MLOps for Compositional AI

Debmalya Biswas
Data Analytics and AI
Wipro Limited
Lausanne, Switzerland

Abstract

Enterprise adoption of AI/ML services has significantly accelerated in the last few years. However, the majority of ML models are still developed with the goal of solving a single task, e.g., prediction, classification. In this context, Compositional AI envisions seamless composition of existing AI/ML services, to provide a new (composite) AI/ML service, capable of addressing complex multi-domain use-cases. In this work, we consider two MLOps aspects that need to be enabled to realize Composable AI scenarios: (i) integration of DataOps and MLOps, and (ii) extension of the integrated DataOps-MLOps pipeline such that inferences made by a deployed ML model can be provided as training dataset for a new model. In an enterprise AI/ML environment, this enables reuse, agility, and efficiency in development and maintenance efforts.

1 Enterprise AI

AI/ML use-cases are pervasive in the enterprise today. The enterprise use-cases can be broadly categorized by the three core AI/ML capabilities enabling them: Natural Language Processing (NLP), Computer Vision/Image Recognition and Predictive Analytics (summarized in Fig. 1).

While a lot of progress has been made in the field, e.g., achieving, or in some cases bettering, human level performance for NLP and Computer Vision tasks; majority of AI/ML models are still developed with the goal of solving a single task, e.g., prediction, classification. In an enterprise context, this leads to significant redundancy and time/effort waste where a new model needs to be developed from scratch for every task.

1.1 Compositional AI

In an enterprise setting with multiple existing AI/ML services, the Compositional AI [1] framework enables formation of a new composite AI/ML service - composed of multiple AI/ML services. The framework considers how we can take the underlying Data, Model, APIs, of potentially different AI/ML services; and compose new service(s) out of them, in a seamless fashion, while addressing governance, privacy, lineage and other ethical/non-functional aspects.

Compositional AI Scenario - Fig. 2. Consider the (Online) Repair Service of a Consumer Electronics Vendor. The service consists of a Computer Vision (CV) model that is able to assess the repairs needed given a snapshot of the damaged product. If the user is satisfied with the quote, the service is transferred to a Chatbot that converses with the user to capture additional details required to process the request, e.g., damage details, user name, contact details, etc.

In future, when the vendor is looking to develop a Product Recommendation Service, the Repair Service is considered. The data gathered by the Repair Service: state of products owned by the users (gathered by CV assessment model) together with their demographics (gathered by the Chatbot) provides additional labeled training data for the Recommendation Service. Privacy policies may

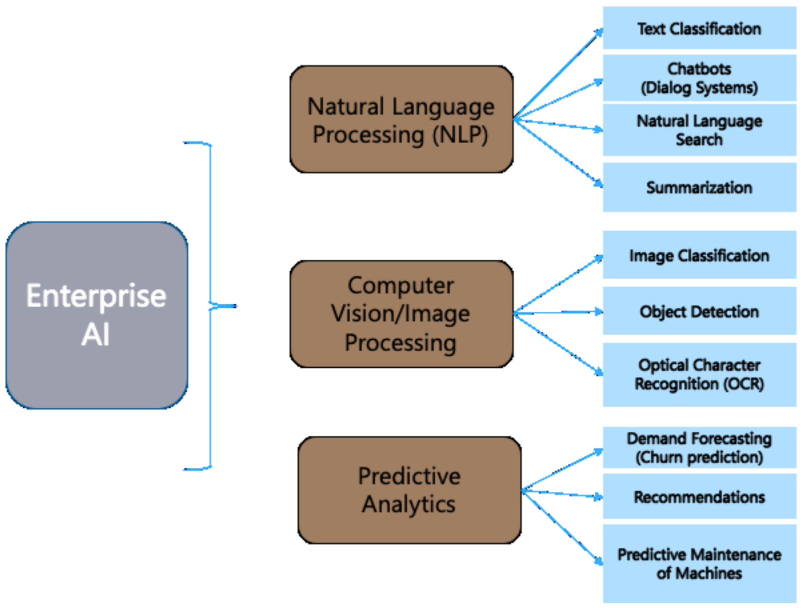
Enterprise AI
Natural Language Processing (NLP)
Text Classification
Chatbots (Dialog Systems)
Natural Language Search
Summarization
Computer Vision/Image Processing
Image Classification
Object Detection
Optical Character Recognition (OCR)
Predictive Analytics
Demand Forecasting (Churn prediction)
Recommendations
Predictive Maintenance of Machines

Figure 1: Enterprise AI/ML use-cases.

prevent their data from being combined, such that, they cannot be used to profile customers "data used for a different purpose than originally intended". So appropriate measures need to be taken here to preserve user privacy [2].

We continue the compositional scenario where the vendor further wants to develop a Computer Vision (CV) enabled Manufacturing Defect Detection Service. Recall that the Repair Service has labeled images of damaged products, so they can be leveraged here as well. The labeled images can also be provided as a feedback loop to the Repair Service - to improve its underlying CV model.

1.2 Related Work

Compositionality refers to the ability to form new (composite) services by combining the capabilities of existing (component) services. The existing services may themselves be composite, leading to a hierarchical composition. The concept is not new, and has been studied previously in different contexts; most notably, Web Services Composition and Secure Composition of Security Protocols.

Web Services follow the Service Oriented Computing (SOC) approach of wrapping a business functionality in a self-contained Service. There are mainly two approaches to composing a service: dynamic and static. In the dynamic approach, given a complex user request, the system comes up with a plan to fulfill the request depending on the capabilities of available Web services at run-time. In the static approach, given a set of Web services, composite services are defined manually at design-time combining their capabilities. There has been considerable work on addressing discovery [3], monitoring [4] and reliability [5] aspects of Web Services Compositions [6]. A good example of a secure composition protocol is the Universally Composable (UC) framework of Ran Canetti [7]. Given a complex task, the basic idea is to partition the task into multiple simpler sub-tasks. Then, design protocols to securely realize the sub-tasks. The UC-framework ensures that a protocol composed from (UC-secure) sub-protocols, will continue to guarantee security in novel, unpredictable and adversarial execution environments, even with other protocols running concurrently.

In a Machine Learning (ML) context, Ensemble Learning [8] attempts to optimize the predictions from multiple models, however *catering to the same problem*. Commonly used Ensemble Learning techniques include: Bagging, Boosting and Stacking. The basic idea is to split the training data, build models on the split datasets, and then combine the models in case of prediction, this might mean averaging the predicted values; in case of classification, this might imply choosing the output with the highest precision. The 'ensemble' in this case refers to fusing models, but not necessarily fusing models addressing different problems - the objective of Compositional AI.

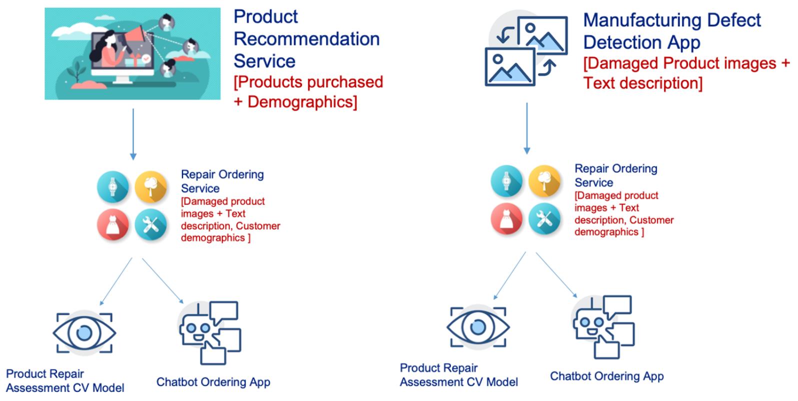
Product Recommendation Service
[Products purchased + Demographics]
Manufacturing Defect Detection App
[Damaged Product images + Text description]
Repair Ordering Service
[Damaged product images + Text description, Customer demographics]
Repair Ordering Service
[Damaged product images + Text description, Customer demographics]
Product Repair Assessment CV Model
Chatbot Ordering App
Product Repair Assessment CV Model
Chatbot Ordering App

Figure 2: Compositional AI scenarios.

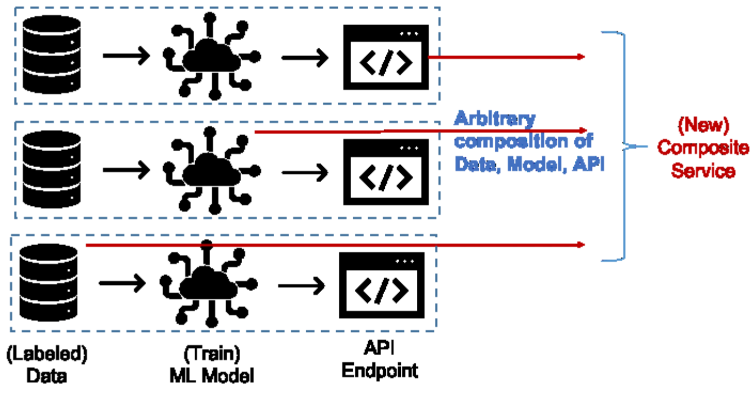

Figure 3: Compositional AI architecture.

2 MLOps for Compositional AI

At its core, an AI Service consists of (labeled) data used to train a model, which is then exposed as an API (Fig. 3). There is of course an alternate deployment pipeline, where a trained model can be deployed on an edge device to be executed in an offline fashion. It is interesting to note that there are overlapping methodologies today, DataOps, MLOps and APIOps (or API Management) addressing the operational aspects of Data, Model and API, respectively.

2.1 Bridging DataOps and MLOps

In this section, we outline how the two key frameworks enabling Data and ML pipelines overlap, and how best to integrate them in the context of Compositional AI.

DataOps [9] is defined as "an automated, process-oriented methodology, used by analytic and data teams, to improve the quality and reduce the cycle time of data analytics." - Wikipedia. A simplified DataOps pipeline is illustrated in Fig. 4 (upper box labeled DataOps). Source data, both structured and unstructured, is ingested into the Bronze layer, where it is cleansed and standardized into the Sliver layer, with further modeling and transformation into the Gold layer. In theory, the data is now ready for consumption by both BI/Reporting tools and ML pipelines. However, the data (pre-)processing part of MLOps [10] focuses on moving data from the Source to ML model, without necessarily including how the model executes on the data itself. This commonly includes a series of transformations that support a learning algorithm. For example, a Data Scientist may choose to build a linear regression pipeline or an exploratory factor analysis pipeline to support ML models.

ML training thus requires performing more complex functions than that supported by traditional Extract Transform Load (ETL) tools. This is often the case in complex data processing, aggregation

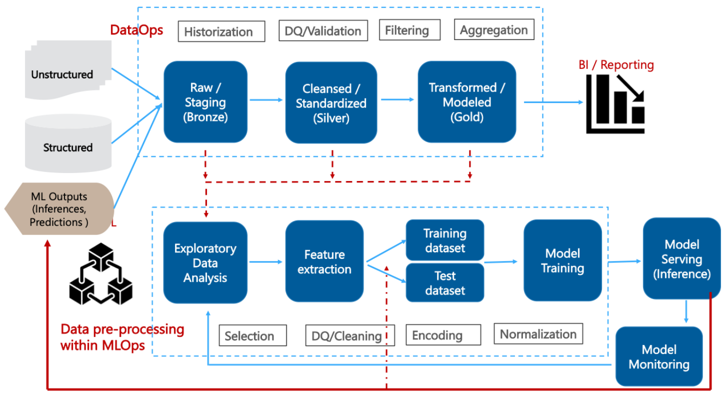

Figure 4: Data/MLOps in Compositional AI.

and regression. The recommended approach here is to complement the data processing strategy with Directed Acyclic Graph (DAG) flows. In contrast to the more linear data flows in case of BI, DAG flows support scalable directed graphs for data routing, statistical transformation and system logic. Tools like Apache Airflow [11] support the authoring, management and maintenance associated with DAG flows, which can then be programmatically authored to integrate with ETL pipelines. Needless to say, this results in redundancy and a fragmentation of the DataOps and MLOps pipelines. It seems fair to say that DataOps today relates more to BI/structured data analytics, and MLOps addresses the full ML pipeline with data (pre-)processing embedded within it.

Tool/platform vendors have started working towards this, and we have seen some initial offerings to resolve this. Snowflake recently announced Snowpark Python API [12] that allows ML models to be trained and deployed within Snowflake, with Snowpark allowing data scientists to use Python (rather than writing code in SQL). Google Cloud Platform (GCP) provides BigQuery ML [13], a tool that allows ML models to be trained purely using SQL within GCPs DWH environment. Similarly, AWS Redshift Data API [14] makes it easy for any application written in Python to interact with Redshift. This allows a SageMaker notebook to connect to the Redshift cluster, and run Data API commands in Python. The in-place analysis provides an effective way to pull data directly into a notebook from DWH in AWS.

2.2 ML Model Inferences as a new Data Source

In this section, we consider another missing 'data' aspect of an MLOps pipeline where a deployed ML model generates new data - acts as a Data Source. In the Compositional AI context, we have seen examples of both scenarios where (i) the inputs provided by users to a deployed ML model (ref. Repair Ordering Service) is provided as a feedback loop to augment the existing training dataset of another deployed model (ref. Product Repair Assessment CV model), and (ii) as training dataset for a new model (ref. Product Recommendation Service).

This aspect is accommodated in Fig. 4 by extending the MLOps pipeline with an additional Data Source Box, such that user inputs provider to a (deployed) ML model and subsequent inferences can be leveraged to augment existing datasets and generate new datasets. Synthetic data, i.e., data that is synthetically generated to closely resemble the original training dataset (based on generative neural networks) can also be considered as an additional data source along the same lines.

3 Conclusion

As ML models proliferate in the enterprise, Compositional AI has the potential to enable reuse of (training) data and models, improving agility and efficiency in ML model development. In this paper, we highlighted two aspects of MLOps needed to enable Compositional AI, primarily an integrated DataOps-MLOps pipeline that leverages model inputs and inferences to augment/generate new training data.

References

[1] Biswas, D. (2021) Compositional AI: Fusion of AI/ML Services. In proceedings of the *Data Fusion Conference*. Also published in Towards Data Science, `https://towardsdatascience.com/compositional-ai-the-future-of-enterprise-ai-3d5289dfa888`

[2] Biswas, D., Vidyasankar, K. (2021) A Privacy Framework for Hierarchical Federated Learning. In proc. of the *International Workshop on Privacy, Security and Trust in Computational Intelligence (PSTCI2021)*. `https://ceur-ws.org/Vol-3052/paper17.pdf`

[3] Biswas, D. (2007) Web Services Discovery and Constraints Composition. In proc. of the *International Conference on Web Reasoning and Rule Systems (RR)*: 73-87.

[4] Biswas, D., Vidyasankar, K. (2005) Monitoring for Hierarchical Web Services Compositions. In proc. of the *International Workshop on Technologies for E-Services (TES)*: 98-112.

[5] Biswas, D. (2004) Popular Ensemble Methods: An Empirical Study. In proc. of the *International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*: 69-80.

[6] Sheng, Q. Z., et. al. (2014) Web Services Composition: A Decades Overview. *Information Science*: 280-218

[7] Canetti, R. (2020) Universally Composable Security. *J. ACM* 67, 5, Article 28.

[8] Opitz, D., Maclin, R. (1999) Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research* 11: 169-198.

[9] Ereth, J. (2018) DataOps - Towards a Definition. In proc. of *Lernen, Wissen, Daten, Analysen Conference*: 104-112.

[10] Canuma, P. (2022) MLOps: What It Is, Why It Matters, and How to Implement It. `https://neptune.ai/blog/mlops`

[11] Chicago Data Science. (2020) Apache Airflow: Operationalizing Machine Learning. `https://chicagodatascience.github.io/MLOps/lecture5/airflow/`

[12] Snowflake. (2022) Snowpark Developer Guide for Python. `https://docs.snowflake.com/en/developer-guide/snowpark/python/index.html`

[13] Google Cloud Platform. (2022) What is BigQuery ML? `https://cloud.google.com/bigquery-ml/docs/introduction`

[14] Amazon. (2022) Using the Amazon Redshift Data API. `https://docs.aws.amazon.com/redshift/latest/mgmt/data-api.html`