CircleNLU: A Tool for building Data-Driven Natural Language Understanding System

Vu Hoang
VinBrain JSC
v.vuhoang@vinbrain.net

Nam Pham Hai
VinBrain JSC
v.nampham@vinbrain.net

Trung Bui
Independent Researcher
bhtrung@gmail.com

Vu Tran Hoang
VinBrain JSC
v.vuth5@vinbrain.net

Minh Nguyen Phuc
VinBrain JSC
v.minhng@vinbrain.net

Steven Q.H. Truong
VinBrain JSC
brain01@vinbrain.net

Abstract

Although many open-source developer tools exist for building intelligent chatbot agents, these still have limitations with a few basic features and do not provide enough support for building commercialized products and low-resource languages like Vietnamese with a whole Natural Language Understanding (NLU) life-cycle. Thus, building an in-house system will be necessary for enterprises to have a unique competitive advantage. To fill this gap, we introduce CircleNLU as a Data-Driven NLU system with a range of features such as pseudo annotation, massive deployment, machine learning (ML) development process, and others. This paper shares our system architecture and sees how it works.

1 Introduction

In recent years, the open-source chatbot frameworks like Rasa [1], DeepPavlov [3], and Botfront [9] have helped enterprises to gain momentum in developing maintainable and scalable software by reusing generic modules and libraries to focus on other areas of applications. Furthermore, some open-source, collaborative text annotation frameworks such as GATE Teamware [2], and WebAnno [6] leverage a client-server architecture to enable multi-user annotation projects. In addition, recently, some annotator tools that apply active learning to improve annotator productivity are Doccano [7], and some other commercial tools. Even though many of the source codes are publicly available, going from an open-sourced implementation to a production-ready system for a specific business domain is not enough. Therefore, we introduce our in-house system named CircleNLU, which is cloud-based built on Botfront, an open-source enterprise-grade conversational platform built with Rasa, and our custom building blocks to support our features. CircleNLU's main principles are:

- **ML development process**: CircleNLU has built-in features that allow users to perform training/evaluating/validating/debugging in a life cycle of ML project.
- **Massive deployment**: CircleNLU is built on the custom components that are easy to scale and deploy to integrate into existing systems.
- **Multi-role support**: CircleNLU is web-based that serves different user roles and responsibilities like development team (Annotator, Reviewer, and Manager), internal agreement measurement team (Quality checker).

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

2 CircleNLU

2.1 Model development process

Our built-in model development process is categorized into the three stages:

- **Development stage**: One of the most difficult problems facing us is how to make ML models generalize better. We probably do not know how users act in reality but only give assumptions as our initializations. At the very early stage, we follow a process with the four steps (data collection and analysis, data preparation, model training, model evaluation). We will build the ML models that are good enough for real-world scenarios through our development process.
- **Validation stage**: We define the hidden test sets collected in real scenarios which are unknown for development teams. The dataset is held by the measurement team and is used to evaluate the performance of ML models on unseen scenarios before deploying on Production. The purpose of this stage is to ensure the responsiveness of the system.
- **Production stage**: The data in this stage is handled by the process presented in the development stage. However, we only focus on data with low confidence scores based on the results of the pre-defined ML tasks to reduce verifying time while preserving data quality. We then correct and leverage data as development data for the tasks.

Repeating this process will improve the performance of the ML models over time. During the above stages, we take both our model and data approach simultaneously. Specifically, the model approach is a methodology for selecting good baseline ML models on the initial clean and high-quality data. For the other approach, observing how the system works in real scenarios will be very important to find a new pattern or failed cases. Furthermore, It is also considered as the process of collecting, cleaning, and labeling data before feeding them into the development stage. We have developed CircleNLU to be an excellent tool for building data-driven natural language understanding systems. More details about CircleNLU will be described in the next section.

2.2 System process

2.2.1 Architecture

CircleNLU is built on top of the NLU system, which are deployed from the qualified ML models described in the previous section. Figure 1 shows the overall architecture of our system. The system consists of the five main components:

- **Web app** stands for web application that provides an interface allowing the users to work on the processes defined in the whole process. The UI and data annotation process are discussed in more detail in the next section.
- **GraphQL server** exposes data as a GraphQL API for the Web app to perform data queries.
- **NLU system** trains and runs the ML models. It exposes the API for the Client to use for text parsing and CircleNLU to use for model development and validation.
- **Event server** is a message streaming service like RabbitMQ or Kafka, which receives messages or texts from the NLU system to forward them to other services.
- **Consumer** is a service to subscribe to events from Event server then make a call to GraphQL server to add events to the database. Events are texts parsed by the NLU system.

As we mentioned in the Section 2.1, at each stage, there will be participants with different roles to ensure the correctness and fairness for the deployment of ML models. Therefore, CircleNLU system defined the four roles as below:

- **Manager** has the role of ensuring the quality of data and ML models but does not know about the hidden test sets. They are knowledgeable about ML models and must understand the problem they are dealing with.
- **Annotator** is responsible for collecting and labeling from end-users.

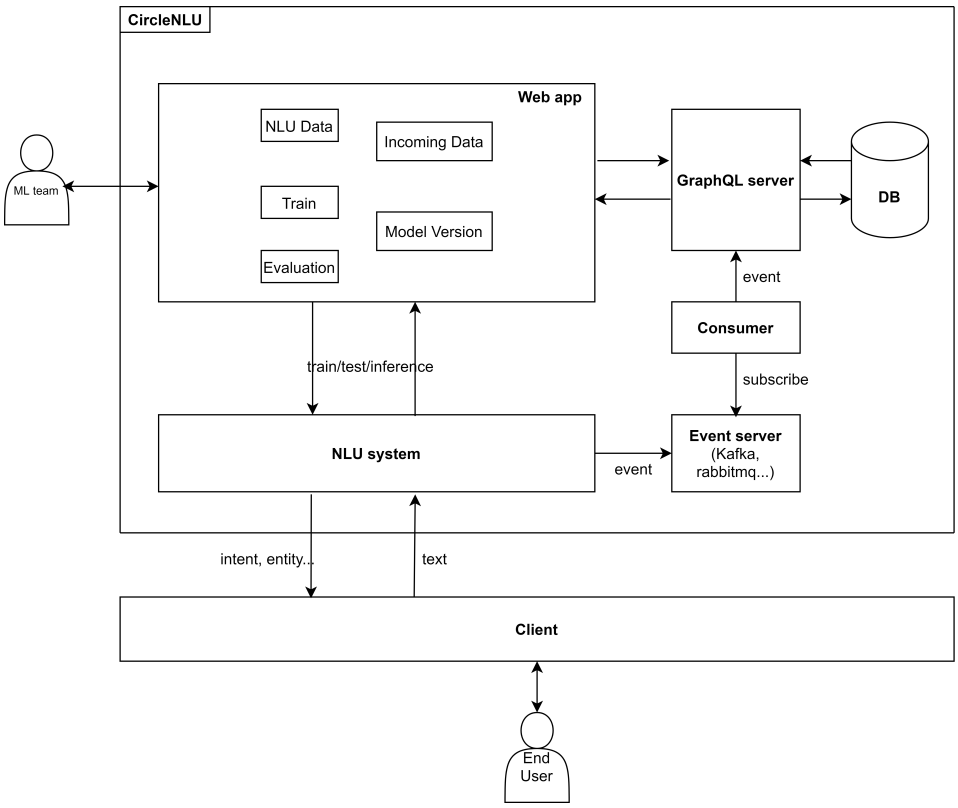

Figure 1: CircleNLU architecture.

- **Reviewer** has the role of reviewing labeled data to ensure the consistency of the development data of the ML models. They are accountable for tackling the conflict and confusion of the Annotator and sending feedback back to the Annotator to help them improve their data labeling skills.
- **Quality checker** who is representative for the measurement team to ensure the things in the validation stage happens.

2.2.2 CircleNLU in the whole process

In the early development stage Preparing clean and high-quality data is the most time-consuming and labor-intensive task. However, this helps build a good baseline ML models with acceptable performance. CircleNLU makes this more straightforward with the data annotation feature. Firstly, the Manager prepares the working environment for the team according to the workflow shown in Figure 2.

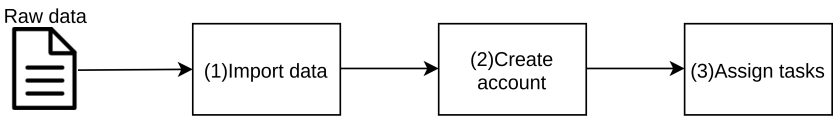

Figure 2: Data annotation process. (1). Manager imports raw data into CircleNLU, (2). Manager creates access accounts for annotators and reviewers, (3). Manager assigns tasks for annotators and reviewers

Annotator and Reviewer then do their roles and responsibilities, which are introduced in the Section 2.2.1. After the data has been labeled and reviewed, the Manager will export and send it to the development team for developing ML models, followed by the development stage. Finally, the trained model is validated on the hidden test set by the Quality checker before releasing.

In the production stage The Web app shows pre-annotated texts or messages (e.g., these are predicted by NLU system with pseudo-labels) from user's interaction by getting data from the Consumer component illustrated in Figure 3. Thus the data is controlled by the model development process.

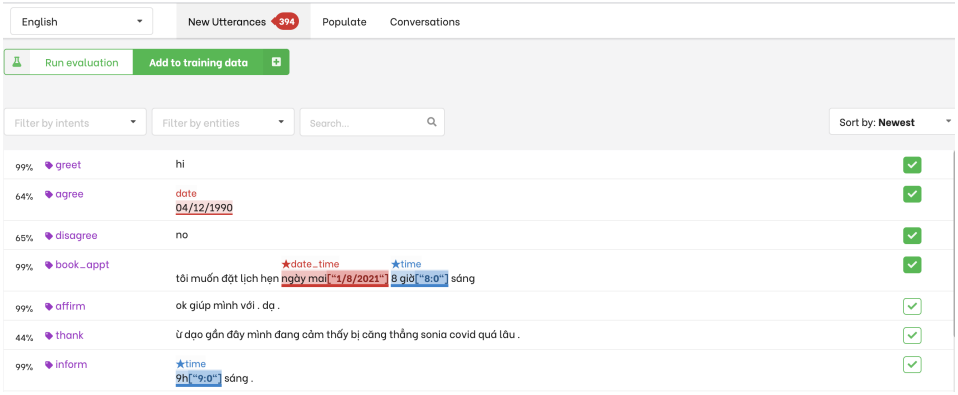

Figure 3: UI shows pre-annotated data from NLU system. (Right) Boxes marked green mean reviewed, unmarked mean not reviewed.

2.3 Case study

In the beginning, we focused on collecting/creating our dataset and building ML models for our use-case (e.g., appointment booking) based on the given scope and booking information such as Entities (Date, Time, and Person name) and Intents (Booking, Canceling, Reschedule, and others). Firstly, we employ a joint intent classification and slot filling model introduced in JointIDSF [5] with pre-trained PhoBERT [8] as a baseline model for Vietnamese. We considered PhoBERT as a good initiation because the pre-trained PhoBERT models are the state-of-the-art language models for Vietnamese. To avoid bias, we then experiment JoinIDSF with a multi-lingual pre-trained model named XLM-RoBERTa [4]. However, this shows a good baseline overall on our dataset and which is shown in Table 1. Finally, we deploy this joint XML-RoBERTa model as a service of the NLU system to run for real scenarios with our CircleNLU.

Table 1: Results with different choices on the dataset. "Intent Acc." denotes Intent accuracy of Intent classification task. Precision, recall, and F1 scores of Named Entity Recognition task are also reported.

Model	Metrics			
	Intent Acc.	Precision	Recall	F1
JointIDSF + PhoBERT	**0.934**	0.879	0.856	0.867
JointIDSF + XLM-RoBERTa	0.928	**0.898**	**0.898**	**0.898**

3 Conclusion and future work

In this paper, we introduced our in-house CircleNLU, a cloud-based system that helps the team focus on model and data approaches at each stage of the product development process to improve system performance. In future work, we will employ the system to serve our more commercial products and apply useful features such as active learning and noise reduction to increase annotator productivity and improve NLU system performance.

References

[1] Tom Bocklisch, Joe Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management. *ArXiv*, abs/1712.05181, 2017.

[2] Kalina Bontcheva, Hamish Cunningham, Ian Roberts, Angus Roberts, Valentin Tablan, Niraj Aswani, and Genevieve Gorrell. Gate teamware: A web-based collaborative text annotation framework. *Language Resources and Evaluation*, 47, 12 2013.

[3] Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lymar, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhreva, and Marat Zaynutdinov. DeepPavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[4] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116, 2019.

[5] Mai Hoang Dao, Thinh Hung Truong, and Dat Quoc Nguyen. Intent detection and slot filling for vietnamese. *CoRR*, abs/2104.02021, 2021.

[6] Seid Muhie, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. Webanno: A flexible, web-based and visually supported system for distributed annotations. pages 1–6, 08 2013.

[7] Hiroki Nakayama, Yasufumi Taniguchi Takahiro Kubo, Junya Kamura, and Xu Liang. Software available from https://github.com/doccano/doccano. `https://github.com/doccano/doccano`, 2018.

[8] Dat Quoc Nguyen and Anh Tuan Nguyen. Phobert: Pre-trained language models for vietnamese. *CoRR*, abs/2003.00744, 2020.

[9] Nathan Zylbersztejn. Software available from https://github.com/botfront/botfront. `https://github.com/botfront/botfront`, 2019.