Continuous natural language processing pipeline strategy

István Pölöskei *
*adesso Hungary Kft.
Infopark sétány 1, 1117 Budapest, Hungary
E-mail: istvan.poeloeskei@adesso.eu

2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI) | 978-1-7281-9544-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/SACI51354.2021.9465571

***Abstract*—Natural language processing (NLP) is a division of artificial intelligence. The constructed model's quality is entirely reliant on the training dataset's quality. A data streaming pipeline is an adhesive application, completing a managed connection from data sources to machine learning methods. The recommended NLP pipeline composition has well-defined procedures. The implemented message broker design is a usual apparatus for delivering events. It makes it achievable to construct a robust training dataset for machine learning use-case and serve the model's input. The reconstructed dataset is a valid input for the machine learning processes. Based on the data pipeline's product, the model recreation and redeployment can be scheduled automatically.**

Keywords— NLP, pipeline, deep learning, machine learning, Data

I. Introduction

Internet of Things, Big data, machine learning, Business Intelligence. All of them are buzzwords associated with the latest information technology trends. They have committed the stream data processing methods and the construction of streaming data solutions. Natural language processing is a branch of artificial intelligence (AI) as a set of various procedures for accumulating information from incoming text channels [1].

The recent advancements in information technology, coupling the streaming technologies and text mining, present a genuine business meaning. Deep learning has been revolutionarily transformed language processing recently, resolving practical obstacles [2]. The field mentioned above demands exceptional data processing measures because the machine learning algorithms necessitate creating a numerical dataset. A data pipeline is a must-have in that domain, but the literature has not focused on that topic.

The examined pipeline strategy builds the association between data sources and machine learning methods based on has distinct procedures [3]. The accomplished scheme is a convention for delivering events, providing an appropriate input for machine learning use-case.

II. Methods

A. Artificial Intelligence

Artificial intelligence (AI) is to make auspicious decisions based on the existing data without realizing the decision-making process's comprehensive precepts [4]. The training from data can be supervised and unsupervised learning. The character of learning fully depends on the quality of the training dataset. If the data source is poor, then the generated model becomes biased [5].

Since the data is decisive in machine learning, the data producer architecture is also critical that fulfill the algorithms with correct input data. From this point, data pipelines act as an influential role in the artificial intelligence ecosystem.

1) Natural language processing

Natural language processing is an area for inventing computation systems for understanding the human language [1]. The texts consist of sentences and words. A document could be summarized, interpreted, extracted, or actively produced [2].

The conventional naive procedure is the bag of words for information extraction [6]. It calculates the word occurrences. It does not use the position and context of each word. The word embeddings approach comprises the analysis of relationships between words [7]. Similar words are close to each other; the formed vectors are reusable by different use-cases like pre-trained models or word vectors.

Transfer learning is a technique to imitate the human's generalization ability with the deep learning methodology. The trained model can be used in a new topic; the already obtained prior knowledge can speed up the learning in a new context [8].

2) BERT

BERT (bidirectional encoder representations from transformers) is the state-of-the-art NLP tool by Google. It consists of self-attention layers and transformer blocks, uniting various approaches. It requires considerable computation power, but it is broadly applied in the industry because it currently offers top performance based on Google's pre-trained model [9].

3) Tensorflow and Keras

State-of-the-art technology for contriving neural networks in several domains [10]. The operations are executed in graph-like tensors; each node indicates a mathematical operation. The training can be orchestrated on the GPU or CPU. Keras provides a high-level API on top of the TensorFlow for designing deep learning models with more effortless [10].

Tensorflow provides a state-of-the-art pipeline specification (Tensorflow Extended, TFX) [10]. This structure extends its functionality in a workflow management manner (Airflow or Kubeflow) [11]. As separated modules, the preparation actions can be utilized independently but in sequence.

B. Data import

Two main methods are expected for importing data. The batch import deals with processing a giant bundle of files in a batch [12]. The streaming data import determines

continuously providing data-source [12]. This paper focuses on streaming use-case because the popular big data pipelines use mostly the streaming solution and continuous communication. It allows quick reactions to the events and the changing of the business.

1) Message broker architecture

Streaming is fundamentally a flow of data in packages. The data is generated continuously, and it is delivered in small records simultaneously [12], providing a real-time connection and real-time processing.

A message broker can supervise the streams [13]. The source is the producer; a message broker manages all of the incoming data. In a parallelized mode, because the architecture can be partitioned according to sources or topics. The collected data is dispatched to the consumer.

Apache Kafka is a distributed open-source streaming and message broker technology for establishing event-based design [13]. As a de-facto standard, diverse industries use their adaptations as a vital element for asynchronous real-time data pipelines. Kafka is a conventional apparatus for producing and receiving data like a workflow.

The Kafka ecosystem can convey messages according to categories (topic). Message can be in any form; each one may have its metadata (with timestamp and topic). The incoming pushed messages (by the producer) are persisted in the Kafka Broker. The consumer can pull those messages and processing them as a stream of messages. The partitions of this architecture can be assigned topic-wise to the streams (Figure 1).

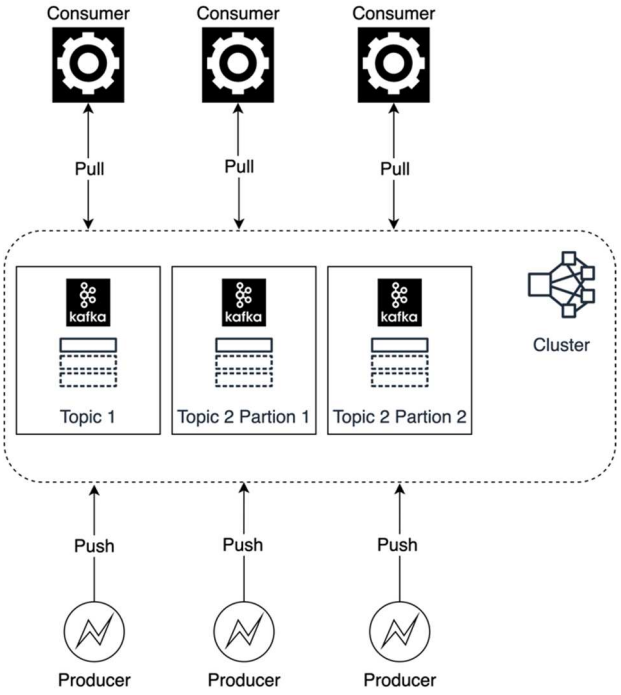

Figure 1. Event Broker architecture

Some mature resolutions have formed on the Kafka, extending its features in the cloud (like Confluent and cloud vendor-specific implementations) [14]. According to availability zones, Confluent's Kubernetes operator can help expose Kafka's features in the cloud-native environment by adopting its traits like load balancing, ingress controller, or assembling the whole cloud landscape.

As a standard event ingestion service, cloud provider-specific implementations are recommended to use in the public cloud (like Azure's Event Hub) [14]. It focuses on data processing instead of designing a new message broker architecture for the current task.

C. Data processing

1) Event processing

Before Event Broker architecture (Kafka), the data digestion resembled processing worker jobs' storage entries [15]. All data is placed in the storage, and the schedule triggers the worker jobs. This approach (Figure 2) might miss the chance of real-time processing.

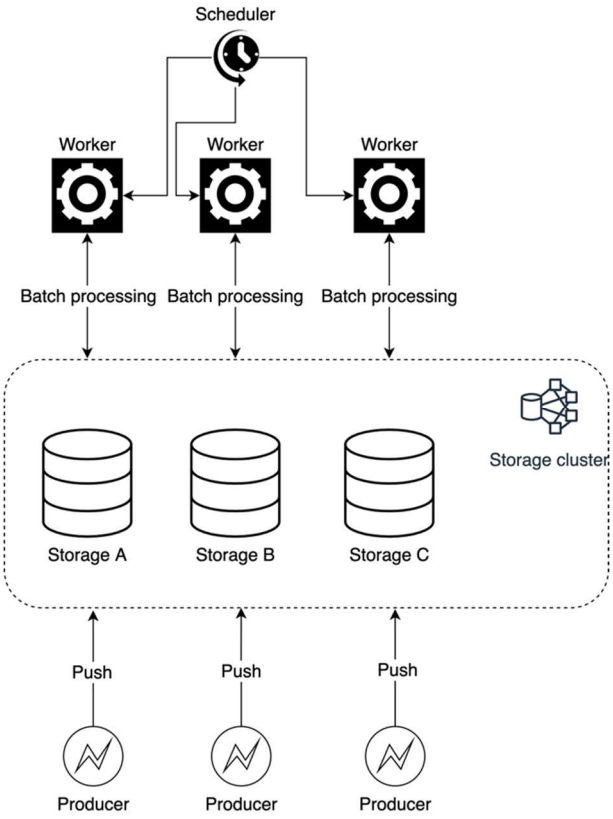

Figure 2. Worker architecture.

The other strategy is based on the events (Figure 3). The data is a stream of events, and it is transmitted instantaneously by the message broker. In the NLP domain, the incoming channel is a text stream. An event broker component forwards and distributes the streamed data source. Additional core functionalities can be attached to the event processing, like monitoring and persistence. It can handle several subscriptions. With an event-based procedure, the pipeline is more responsive and more resource-conscious than in the scheduler use-case.

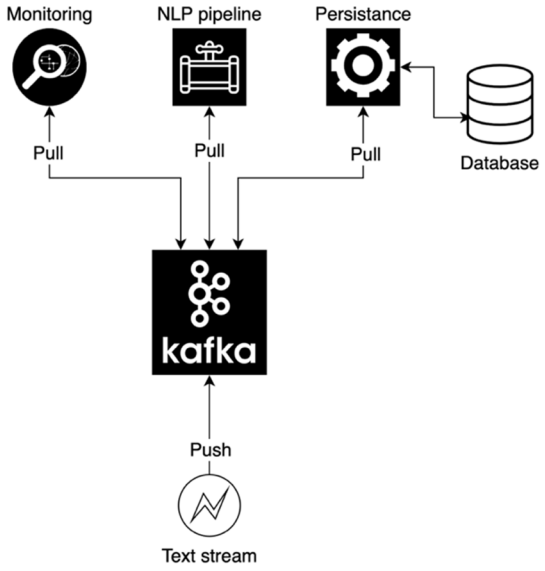

Figure 3. Event broker architecture, NLP scenario.

2) Data pipeline

A data workflow is a glue script, providing a supervised connection between data sources and machine learning systems. It is entirely affected by the data schema. The raw input data is collected and consolidated through a pipeline, assembling a high-level input for the algorithms. Usage of native cloud computing is essential for reaching an optimized performance and expense. Achieving such a pipeline requires some standard operations. The application of a standard framework hides these unnecessary low-level technical details [3].

At the end of the pipeline, a model is generated. The model is servable; a REST-API provider web server can publish it [16].

III. Results

A. Current situation

A multi-national enterprise needs to process its text data produced during the production. The data is released in a streaming manner, continuously. The information system should accumulate it and storing a machine learning modeling on this text dataset. The current machine learning domain is natural language processing.
An NLP pipeline context suggests a unique mixture of the different techniques, like a sequence of the specified transformation steps. It requires pre-processing steps, which must be performed by automatizations. The deployment and retraining of a new model need to be scripted, minimizing the hand-operated steps. The architecture needs to allow using the model on the incoming text channel.

B. Actions taken

Tokenization is the activity for splitting the text into atomic elements (words or tokens). They should be standardized, like eliminating the punctuations. The created tokens must be cleaned by removing the problematic noise (Figure 4). After that, a word incidence can be calculated based on the generated standard tokens (naive approach). This statistic can refer to the text's original meaning, like introducing each word's relative importance [17].

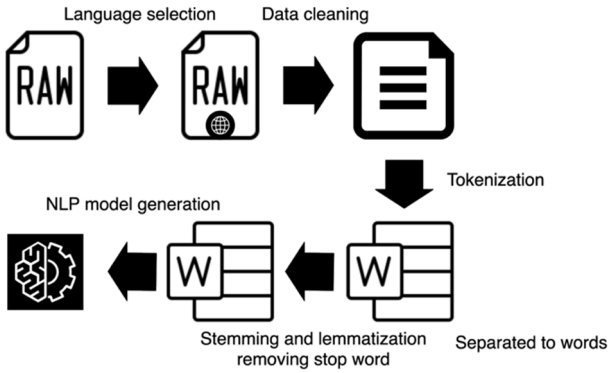

Figure 4. Basic NLP pre-processing.

The processing of the text has distinct actions. The data is plain text—each step is referred to as an annotator.

- Selecting language
- Dropping the missing values, redundancy, and meta-data.
- Tokenization, for transforming the text into an array of single lowercase words.
- Purifying the stop words and punctuation.
- Stemming, retaining the stem form of the world.
- Lemmatizations, creating each word's base synonym.
- Grouping sentences from tokens.
- Part of the Speach (POS) classes are assigned to the tokens, like providing a grammatical categorization.
- Named Entity Recognition, identifying the entities in the text.

The machine learning models do not read the raw text input effectively. The text content should be converted to numerical arrays. For this, the first principal step is tokenization. The standard Tokeniser classes implement the following functionalities: converting words into indexes and removing the unnecessary characters and words, the tokenization can be managed on either character or word base. It adds padding for providing the same dimension of each array. The transformed dataset can be a valid input for various machine learning techniques.

The relevant statistic can be easily generated based on the word count. The dataset is being used for training and testing purposes, granting an evaluation for the model.

The text data has been produced from several sources. An application (like a Spring Boot application) is responsible for publishing the documents (for example, in JSON format) to Message Broker architecture (Kafka), giving a real-time fault-tolerant streaming resolution. The data digest section is qualified to structurize the incoming channel, with primary conversion, for extracting the essential sections. The result is persisted in storage according to the topics (or data-source). The cloud-based modern database provides native scalability, availability, and fail tolerance, in line with effective integration with modern data analytics tools. The pre-processing pipeline can trigger the Deep Learning-based

(BERT-based, with high-level Keras API) model generation (Figure 5).

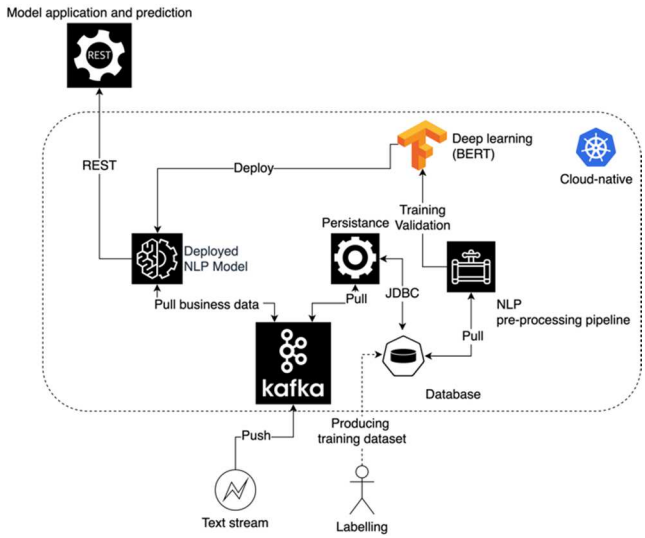

Figure 5. Cloud-native NLP architecture

C. Evaluation

The proposed cloud-based solution has improved the overall productivity in each use-case (classification, entity recognition). The human resource involvement only at the Labelling part is needed for creating a valid training dataset. The main issue was integration. In the production, each device provides its own measurements. Creating a common schema is challenging for the implementation team.

IV. Conclusion

The conventional big data workflows apply mostly the streaming methodology. It empowers quick reactions to the events in the changing world. A data pipeline is an adhesive program, serving a supervised connection from data sources to machine learning systems.

The data schema ultimately influences the AI's effectiveness. The imperfect input is handled and condensed through a pipeline, accumulating a high-quality input for the algorithms. Kafka is a standard apparatus for delivering messages (data). The pushed messages are stored in the Kafka Broker. The consumer can pull the content and treat them as a stream.

The discussed processing has well-defined operations. The transformed dataset is a legitimate input for the machine learning methods. Based on the data pipeline's output, the retraining and redeployment can be triggered without noticeable human assistance.

References

[1] S. Fu *et al.*, "Automated Detection of Periprosthetic Joint Infections and Data Elements Using Natural Language Processing," *Journal of Arthroplasty*, pp. 1–5, 2020, doi: 10.1016/j.arth.2020.07.076.

[2] S. Das, S. K. Das Mandal, and A. Basu, "Mining multiple informational text structure from text data," *Procedia Computer Science*, vol. 167, pp. 2211–2220, 2020, doi: 10.1016/j.procs.2020.03.273.

[3] D. Wu, L. Zhu, X. Xu, S. Sakr, D. Sun, and Q. Lu, "Building pipelines for heterogeneous execution environments for big data processing," *IEEE Software*, vol. 33, no. 2, pp. 60–67, 2016, doi: 10.1109/MS.2016.35.

[4] P. Sardar, J. D. Abbott, A. Kundu, H. D. Aronow, J. F. Granada, and J. Giri, "Impact of Artificial Intelligence on Interventional Cardiology: From Decision-Making Aid to Advanced Interventional Procedure Assistance," *JACC: Cardiovascular Interventions*, vol. 12, no. 14, pp. 1293–1303, 2019, doi: 10.1016/j.jcin.2019.04.048.

[5] B. M. Henrique, V. A. Sobreiro, and H. Kimura, "Literature review: Machine learning techniques applied to financial market prediction," *Expert Systems with Applications*, vol. 124, pp. 226–251, 2019, doi: 10.1016/j.eswa.2019.01.012.

[6] W. Etaiwi and G. Naymat, "The Impact of applying Different Preprocessing Steps on Review Spam Detection," *Procedia Computer Science*, vol. 113, pp. 273–279, 2017, doi: 10.1016/j.procs.2017.08.368.

[7] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP," *Procedia Computer Science*, vol. 117, pp. 256–265, 2017, doi: 10.1016/j.procs.2017.10.117.

[8] L. Gligic, A. Kormilitzin, P. Goldberg, and A. Nevado-Holgado, "Named entity recognition in electronic health records using transfer learning bootstrapped Neural Networks," *Neural Networks*, vol. 121, pp. 132–139, 2020, doi: 10.1016/j.neunet.2019.08.032.

[9] H. Liu, Y. Perl, and J. Geller, "Concept placement using BERT trained by transforming and summarizing biomedical ontology structure," *Journal of Biomedical Informatics*, vol. 112, no. October, p. 103607, 2020, doi: 10.1016/j.jbi.2020.103607.

[10] Google, "Tensorflow," 2020. .

[11] R. Mitchell *et al.*, "Exploration of Workflow Management Systems Emerging Features from Users Perspectives," *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, pp. 4537–4544, 2019, doi: 10.1109/BigData47090.2019.9005494.

[12] T. Kolajo, O. Daramola, and A. Adebiyi, "Big data stream analysis: a systematic literature review," *Journal of Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0210-7.

[13] M. Nahri, A. Boulmakoul, L. Karim, and A. Lbath, "IoV distributed architecture for real-time traffic data analytics," *Procedia Computer Science*, vol. 130, pp. 480–487, 2018, doi: 10.1016/j.procs.2018.04.055.

[14] R. Sahal, J. G. Breslin, and M. I. Ali, "Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case," *Journal of Manufacturing Systems*, vol. 54, no. March 2019, pp. 138–151, 2020, doi: 10.1016/j.jmsy.2019.11.004.

[15] A. Oussous, F. Z. Benjelloun, A. Ait Lahcen, and S. Belfkih, "Big Data technologies: A survey," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 4, pp. 431–448, 2018, doi: 10.1016/j.jksuci.2017.06.001.

[16] Z. Li *et al.*, "DLHub: Simplifying publication, discovery, and use of machine learning models in science," *Journal of Parallel and Distributed Computing*, vol. 147, pp. 64–76, 2021, doi: 10.1016/j.jpdc.2020.08.006.

[17] N. Passalis and A. Tefas, "Learning bag-of-embedded-words representations for textual information retrieval," *Pattern Recognition*, vol. 81, pp. 254–267, 2018, doi: 10.1016/j.patcog.2018.04.008.