DESIGN AND IMPLEMENTATION OF A COGNITIVE CLOUD ASSISTANT: A TEXT CLASSIFICATION MODEL APPROACH

Dávid VAĽKO[*], Jakub KIPIKAŠA[**], Norbert ÁDÁM[*], Heidar KHORSHIDIYEH[*]
[*]Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic, Tel. +421 55 602 2839
E-mails: david.valko@tuke.sk, norbert.adam@tuke.sk, heidar.khorshidiyeh@tuke.sk
[**]IBM Slovakia, Aupark Tower, Protifašistických bojovníkov 11, 040 01 Košice, Slovak Republic
E-mail: jakub.kipikasa@ibm.com

ABSTRACT

The main objective of this research was to develop a classification model for the design of a cloud cognitive assistant. The classification model should be able to classify the textual description of the cloud architecture into desired multiple classes. For the purpose of implementation of such assistant, the analysis of current state of cognitive assistants and cloud computing was researched. As for the implementation of classifier, the overview of the possible ways how to create such classifier was also added to the research. Based on the analysis, a solution was proposed for the implementation of a model for text classification, and also a proposal for the implementation of an assistant that would use the proposed model. The Keras library was used to create a sequential text classifier. The IBM Watson cloud services were used to deploy the created model into live environment, and the services from the same group were used to develop a proposed cognitive assistant that was in the end connected to the classifier model.

Keywords: *Classification, Cloud Computing, IBM Watson, Neural Networks*

1. INTRODUCTION

The motivation for this research paper is to design and develop a classification model for a cognitive cloud assistant, and then implement the basic solution on a representative sample of data. The cognitive cloud assistant is a solution based on a set of methodologies, using artificial intelligence and machine learning, designed for the selection of the targeted system architecture in a cloud environment.

The first step in the implementation of the classification model will be to define the format of the reference dataset that will be used to train the classifier. The reference data of cloud architectures should be collected using reference documentation from different cloud providers.

Once the reference data is obtained, it will be possible to implement a classification model, which based on a textual description of the architecture will be able to classify it into one or more result categories. When implementing a text classification solution, it will be necessary to consider natural language processing and classification into multiple classes.

One option for implementing a cognitive assistant is to use existing services, such as the IBM Watson family of services, to implement a working prototype of a cognitive cloud assistant.

The research paper deals with the design of the classification model, the design of the reference dataset, as well as the design of the cloud environment for the deployment of the model and the implementation of the cognitive assistant.

2. CURRENT STATUS OF COGNITIVE ASSISTANTS

In recent years, cognitive assistants have emerged, leveraging artificial intelligence and natural language processing technologies to enhance the interaction between humans and computers. These intelligent personal assistants, also known as virtual assistants, are increasingly entering our daily lives and revolutionizing the way we access information, perform tasks, and communicate with technologies. Cognitive assistants have gained remarkable popularity and have become an integral part of the digital ecosystem.

The current state of cognitive assistants demonstrates remarkable progress in their capabilities and functions. These AI-powered assistants are designed to understand human queries and respond to them, perform tasks, and provide recommendations based on user preferences.

One significant aspect of the current state of assistants is their ability to provide personalized experiences. By utilizing machine learning algorithms and data analysis, they can learn from user interactions, their preferences, and historical data to provide tailored responses and recommendations.

Cognitive assistants have found their application in various fields and industries. In health care, they assist healthcare professionals in diagnosing diseases, exploring treatment options, and providing patient care support [1]. Besides healthcare, they have also found application in the financial [2] and legal sectors [3] [4], as well as in customer support.

To understand the current state of cognitive assistants, it is necessary to first divide them. The most basic division will be according to the use of the assistant, in this case, they can be divided into general and domain-specific cognitive assistants.

General cognitive assistants, also referred to as Intelligent Personal Assistants (IPA), according to Benjamin R. Cowan et al. [5], are intelligent assistants that utilize input data recognition such as voice or text, and in some cases contextual information, to provide responses in natural language or perform requested tasks.

In contrast to general cognitive assistants, domain-specific cognitive assistants are designed to perform tasks of a particular type. It is precisely the domain

specificity that has the greatest impact in areas where specialized knowledge, expertise, and decision-making play a key role.

3. CURRENT STATE OF CLOUD COMPUTING

Authors of the paper [6] described public cloud services offered by global giants such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform. They have gained immense popularity due to their scalability, cost-effectiveness, and extensive service offerings. On the other hand, private clouds provide dedicated resources to a single organization offer increased security and control. Hybrid and multicloud environments combine the benefits of both public and private clouds, enabling organizations to leverage the advantages of different cloud models while ensuring data portability and interoperability. The definition of cloud services described in article [7] defines three main service categories:

- Software as a Service
- Platform as a Service
- Infrastructure as a Service

In addition to classification based on the type of services provided, clouds can be categorized based on access approach. Currently, these services are divided into another three groups:

- Public
- Private
- Hybrid cloud services

With the emergence of public, private, and hybrid cloud services, organizations are empowered with unprecedented scalability, flexibility, and cost-effectiveness.

4. IMPLEMENTATION OF CLASSIFICATION MODEL

There are a number of ways in which a neural network model can be constructed for text classification. A big factor is the data and its format on which the model will be trained on.

4.1. Defining model layers

In order to work with text-type data, it is necessary for text to be transformed into so-called text vectors. This process can be done independently, without the need to insert a layer directly into the neural network model, or directly in the model using a dedicated layer. This layer was independently adapted on the architectures descriptions from the dataset created. In this case, the vectorization of the input data is taken care of by the vectorization layer, which is defined second in order. Afterwards comes the Input layer, which defines the format of the input data to the model. Subsequently, the Embedding layer, which can capture semantic and contextual meaning in text vectors was used. The usage of Embedding layers presents a very popular practice in Natural Language Processing (NLP) problem solving.

The following layers, except for the last one, can be defined almost without restrictions, only the consistency of the data must be preserved. These layers define how the network solves the problem for which it is being developed. In this case, it is the problem of text classification, which is usually solved by recurrent neural networks (RNNs).

RNNs are used because of their ability to capture sequential dependencies and contextual information within a sequence of data [8]. A special category of RNNs are LSTM (Long short-term memory) networks, which have long short-term memory, and can learn more complex and longer-term sequences of input data [8]. This functionality of RNNs and LSTMs was the key in selecting the appropriate layers for building the actual neural network model.

The GlobalMaxPool1D layer was chosen as the next layer to reduce the dimensionality of the output data from the previous LSTM layer. By this step, the overtraining of the neural network with fully connected layers was prevented.

Subsequently, a combination of Dropout and Dense layers was used. The Dropout layer randomly deactivates a portion of neurons during training, forcing the network to learn general representation of the data.

The last layer is very important, as on this layer, the resulting parameter for classification depends. In the implementation of the network, a Dense layer was used as the last layer, which is connected to all the neurons of the previous layer, and based on the activation function, it performs the computation of the input data provided by the previous layer. The dimension of this last layer is determined by the number of classes that can be classified.

In addition to the correct number of neurons in the last layer, it was also necessary to define the loss function, which is used to calculate the error rate of the model. This loss function was defined directly when compiling the assembled sequential model. When building the model, binary cross entropy was used as the loss function, as there are multiple classes that need to be classified to categorize the input requirement.

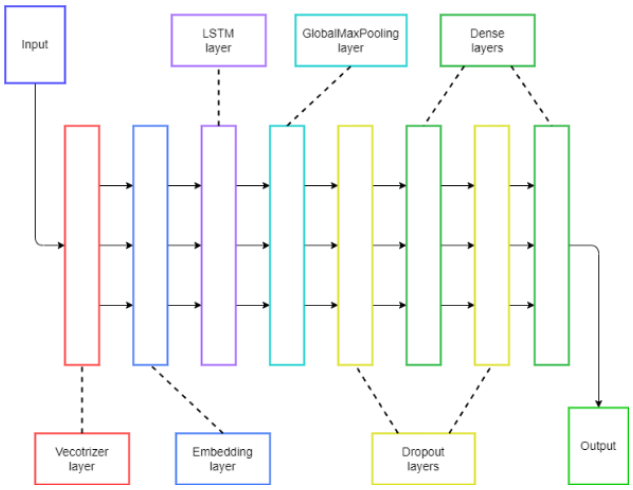

Fig. 1 Sequential neural network

In Fig. 1 we can see a graphical representation of the layers of the constructed neural network. This

representation also shows the output from the last Dense layer, which is in the form of a one-dimensional vector containing the individual probabilities for each predicted category.

4.2. Custom Thresholding layer

The previous section of this chapter discusses that the output of a classification model presents a series of probabilities in the form of a one-dimensional vector. These values are represented by a decimal number ranging from zero to one. Since this numerical probability is not very informative for the user who has made an input request to the classifier, it is necessary to transform these values into their textual form. In addition to the transformation itself, it is essential to set a threshold value that will define the acceptability of the prediction.

Meeting these requirements for back-transformation of the resulting percentage probabilities and exclusion of non-matching parameters was achieved by implementing a custom layer that is a descendant of the generic Layer class of the Keras library [9].

The ThresholdingLayer() class defines a layer of the sequential model that accepts a one dimensional vector from the previous layer, applies thresholding values to that vector, keeping only the values that are acceptable. After applying the threshold to the prediction vector, the class applies the percentage probability transformation to the real naming of the predicted classes. To create an instance of this class, it needs to be given a threshold value, and also a list of all the classified categories.

After the application of ThresholdingLayer() to the created neural network, the model can be utilized with various IBM Cloud Watson services, without any need to modify the output data, as the desired classes are already returned as output from the model itself.

Fig. 2 shows the sequence of steps from the input request, to the final output of the predicted categories.

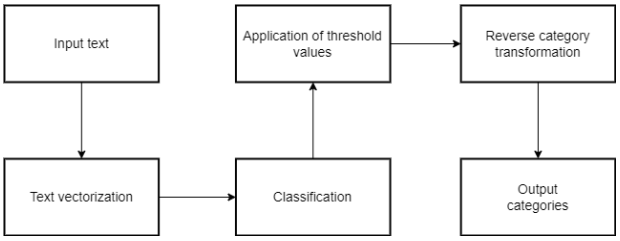

Fig. 2 Classification Sequence Diagram

5. MODEL VALIDATION

In the field of machine learning, the effectiveness of classification models depends heavily on the quality and quantity of the trained data. The dataset required for the specific functionality of this classification model was created manually, using reference architectures documented directly by specific cloud providers. The data extracted from these documentations was often specific, and utilized mostly one or more major classes to which the architectures belong. As a consequence, the resulting dataset could appear unbalanced, and even after applying data augmentation techniques to increase the number of data, the individual classes were uneven.

This problem was solved by validating the created model on an external dataset, which had to be usable for multi-class classification. The final data for validation was a dataset containing textual descriptions of scientific articles, and multiple classes to which these articles could belong. This data representation matched the dataset representation of the architectures, which also consisted of textual descriptions, and the classes belonging to each solution.

With a larger dataset, the evaluation process becomes more reliable, allowing for a more comprehensive analysis of the model's predictive capabilities in different scenarios. This approach not only provides insight into the effectiveness of the model, but also increases confidence in its deployment in real-world applications.

In Fig. 3 we can see the so-called classification report, which was made on a test sample of data that constituted twenty percent of the total dataset. As already mentioned, this validation data contained the textual descriptions of the scientific articles, and the classes to which they belonged. From this report, it is known that the model was able to classify almost all classes uniformly. Only a single class, Quantitative Finance, failed due to insufficient data sampling, with only sixty-seven out of the total. The Quantitative Biology class from this dataset also fell behind the other classes due to a low representative data sample.

Class	Accuracy	Recall	F1	Count
Computer Science	0.87	0.92	0.89	2535
Physics	0.94	0.92	0.93	1849
Mathematics	0.93	0.84	0.88	1711
Statistics	0.84	0.86	0.85	1554
Quantitative Biology	0.79	0.60	0.68	194
Quantitative Finance	0.00	0.00	0.00	67

Fig. 3 Classifier metrics for validation dataset

Evaluation of loss function and accuracy was done using the validation sample, which was set to twenty percent of the total dataset size. Fig. 4 and Fig. 5 shows these values. As we can see in these figures, the loss function reached a minimum in the fourth epoch of the validation process, while the accuracy reached its highest value in the second epoch of the process.

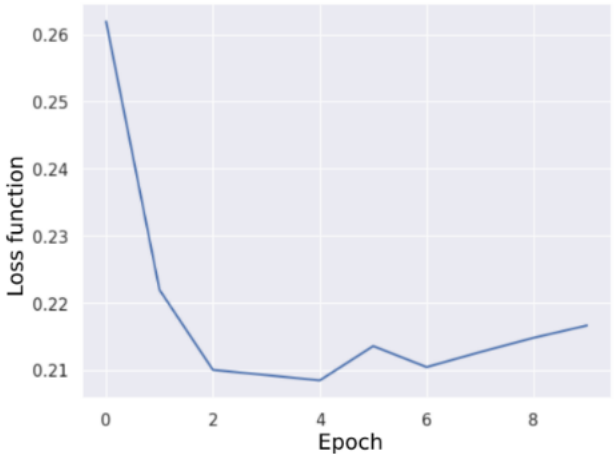

Fig. 4 Validation sample accuracy

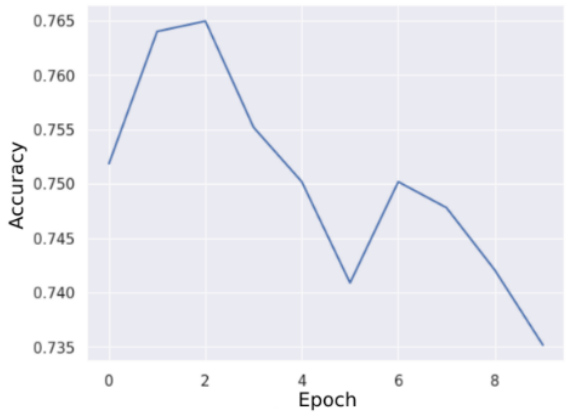

Fig. 5 Validation sample loss function

6. COGNITIVE CLOUD ASSISTNAT IMPLEMENTATION

The primary goal in designing the implementation of a basic cognitive assistant solution was to select services and a provider that offers the right combination of required criteria.

6.1. Solution proposal for assistant implementation

IBM Cloud was chosen as the provider for this solution because of its good offer of tools and products for working with machine learning. It is easily scalable, secured and also possesses the possibility of internal consultation with IBM Cloud support. The products for working with machine learning tools are directly integrated with Watson, which is IBM's AI platform.

This cloud service provider offers many services for working with Wattson AI in a so-called lightweight, free version with specified limits. These limits should not be restrictive in the design and subsequent testing of this implementation. It is important to note that the Watson Assistant service should be accessible via the Application Programming Interface (API).

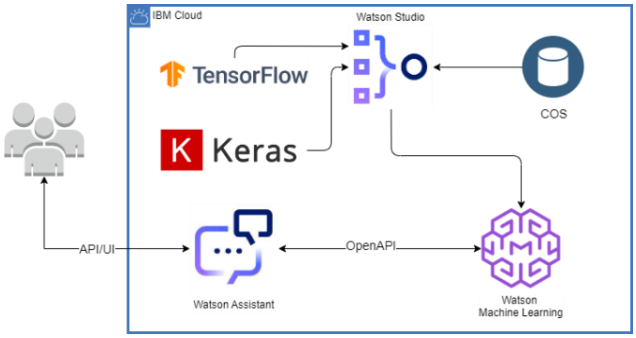

Fig. 6 Cloud assistant architecture

Fig. 6 describes the cloud services, that could be used within the implementation of the cognitive assistant. The main group of components from IBM Cloud Watson family are used to successfully prepare, deploy, and integrate the classifier model. The Cloud Object Storage is used to store required datasets for model learning, and for storing the source codes of implementation, that are then deployed to real world environment via the Watson Machine Learning.

The Watson Studio is required for the final setup of the deployable model, to adjust the connection to the Cloud object storage (COS) instance. Lastly the Watson Assistant was used to develop simple dialogue flow, that was in the end integrated with the deployed model via the OpenAPI [10] specification.

6.2. Model deployment

The process of deploying a classification model can be done in a number of ways, in this case the deployment of the model was achieved by defining a deployable function that included the definition of the model and its components. The use of function deployment was necessary due to the limitations of the Watson Machine Learning API functionality, which specifies a well-defined input in JSON format for which the model was not ready. In addition to the strict specification of the input data, the service requires specified output data format. The processing of the input request, the execution of the prediction, and the definition of the output of the deployed function is handled by the additional function score(), which is then executed after API call to the predictions endpoint. The score() function is essential for model implementation using the deployable function.

6.3. The preparation of dialogues

In the application window of the Watson Assistant service it is possible to define different actions to which the cognitive assistant should respond. As this is a basic implementation of the assistant, these actions do not need to be particularly deep, a simple recognition of the individual requirements on the desired cloud architecture or the specification of the provider of the requested services should be sufficient.

The first important part of the dialogue, is the initial request, which should trigger the communication flow. As the intention of the assistant is to help with the cloud architectures, messages such as ,,Help me with the cloud architecture" should be good enough. The advantage of Watson Assistant service is that it uses our defined requests and responses to learn its NLP model, to better recognize user input, and possibly skip some following actions if the request already included the desired information for the next dialogue step.

The following actions included general information gathering, as the desired provider of cloud architecture, hardware and storage specifications, and in the end, the actual description of the architecture.

6.4. Watson service interconnection

As described inside the service documentation, the Watson Assistant can be integrated with various external services via the extensions. The usage of these extension services is embedded directly in the dialog action definitions, just as a consequence triggered by different requests. The user can send a request to the cognitive assistant, to which this action defined as an extension is invoked.

Before these extensions can be used in dialogue sequences, the necessary extensions need to be defined first. The creation of this extension consists of linking the

Watson Assistant service with another external service, using a definition file in JSON format, which follows the OpenAPI standard. This definition file contains a description of the authentication to the external service and the endpoints that the extension will subsequently use.

In this case, the OpenAPI JSON included authentication definition for API key, deployment ID and version, the basic data format, and the definition of the prediction endpoint, at which the deployment is listening. For security reasons, the API key, model ID and version were defined as variables, which were then requested inside the dialogue action, when extension was integrated to the flow.

7. DISCUSION

The proposed classification model for the cognitive cloud assistant demonstrates significant practical implications by enhancing the efficiency and accuracy of cloud architecture selection. Its integration with IBM Watson services facilitates scalable and adaptable deployment in diverse cloud environments. However, potential limitations include the model's dependence on high-quality, balanced datasets, which may not always be available in real-world scenarios. Additionally, the model's performance could be hindered by evolving cloud technologies and architectures that require continuous updates and retraining. Ensuring data privacy and security while processing sensitive information is also a critical challenge.

8. CONCLUSION

The main goal of this research was to design and develop a classification model for a cognitive cloud assistant, and then implement the underlying solution on a representative sample of data. The resulting classifier solution, based on the input textual data, can provide a set of service categories that are required for the execution of the final system architecture.

A secondary goal was to produce a simple concept of a cognitive assistant that would integrate the implemented classification model, and use it in a way to evaluate and classify the textual input descriptions of the different architectures.

The process of implementing the classification model consisted in defining the targeted architecture of the cognitive assistant system into which it was subsequently integrated. Once the architecture and data flows were designed, a reference dataset of different cloud architectures was defined with which the final model was trained. Subsequently, the implementation process of the classifier was described in detail, which was built using machine learning libraries. After the successful implementation of the model, a cloud environment was created in which several services were made available to enable the deployment of the cognitive assistant service, and its interconnection with the created classifier.

The main contributions of this research are the exploration of a possible solution for the implementation of a classification model needed for specific use cases, as well as the broadening of the horizons in cloud systems, and their utilization in the creation of the concept of a cloud cognitive assistant.

REFERENCES

[1] COSTA, A. – NOVAIS, P. – JULIAN, V.: A Survey of Cognitive Assistants. Cham: Springer International Publishing, 2018, pp. 3–16. [Online]. Available: https://doi.org/10.1007/978-3-319-62530-0 1

[2] KLIMAN, R. – ARINZE, B.: “Cognitive computing: Impacts on financial advice in wealth management,” Aligning Business Strategies and Analytics: Bridging Between Theory and Practice, pp. 11–23, 2019.

[3] NUNEZ, C.: “Artificial intelligence and legal ethics: Whether ai lawyers can make ethical decisions,” Tul. J. Tech. & Intell. Prop., vol. 20, p. 189, 2017.

[4] JOSHI, K. P. – GUPTA, A. – MITTAL, S. – PEARCE, C. – FININ, T. et al.: “Alda: Cognitive assistant for legal document analytics,” in 2016 AAAI Fall Symposium Series, 2016.

[5] COWAN, B. R. – PANTIDI, N. – COYLE, D. – MORRISSEY, K. – CLARKE, P. – AL-SHEHRI, S.– EARLEY, D. – BANDEIRA, N.: “what can i help you with?”: Infrequent users’ experiences of intelligent personal assistants,” in Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services, ser. MobileHCI ’17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3098279.3098539

[6] IBM. What is hybrid cloud? [Online]. Available: https://www.ibm.com/topics/hybrid-cloud

[7] P. SRIVASTAVA – R. KHAN, “A review paper on cloud computing,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 8, no. 6, pp. 17–20, 2018.

[8] YU, Y. – SI, X. – HU, C. – ZHANG, J.: (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Computation, 31, 1235-1270.

[9] CHOLLET, F.: (2018). Keras: The Python Deep Learning library.

[10] APOSTOLAKIS, I. – MAINAS, N. – PETRAKIS, E.: (2023). Simple querying service for OpenAPI descriptions with semantic extensions. Information Systems, 117, 102241.

Received June 24, 2024, accepted July 8, 2024

BIOGRAPHIES

Dávid Vaľko (Ing. Bc. PhD. student) completed his master’s degree in Computer science in 2021 at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. Since 2021, he has been a PhD. student at the Technical university in Košice. His scientific research focuses on the use of computer science in medicine, particularly the classification and processing of

medical signals such as ECG. He also received a bachelor's degree in Urgent medical care in 2023.

Jakub Kipikaša (Ing.) completed his master's degree in Computer science in 2024 at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. Since 2022, he has been working at IBM, mainly with cloud services, which led to the topic of his scientific research. His scientific research focused on the practical use of cloud technologies and machine learning for cognitive assistant development.

Norbert Ádám (Assoc. Prof.) in 1998 graduated (MSc.) summa cum laude at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. He defended his PhD. in the field of Computers and computer systems in 2007; his thesis title was "Contribution to simulation of feed-forward neural networks on parallel computer architectures". Since 2018 he is working as an associate professor at the Department of Computers and Informatics. Since 2008 he is the head of the Computer Architectures and Security Laboratory at the Department of Computers and Informatics. His scientific research is focusing on the parallel computer architectures.

Heidar Khorshidiyeh (Ing., PhD. Student) holds Iranian nationality. In 2019, he graduated with a Bachelor's degree (Bc.) from the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. In 2021, he earned his Master's degree (Ing.) from the same department and faculty. Since 2021, he is PhD student at the Technical University of Košice. His scientific research is centered on computer graphics, specifically on the representation of 3D scenes using hierarchical data structures.