Botanicum: a Telegram Bot for Tree Classification

Daria Korotaeva, Maksim Khlopotov, Anastasiia Makarenko, Ekaterina Chikshova, Natalia Startseva, Anastasiia Chernysheva
ITMO University
Saint Petersburg, Russia
daria.korotayeva@yandex.ru, khlopotov@corp.ifmo.ru, anastasiamakarenko95@yandex.ru, katya.chikshova@gmail.com, st.natalie.eilatan@gmail.com, anastasiia.chernysheva.2305@gmail.com

***Abstract*—The article presents Botanicum – a telegram bot for tree classification based on the image of a leaf. In this work, we used our previously developed classification model which underlies the bot. Botanicum can identify 20 different tree species typical for Russian flora. The process of user interaction includes a simple chat and a short instruction on how to take a photo. After image verification, the bot either outputs error description or the classification results. We managed to increase the classification accuracy to 97,8% by outputting top results instead of one single option.**

I. INTRODUCTION

The ability to recognize plant species in situ is an actual problem that can be solved by computer vision methods. One of the features of this research area is the incapability to find a universal solution due to the diversity of flora in different geographical areas. Despite the considerable number of approaches, there are not so many final products designed for plant recognition, and none of them covers the Russian audience. We have built our dataset using several public ones and developed an algorithm for classification of trees typical to European Russia, which showed high accuracy.

Telegram Messenger was used to implement the algorithm as it allows developers to create bots with a diverse functionality. The main advantage of this approach is the support of most platforms, as well as the ability to quickly implement and test the application. In this article, we present a telegram bot that allows users to define 20 species of trees by the image of a leaf.

The paper is structured as follows: Section II reviews related works including the research on which this paper is based. Section III provides an overview of the tree classifier. Section IV emphasizes on design principles and reasons behind component and technology selection. Section V describes the process of communicating with the bot. Finally, Section VI discusses results achieved by the bot and concludes the work.

II. RELATED WORK

Methods for automatic tree recognition based on a leaf image have been studied previously and several working applications have been implemented. Kumar et al. demonstrated Leafsnap, a mobile application for identifying plant species using automatic visual recognition. The system identifies 184 tree species of the Northeastern United States from photographs of their leaves [1].

Çuğu et al. created a tree classification system called Treelogy. It fuses deep representations with hand-crafted features obtained from leaf images to perform leaf-based plant classification. The system identifies the species from a dataset of 57 trees [2].

Belhumeur et al. describe a working computer vision system that aids in the identification of plant species. It is currently in use by botanists at the Smithsonian Institution National Museum of Natural History. The system extracts the leaf shape from a user photograph of an isolated leaf on a blank background and matches it to the shape of leaves of known species [3].

P. Novotný and T. Suk [4] developed a web application for tree recognition based on a leaf image. Users may input the size of a leaf and its type and adjust thresholding and resolution. A dataset called Middle European Woody Plants used for the system implementation currently consists of 199 species and was partly used for building our own dataset.

Pierre Barré et al. described a system for automatic plant species identification trained on various datasets [5]. The authors implemented a deep learning convolutional neural network and compared it to hand-crafted algorithms which are usually applied for solving plant identification tasks. The high results achieved in the work showed advantages of applying deep learning. The system is available for downloading and testing. However, an installed deep learning environment and some programming skills are required, so ordinary users cannot experience the system.

With the growing computational power in smartphones, more mobile recognition systems are being developed. A real-time system developed by Yoshiyuki Kawano and Keiji Yanai can help users recognize food and estimate its nutritional characteristics [6]. Unlike most recognition systems, FoodCam doesn't upload images to a server, but users are required to manually draw the bounding boxes for correct food recognition. The authors achieved good results in food classification and implemented the system as a mobile application for smartphones on Android platform.

The work presented in this paper is the implementation of the study published in [7] as a Telegram Bot. In the paper classification model is described. The method of the nearest neighbors was used for the classifier. The model was trained on 1464 photographs of 18 species of trees that are typical for Russian flora. The accuracy of the developed algorithm showed a high value - more than 95% on the test sample.

III. THE TREE CLASSIFIER

For our classifier we used a dataset based on MEW12 [4] and Swedish Leaves Dataset [8] taking into account the characteristics of the local flora. More information about the dataset is given in a previous study [7] devoted to the method of extracting features for tree classification.

We completed the previous dataset with three classes: common beech (*fagus sylvatica*), common barberry (*berberis vulgaris*) and rowan (*sorbus aucuparia*), thus obtaining 1667 images of 20 tree species (Fig. 1). The dataset can be extended by using various sources. In the future, we plan to do so and complete it with images of tree species typical for Russian flora.

The implementation of the algorithm was carried out in the Python programming language using the OpenCV computer vision library [9] and the machine learning library scikit-learn [10]. Python provides a powerful environment for learning and experimenting with Computer Vision and Machine Learning together with OpenCV and scikit-learn. OpenCV is one of the most popular image processing libraries and it is suitable for solving a wide range of tasks, including the one set before us. Scikit-learn is currently the most widely used library for Machine Learning applications. Both mentioned libraries are open source and free to use.

During feature engineering, we reduced the number of extracted features to 20 in comparison with the previous work. A new feature, *circularity*, was added, which increased the accuracy to 1%. Circularity is a shape descriptor that can mathematically indicate the degree of similarity to a perfect circle. It is a ratio of area of the leaf to its perimeter squared multiplied by 4π:

$$4\pi \cdot \frac{a}{p^2} \quad (1)$$

where a – area of an object (leaf), p – perimeter of an object. Circularity with a value of 1 indicates a perfect circle [11].

Even though the size of a leaf is considered an important characteristic, we cannot use it as a feature, since we are not aware of the size of a leaf based solely on its photograph. Moreover, photos of leaves are likely to be made in different conditions. Our program code includes scaling images to approximately the same size. Large images are shrinked while considerably small images are enlarged.

For the classification accuracy evaluation, we divided our images on the training and testing set which accounted for 75% and 25% of the dataset, respectively. The accuracy obtained on the testing set was 93,7%, and the confusion matrix indicates that most errors are single leaves of different classes and a slightly more visible confusion occurs in species belonging to one genus, *Betula*. (Fig. 2).

In some cases, incorrect classification occurred when the probabilities of two classes were relatively close. We decided to output all classes with nonzero probability, so that the result could be either one or several classes ranked according to their probability, and the classification accuracy increased to 97,8%.

For the classification we used the nearest-neighbor algorithm (k-NN) with k = 3. This value was chosen after a series of experimental measurements of the classification accuracy with various values of k. Since we needed a

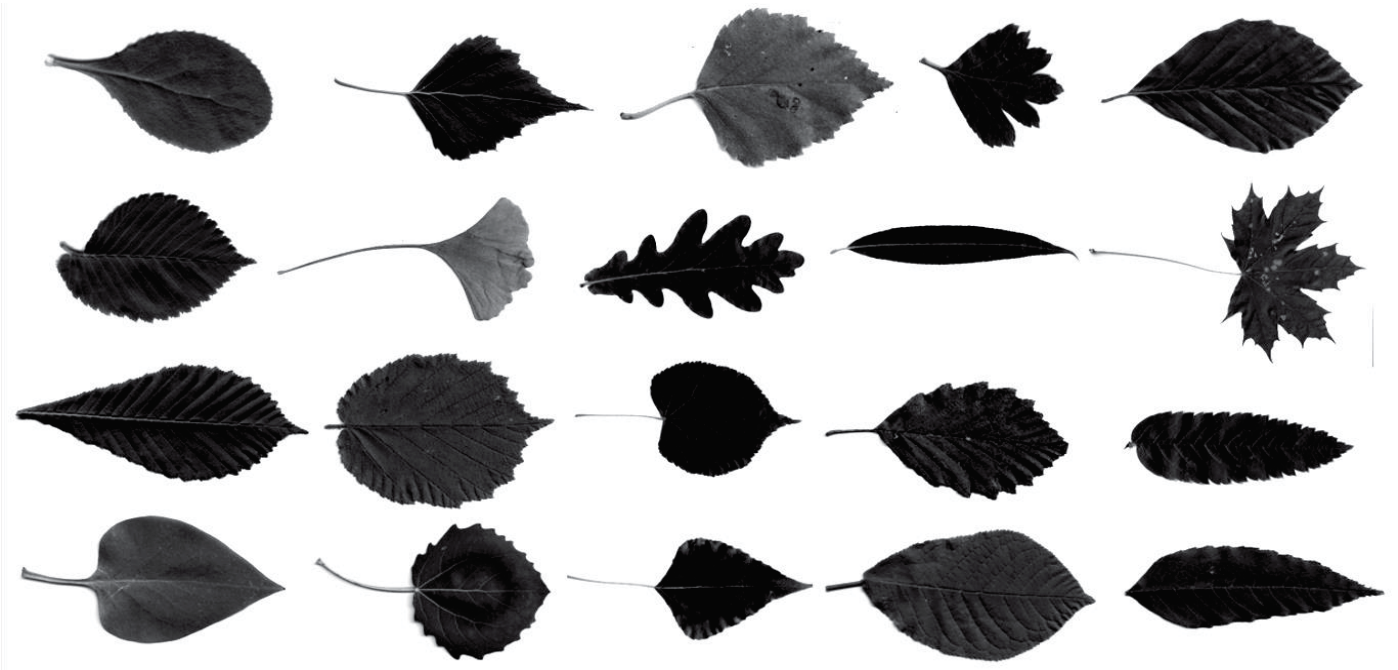

Fig. 1. Examples of leaves of all 20 species

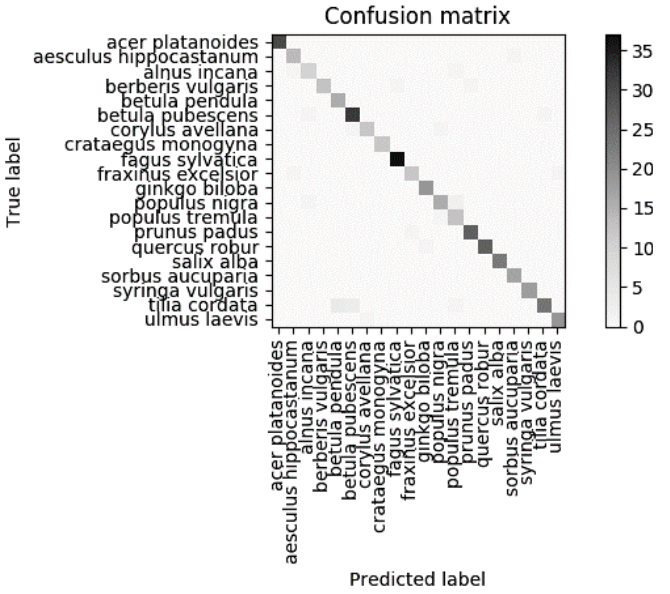

Fig. 2. Confusion matrix

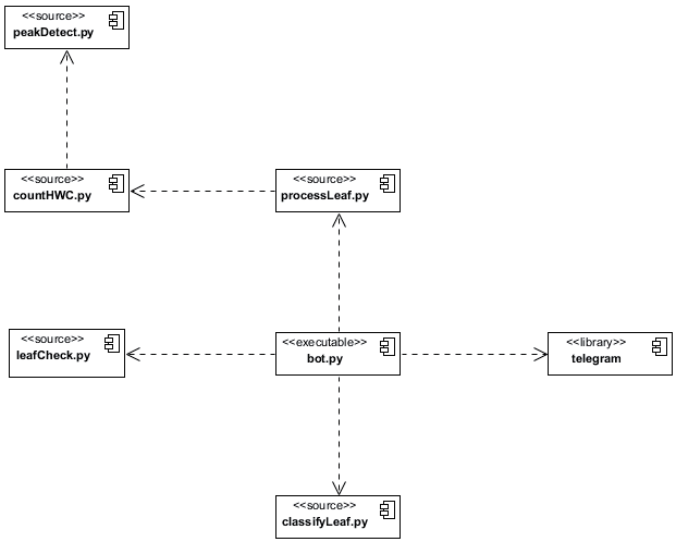

Fig. 3. Component diagram

classification model that would allow us to output several possible results, we were looking for the largest value of k among those that allowed us to obtain the highest accuracy. The number of possible classes in the output depends on the number of neighbors in k-NN, and our classifier may output up to 3 classes as a result. While other classification algorithms performed with a lower or similar accuracy, it was decided to keep the k-NN as the simplest one while our project is still in development.

IV. BOTANICUM BOT

Telegram Bot is an account in Telegram driven by software, which often has the functions of artificial intelligence. Bots have become a popular tool for any purpose. They can teach, play, use search, remind of current affairs, integrate with other systems or even transfer commands to other services.

The main task of the bot is an automatic response after the command entered by the user. At the same time, working directly through the Telegram interface, the program simulates the actions of a live user, due to which the use of such a bot is much more convenient and understandable. That is why, people who develop their ideas through the Internet, use the capabilities of bots for several reasons: bots allow to use the new channel of communication with the target audience, and they quickly perform monotonous work, allowing to discharge hands, letting them do more valuable work.

For the implementation of our classifier, we have created a telegram bot @botanicum_bot using a python-telegram-bot library [12]. The component diagram describing the structural relationships between program elements is shown on Fig. 3.

The components of the developed system are python code files. The executable file “bot.py” contains the main code of the program including user instructions. This file requires three source files and the python-telegram-bot library, which provides python interface for the Telegram Bot API and some necessary for telegram bot development functions. The first required file is “leafCheck.py”. It contains the algorithm of verifying whether the image meets the requirements for the further processing. The second one, “processLeaf.py”, is responsible for feature extraction. This file, in turn, uses “countHWC.py” functions needed for peak related features. For peak detection we used existed implementation [13] stored in the “peakDetect.py” file.

One of the main innovations in the algorithm is the program implementation of image verification. Discarding of unsuitable images reduces the load on the server. Low-quality photos of leaves can lead to incorrect classification. To ensure the correct work of the classifier, a leaf should be photographed intact, with a petiole, and must occupy more than 10% of the image space and be on a light neutral background.

Information on how to properly take a photo of a leaf is provided to the user, but often images have poor quality and can cause errors in the classification process. To avoid such errors, a “leafCheck” module was implemented.

The process of image verification occurs in parallel with the main image processing, and the error search is performed at its different stages. After binarization, small objects are removed, and if the leaf is smaller or larger than the specified value, an appropriate error will be issued. During the search for the petiole, the background noise is also checked. In case several contours are detected during the object shape processing, an error is issued. The object boundaries must not touch the image borders, except for the thin elements (e.g. the petiole or the leaf tip), since the leaf shape is its main feature, and must be present in the image wholly. A too light or too dark image will be incorrectly processed during binarization, and a corresponding verification is also performed. In the absence of errors, the processed image is then sent to the "processLeaf" module for feature extraction.

It is important to note that the object on the image may be anything as long as the image meets the requirements. In this case, the result of classification is unpredictable. We do not evaluate whether the image contains a leaf or any other object due to high diversity of leaves, and the "leafCheck" module is implemented in order to filter the images that might cause problems during processing.

While our target audience are users who live in Russia, a Russian version of the bot @TreeLeafBot was created. Having the same structure and classification model, the bot operates entirely in Russian language. Common names are used to refer to tree species and description buttons provide links to the Russian Wikipedia.

V. User Interaction with the Bot

The process of user interaction with the bot is shown on activity diagram (Fig. 4). The bot contains a short instruction on how to take a picture of a leaf. After receiving an image, Botanicum runs a "leafCheck" module in order to verify that the image meets the requirements for the further processing. A picture must be taken on a light neutral background with only one leaf present. In case the uploaded image doesn't fit this description, the user is notified of the error and its possible causes.

The following errors may be raised during image processing:

- The leaf is too small
- The background contains noise
- Several contours of objects are detected
- The leaf exceeds image borders
- The image is too dark or too light

Examples of such errors are shown on Fig. 5 and Fig. 6

In case there were no problems detected, the image is processed for feature extraction and classification and a message on successful verification is sent to the user. The result is displayed as a short chat message with the name of one or several possible species (Fig. 7, Fig. 9). The user is also given a button for each species allowing them to read a short piece of Wikipedia article along with a link to that article (Fig.8).

Apart from image processing, the bot supports a simple chat with users by responding to text messages with random lines, e.g. "Please send me a photo of the leaf", and via the \help menu which provides access to the following commands:

- \start
- \help
- \howto
- \trees_list

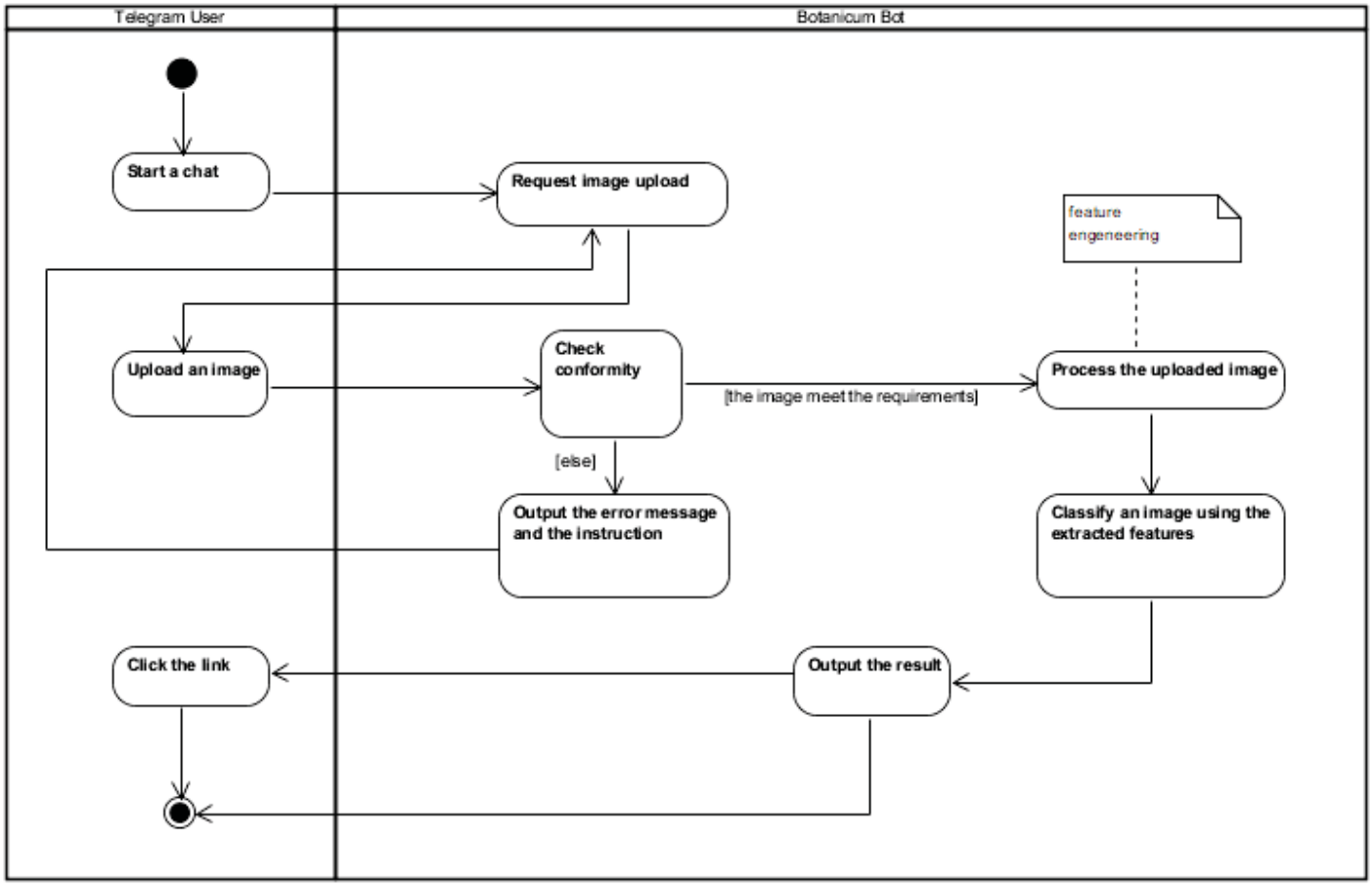

Fig. 4. Activity diagram

Fig. 5. Error processing

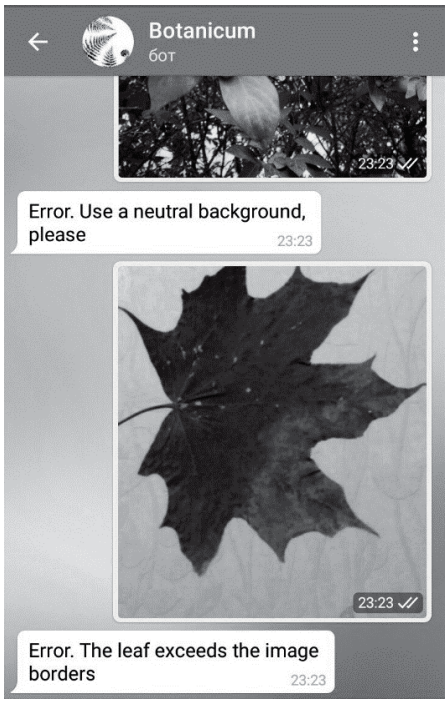

Fig. 6. Error processing

The \howto command provides the instruction on how to take a correct photograph of a leaf, and \trees_list reveals description buttons for all tree species recognized by the bot.

To test our bot we asked a group of people with different mobile platforms and browsers to perform a series of experiments. The group tested all of the bot's functions with real-taken leaves pictures.

Since the tests were conducted in late autumn, it was impossible to perform them outdoors, as there was no foliage left on the trees and the participants had to use our personal herbarium instead. While the bot showed good working capacity on every device and worked even on forwarded messages, it is still expected that a series of experiments in situ with real leaves will be performed in the summer.

VI. Conclusion and Future Work

In this article, a telegram bot for tree classification was described. An image of a leaf is required for tree recognition. Users may take pictures of leaves and upload them directly to the chat. As the images are rescaled, their size is not important as long as the images meet the requirements described in the instruction provided. The current dataset is focused on Russian flora and the bot can currently recognize leaves of 20 tree species. A k-NN classification algorithm was used with k = 3, and the accuracy of 93% was obtained on the testing set. By setting the classifier to output up to 3 possible species we increased the accuracy to 97%.

The bot is able to imitate a simple chat and show an instruction on how to correctly take a photograph of a leaf. Users are provided with buttons, pressing which allows them to receive information on tree species along with links for

Fig. 7. Example of a correctly classified leaf

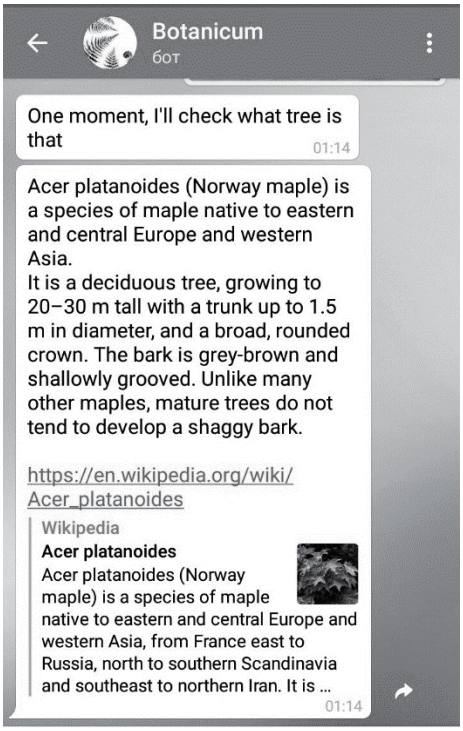

Fig. 8. A short description given to a user

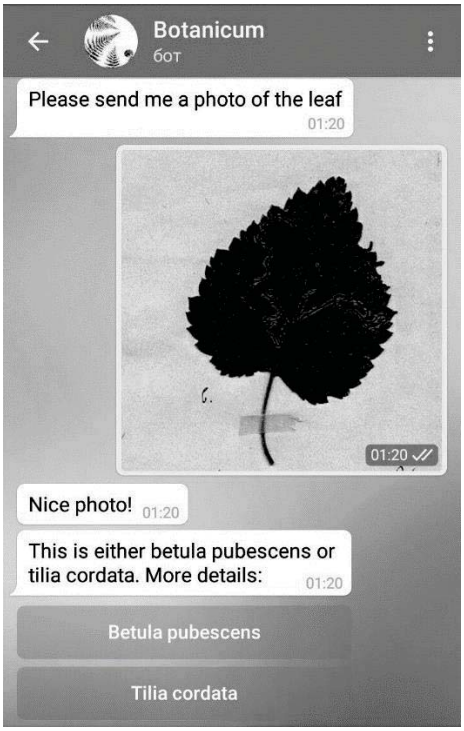

Fig. 9. Two possible results are shown

further reading. Various error descriptions may be shown in case the image doesn't meet the requirements.

Our further work will be focused on the dataset extension along with developing new feature extraction methods in order to obtain high classification accuracy. It is also likely that we might use other plant organs for a more accurate classification during blooming when many species become allergenic. We will focus mainly on the flora of Saint Petersburg, Russia.

We are already in process of cooperation with Saint Petersburg Botanical Garden and Institute in order to collect our own leaf dataset and discover new features for better classification performance. Our next step is implementing the geographic data extraction, which will help us create an interactive tree map as well as increase accuracy of classification of species with known location. Thus, the system will be valuable to green area specialists and urban planners.

As we go forward, we plan to implement more up-to-date machine learning methods such as convolutional neural network in order to improve our algorithm. Our dataset is large enough to conduct this kind of experiments, so we will evaluate the results by comparing them with those we have obtained using our current algorithm.

REFERENCES

[1] N. Kumar, P. Belhumeur, A. Biswas, D. Jacobs, W. Kress, I. Lopez and J. Soares, "Leafsnap: A Computer Vision System for Automatic Plant Species Identification", *Computer Vision – ECCV 2012*, 2012, pp. 502-516.

[2] İ. Çuğu, E. Şener, Ç. Erciyes, B. Balcı, E. Akın, I. Önal and A. O. Akyüz, "Treelogy: A Novel Tree Classifier Utilizing Deep and Hand-crafted Representations", *arXiv preprint arXiv:1701.08291*, 2017.

[3] P. Belhumeur, D. Chen, S. Feiner, D. Jacobs, W. Kress, H. Ling, I. Lopez, R. Ramamoorthi, S. Sheorey, S. White and L. Zhang, "Searching the World's Herbaria: A System for Visual Identification of Plant Species", *Lecture Notes in Computer Science*, 2008, pp. 116-129.

[4] P. Novotný and T. Suk, "Leaf recognition of woody species in Central Europe", *Biosystems Engineering*, vol. 115, no. 4, 2013, pp. 444-452.

[5] P. Barré, B. Stöver, K. Müller and V. Steinhage, "LeafNet: A computer vision system for automatic plant species identification", *Ecological Informatics*, vol. 40, 2017, pp. 50-56.

[6] Y. Kawano and K. Yanai, "FoodCam: A real-time food recognition system on a smartphone", *Multimedia Tools and Applications*, vol. 74, no. 14, 2014, pp. 5263-5287.

[7] D. Korotaeva, M.V. Khlopotov, "Feature Engineering for Tree Leaf Classifier", *CEUR Workshop Proceedings, IET - 2017*, Vol. 1975, 2017, pp. 224-235.

[8] Linköpings universitet, Computer Vision Laboratory, Swedish Leaf Dataset, Web: http://www.cvl.isy.liu.se/en/research/datasets/swedish-leaf/.

[9] OpenCV library, Web: https://opencv.org/.

[10] scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation, Scikit-learn.org, Web: http://scikit-learn.org/stable/#.

[11] M. Zdilla, S. Hatfield, K. McLean, L. Cyrus, J. Laslo and H. Lambert, "Circularity, Solidity, Axes of a Best Fit Ellipse, Aspect Ratio, and Roundness of the Foramen Ovale", *Journal of Craniofacial Surgery,* vol. 27, no. 1, 2016, pp. 222-228.

[12] python-telegram-bot, Python-telegram-bot.org, Web: https://python-telegram-bot.org/.

[13] Peak detection in Python, Gist, Web: https://gist.github.com/sixtenbe/1178136.