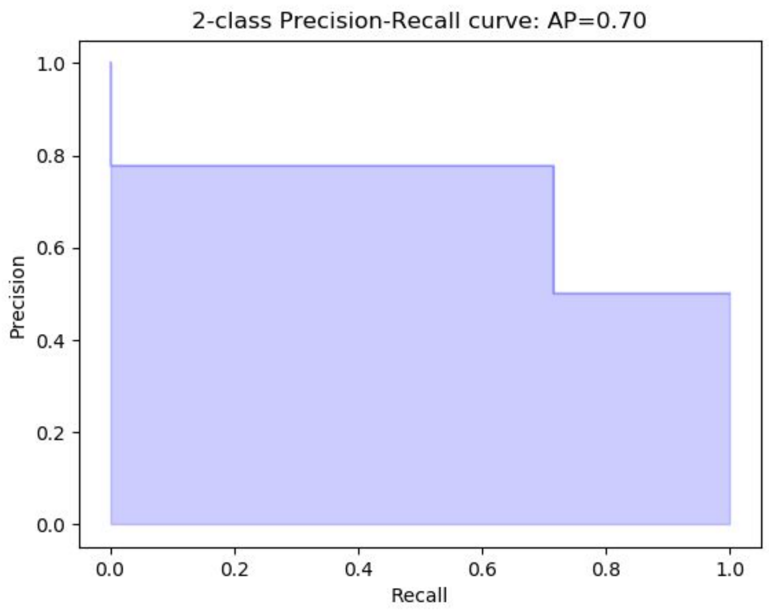

Fig.9: Precision-recall curve for for hybrid model (LSTM: 65%, Naive Bayes: 35%)

5.4 Fine tuning results

After pretraining we used the fine-tuning method on LSTM to enhance the results. Figures 10, 11 present analogical metrics:

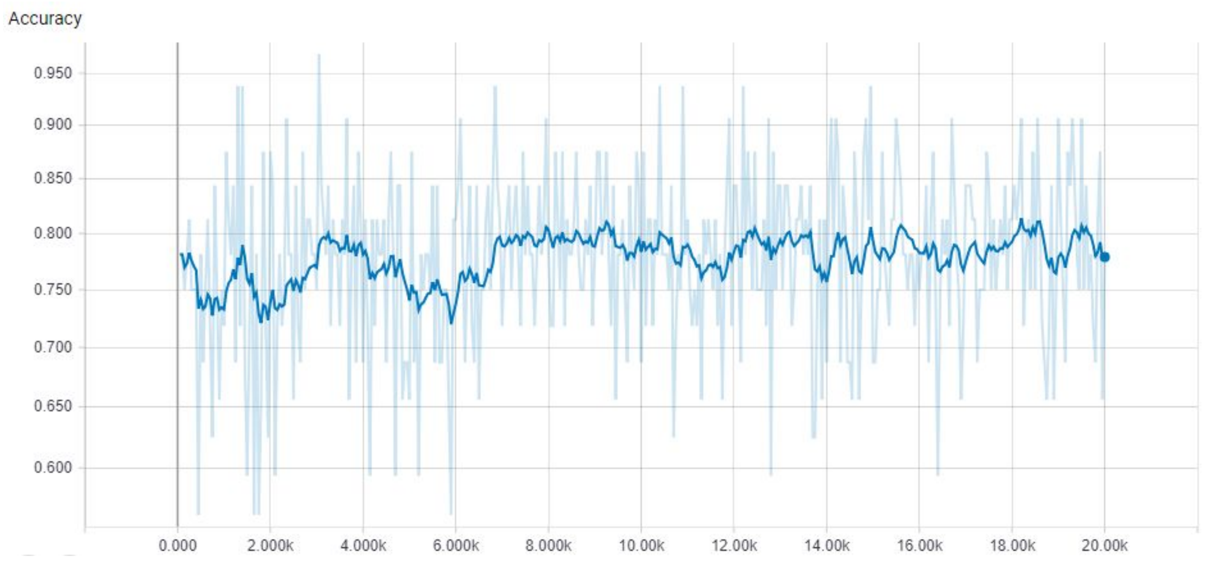

Fig. 10: Fine-tuned LSTM method accuracy

(X axis presents number of iterations)

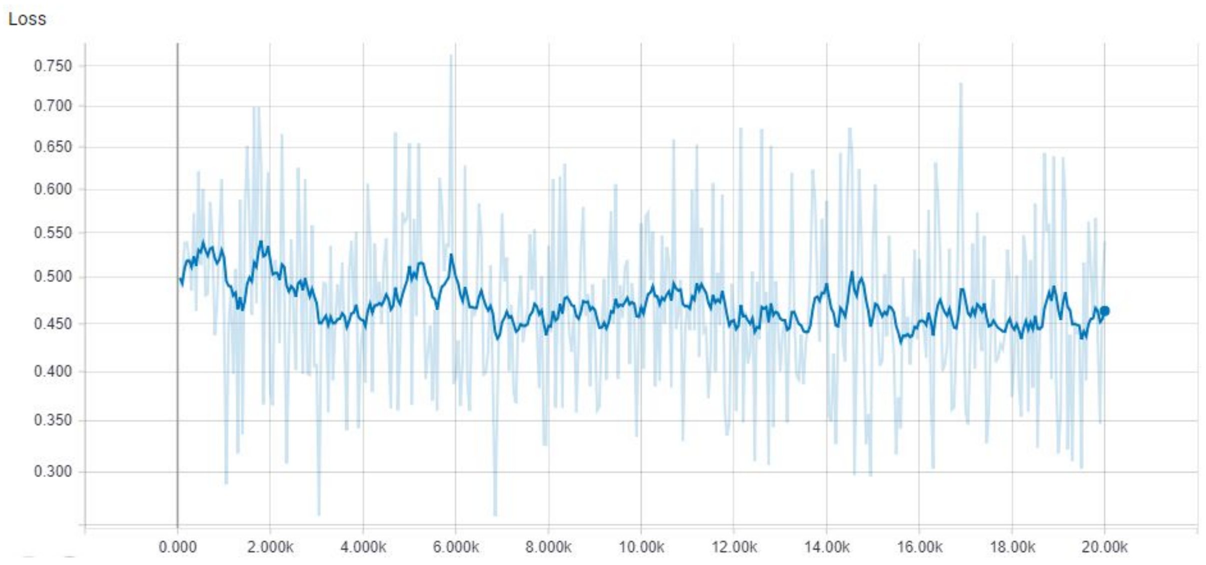

Fig. 11: Fine-tuned LSTM method loss

(X axis presents number of iterations)

As we can see in Fig. 10 and 11, fine-tuning has improved results in comparison to Fig. 5 and 6.

```
Before:
     Incorrect predictions: 4646 (23.54%)

After:
     Incorrect predictions: 4433 (22.46%)
```

Table 4: Pre-trained vs Fine-tuned global model predictions ratio

Table 4 presents improvement for fine-tuned global mode. Analogical fine-tuning techniques were used to enhance the results on local and hybrid models. Tables 5 and 6 present analogical improvements for predictions ratios.

```
Before:
     Incorrect predictions: 4924 (24.95%)

After:
     Incorrect predictions: 4507 (22.84%)
```

Table 5: Pre-trained vs Fine-tuned local model predictions ratio

```
Before:
     Incorrect predictions: 2182 (11.06%)

After:
     Incorrect predictions: 1983 (10.05%)
```

Table 6: Pre-trained vs Fine-tuned hybrid model predictions ratio

6. Summary

Within this project we created a full-blown architecture for training and employing machine learning algorithms in a distributed mobile environment in order to predict sentiment of incoming e-mails, SMSes etc. We used two different learning models - the LSTM neural network holding global knowledge and the Naive Bayes algorithm performing predictions based on local, personalized samples. Additionally, we evaluated multiple scenarios of global and local knowledge hybridization, especially in case of conflicting predictions returned by both models. The evaluation results confirm that the combination of LSTM and Naive Bayes models allows to obtain better results (in terms of prediction accuracy) than for both models working in isolation. Surprisingly, it turned out that the best results (with 90% prediction accuracy) in case of inference outcomes conflict were obtained for a combined model with randomly picking one of the labels returned by base classifiers (however a probability distribution of the random generator should be accordingly weighted).