DOI: 10.58168/MoInSyTe2024_289-296
УДК 004.9

THE DEVELOPMENT OF A WORD RECOGNITION SYSTEM

Joy MD Tanvir Hasan[1], E.A. Anikeev[1]

[1]Voronezh State University of Forestry and Technologies named after G.F. Morozov

Abstract. The development of a word recognition system with Python is examined in this article, with particular attention paid to the process from dataset preparation to model assessment. Utilizing the powerful libraries provided by Python, such as TensorFlow, PyTorch, and OpenCV, the research tackles the problem of text recognition in various media, such as printed and handwritten texts. The approach places significant emphasis on the utilization of Convolutional Neural Networks (CNNs) [1] to extract visual features, as well as Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) [2] networks, for effectively processing the sequential characteristics inherent in textual data. By employing rigorous preprocessing approaches and utilizing established assessment criteria, this study demonstrates the efficacy of machine learning methods in augmenting word recognition skills. By demonstrating the potential of Python in facilitating the integration of technology and human language, this work provides a significant addition to the academic field. The ramifications of the research have substantial importance for advancing data processing and creating accessible technology.

Keywords: Convolutional Neural Networks, Recurrent Neural Network, Long Short-Term Memory, word recognition, Python

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ РАСПОЗНАВАНИЯ СЛОВ

Джой МД Танвир Хасан [1], Е.А. Аникеев[1]

[1]ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова»

Аннотация. В статье рассматривается разработка системы распознавания слов на Python, при этом особое внимание уделяется процессу от подготовки набора данных до оценки модели. Используя мощные библиотеки, предоставляемые Python, такие как TensorFlow, PyTorch и OpenCV, в исследовании рассматривается проблема распознавания текста на различных носителях, таких как печатные и рукописные тексты. В этом подходе значительный акцент делается на использовании сверточных нейронных сетей (CNN) [1] для извлечения визуальных признаков, а также рекуррентных нейронных сетей (RNN), в частности сетей с длительной кратковременной памятью (LSTM) [2], для эффективной обработки последовательных символов, присущих текстовым данным. Используя строгие подходы к предварительной обработке и установленные критерии оценки, это исследование демонстрирует эффективность методов машинного обучения в совершенствовании навыков распознавания слов. Демонстрируя потенциал Python в области интеграции технологий и человеческого языка, эта работа представляет собой значительное дополнение к научной деятельности. Результаты исследования имеют существенное значение для совершенствования обработки данных и создания доступных технологий.

Ключевые слова: Сверточные нейронные сети, рекуррентная нейронная сеть, Долговременная кратковременная память, распознавание слов, Python.

With advancements in machine learning and artificial intelligence, it is becoming feasible to establish a connection between the content of our papers and the information stored in our files. Word recognition is a crucial component of this bridge that serves a purpose beyond mere book scanning. With this technology, robots can learn and use human words. The difficulties of creating a word recognition system with the Python computer language are looked at in this piece. Python is a widely used programming language renowned for its user-friendly interface and robust libraries, which are recognized for promoting innovative concepts in the fields of data science and machine learning.

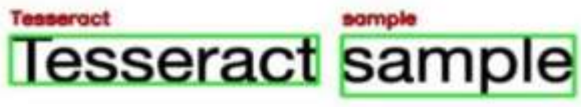

Figure 1 – Example Image for Word Recognition

In the beginning, there was an effort to convert handwritten or typed documents into a form that computers could read and understand. Word recognition started. Data input automation, record security, real-time translation, and text-to-speech systems for the visually impaired all require this position. A multitude of display options, including various types of handwriting, fonts, sizes, positions, and backdrops, contribute to the issue. It becomes more difficult to find the right words when all of these factors combine. Due of its extensive library and tool support, Python makes an excellent protagonist. Notable examples are TensorFlow, PyTorch, Keras, and OpenCV. No matter your level of expertise, these tools will make short work of these difficulties. How to use Python to create a word recognition system is the subject of this article. Some of the most crucial processes are data preparation, model selection, model training, and system success testing. We are recognizing the enormous impact of word-reading technology by doing this. In addition to improving computers' efficiency and usefulness, improving computers' understanding of human language opens the door to innovative uses of technology that capture people's attention. A word recognition system's improvement goes beyond only its technical features. This occurrence shows that we are getting closer to a future where technology and human connections are more integrated. Every machine learning model is constructed on top of a sample that has been meticulously prepared. In the context of word recognition tasks, files typically consist of images or scanned documents containing textual content, along with corresponding transcriptions. The selection of appropriate information holds significant importance. It is imperative for the model to accurately represent the diverse range of text encountered in real-world scenarios, encompassing variances in handwriting style, typefaces, text arrangement, and the intricacy of the backdrop. The IAM Handwriting Database [4] and the COCO-Text dataset [5] are crucial datasets for these activities. The IAM Handwriting Database is utilized for handwriting recognition, while the COCO-Text dataset is employed for text detection in nature photographs.

Preprocessing prepares raw data for model training and assessment, which is vital to word recognition system development. This phase aims to normalize input data to improve the model's capacity to learn from various text pictures. This article covers each preprocessing stage, emphasizing its value and approach. Normalization is an important step that makes sure all the pictures in a file are the same size and scale. It is very important to do this so that the model can handle the data correctly. At this step, the pictures are changed to the same size so that there aren't any differences in size that could make learning unfair or ineffective. For instance, larger images might have a greater impact on the learning process, or a model might struggle to correctly identify small text. Normalization is the process of changing the pixel values of pictures so that they all have the same size, which is usually from 0 to 1. By making sure that gradients are calculated more consistently during training, normalizing pixel values speeds up the model's convergence [6]. The process of grayscale conversion is employed to decrease the number of color channels from three (RGB) to one, to streamline the input for the model. The decrease in color information is substantial due to the redundancy of color information in text recognition tasks, which prioritize the shape and structure of letters rather than their color. Using grayscale conversion has the benefit of making computations simpler. This lets the model focus on finding relevant textual attributes, which greatly improves its training effectiveness [7]. Denoising removes background noise from images so models don't have to deal with it. Shadows, paper texture, and scan defects are among the numerous noise-causing elements. Gaussian blurring and median filtering smooth images, making text easier to read. When dealing with datasets of natural pictures with background noise, these procedures are crucial [7]. Thresholding increases text-background contrast. So, the text sticks out more. To convert a grayscale image to binary, pixels must be identified as black or white using a cutoff value. Otsu's thresholding [8] makes text-background separation easier by employing computer methods to establish the optimal threshold setting. A binary format improves the picture by focusing the model on textual structure rather than color or brightness [7]. Histogram equalization can be used to modify the contrast of photographs. The photos' contrast is improved with the equalizeHist() function. The design of the neural network is crucial, necessitating a harmonious equilibrium between intricacy and effectiveness. The initial layers of CNNs are responsible for feature extraction, whereas the filters that are included in these levels are responsible for independently identifying edges, shapes, and textures. The design of these layers, which includes aspects like number, size, and stride, has a significant impact on the degree to which the network is sensitive to textual components [1].

In the realm of sequential data processing, RNNs and their more advanced equivalents, LSTMs, perform exceptionally well. Their capacity to parse lists of text or properties in a certain order is one of their capabilities. Because it helps computers comprehend the sequence of letters and the semantic meaning of words contained inside lines, proficiency in this ability is especially helpful for word recognition [2]. During the implementation phase, the conceptual design is transformed into a functional model using programming techniques. Python modules like Pandas, which is used for data administration, and Matplotlib, which is used for visualization, are crucial for the early investigation of datasets. Both OpenCV and PIL are powerful libraries that may be used for image loading and preprocessing tasks when dealing with picture data. To properly apply pretreatment operations, it is vital to have a complete understanding of the techniques involved in picture transformation. OpenCV and other libraries have comprehensive capabilities for transforming images, which include scaling, filtering, and thresholding, among other possible transformations. NumPy, on the other hand, may be utilized for operations that are effective when applied to image arrays. PyTorch and TensorFlow are the two libraries that are considered to be the most effective models for neural networks. These frameworks enable the stacking of CNN and RNN layers, which enables the building of complicated model architectures. They do this by providing an intuitive application programming interface (API). The Keras application programming interface (API) of TensorFlow, for example, makes the creation of each layer, activation function, and connections between layers simpler [9]. The process of training a model requires a significant amount of computer power and involves adjusting the weights of the network depending on the disparity between its predictions and the actual outputs. To complete this procedure, you will need to choose an appropriate loss function (for example, categorical cross-entropy for multi-class classification) and an optimizer (for example, Adam or SGD). When it comes to monitoring the training process, many tools, such as TensorFlow's TensorBoard, might prove to be quite useful. These tools allow for the visualization of parameters such as loss and accuracy.

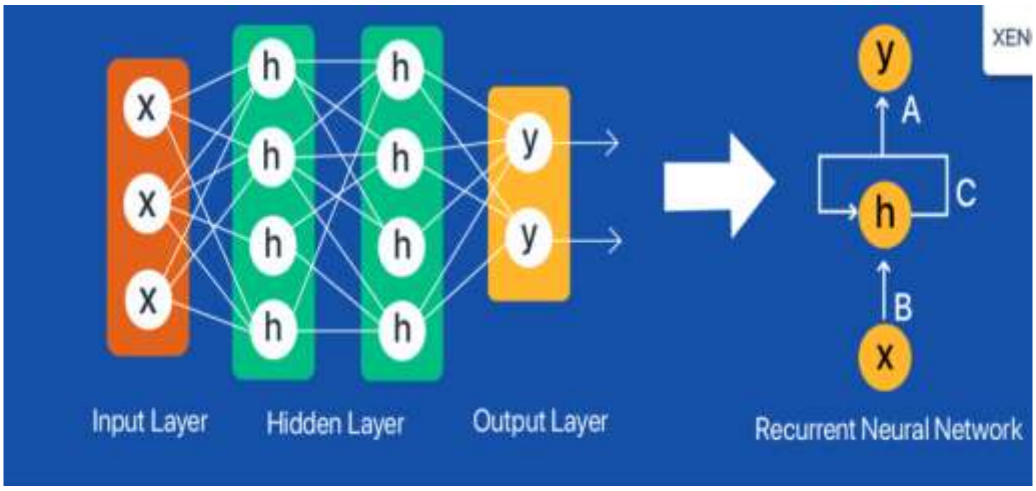

Figure 2 – CNN for Text Recognition [10]

A comprehensive evaluation approach includes both quantitative metrics and qualitative assessments to guarantee the model's effectiveness across diverse circumstances. The concept of accuracy, although serving as a broad measure of performance, can be deceptive, particularly when dealing with imbalanced datasets. These measures offer a more detailed perspective on the performance of the model, emphasizing its capacity to accurately detect and avoid overlooking pertinent events. Provides a valuable understanding of the model's mistake patterns, which is essential for creating incremental enhancements.

Conclusion

The process of developing a word recognition system using Python demonstrates how creativity and technology can work together. This piece talked about the many steps that go into making a system. Art and science have both had an impact on the creation of datasets, which we use to improve our raw materials and build neural network models for word recognition. Python's machine learning catalytic function is shown by using theory models in code. By training these models, we saw that machine learning needs to be evaluated, learned, and fixed. Evaluation tools show both success and potential problems. The high marks for accuracy, precision, recall, and F1 in our technology and word recognition systems show that they can make apps better. Both machine learning and AI are always changing. The new goals include making the model smarter about writing styles, language nuances, and the bigger picture. When word recognition algorithms are built into more complex systems, they open up new ways to learn and come up with ideas. The goal is to create a word recognition system that blends human knowledge with machine understanding in a way that works well. Improvements made to these systems make it easier for technology and natural language processing to come together. This will make it easier for people to share new information and knowledge and communicate clearly. Python, along with researchers and writers, is making great strides in the area of word recognition, which helps people understand written words in a variety of forms.

References

1. "What are Convolutional Neural Networks? | IBM." Available: https://www.ibm.com/topics/convolutional-neural-networks

2. "Understanding LSTM Networks -- colah's blog." Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

3. M. Bisht and R. Gupta, "Offline Handwritten Devanagari Word Recognition Using CNN-RNN-CTC," SN Computer Science, Dec. 13, 2022. Available: https://link.springer.com/article/10.1007/s42979-022-01461-x

4. "Research Group on Computer Vision and Artificial Intelligence – Computer Vision and Artificial Intelligence." Available: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database

5. "COCO-Text V2.0." Available: https://bgshih.github.io/cocotext/

6. S. Jaiswal, "What is Normalization in Machine Learning? A Comprehensive Guide to Data Rescaling," Jan. 04, 2024. Available: https://www.datacamp.com/tutorial/normalization-in-machine-learning

7. M. Patel, "The Complete Guide to Image Preprocessing Techniques in Python," Medium, Oct. 23, 2023. Available: https://medium.com/@maahip1304/the-complete-guide-to-image-preprocessing-techniques-in-python-dca30804550c

8. A. Murzova and A. Murzova, “Otsu’s Thresholding Technique | LearnOpenCV,” LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow with code, & tutorials, May 05, 2021. Available: https://learnopencv.com/otsu-thresholding-with-opencv/.

9. H. Scheidl, “Build a Handwritten Text Recognition System using TensorFlow,” *Medium*, May 22, 2023. Available: https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5

10. N. S. Gill, “Convolutional Recurrent Neural Network For Text Recognition,” *XenonStack*, May 03, 2023. Available: https://www.xenonstack.com/insights/crnn-for-text-recognition

Список литературы

1. “What are Convolutional Neural Networks? | IBM.” Available: https://www.ibm.com/topics/convolutional-neural-networks

2. “Understanding LSTM Networks -- colah’s blog.”. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

3. M. Bisht and R. Gupta, “Offline Handwritten Devanagari Word Recognition Using CNN-RNN-CTC,” SN Computer Science, Dec. 13, 2022. Available: https://link.springer.com/article/10.1007/s42979-022-01461-x.

4. “Research Group on Computer Vision and Artificial Intelligence — Computer Vision and Artificial Intelligence.” Available: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database

5. “COCO-Text V2.0.” Available: https://bgshih.github.io/cocotext/

6. S. Jaiswal, “What is Normalization in Machine Learning? A Comprehensive Guide to Data Rescaling,” Jan. 04, 2024. Available: https://www.datacamp.com/tutorial/normalization-in-machine-learning

7. M. Patel, “The Complete Guide to Image Preprocessing Techniques in Python,” Medium, Oct. 23, 2023. Available: https://medium.com/@maahip1304/the-complete-guide-to-image-preprocessing-techniques-in-python-dca30804550c

8. A. Murzova and A. Murzova, “Otsu’s Thresholding Technique | LearnOpenCV,” LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow with code, & tutorials, May 05, 2021. Available: https://learnopencv.com/otsu-thresholding-with-opencv/.

9. H. Scheidl, “Build a Handwritten Text Recognition System using TensorFlow,” *Medium*, May 22, 2023. [Online]. Available: https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5

10. N. S. Gill, “Convolutional Recurrent Neural Network For Text Recognition,” *XenonStack*, May 03, 2023. Available: https://www.xenonstack.com/insights/crnn-for-text-recognition.