Jin Dong

dongjin@microsoft.com | +1 438 334 3456 | www.linkedin.com/in/jdong95

Education

McGill University | School of Computer Science Montreal, Canada

M.S. in Computer Science, GPA: 4.00 / 4.00 *2018.09 - 2020.08*

- **Relevant Course:** Distributed System, Deep Learning, Graph Representation Learning, Reinforcement Learning, Natural Language Processing, Computer Vision, Applied Machine Learning (Teaching Assistant).

Jilin University | School of Computer Science Changchun, China

B.S. in Computer Science *2013.09 - 2017.07*

- **Relevant Courses:** Analysis of Algorithm, Data Structures, Database Fundamentals, Operation System, Java Programming (Teaching Assistant).

Technical Skills

Programming Languages	Python, C/C++, GoLang, C#, Java, Scala
Frameworks	PyTorch, TensorFlow, Kubernetes, Docker, Spark, MySQL, Flask, Bootstrap
Tools	Git/Github, Azure/GCP, VS, PyCharm, IntelliJ, Scrum

Work Experiences

Microsoft

Software Engineer, Data&Insight, WebXT *2020.10 - Present*

- Built an end-to-end data pipeline using C# and Cosmos Scope that collects data from Bing users.
- Reduced the runtime of an existing MSN data pipeline from 1 hour to 20 minutes, by optimizing the data extractor and processor.
- Built a data quality data model and dashboard for data pipelines from 21 revenue partners across 5 pipeline stages.

Google Summer of Code | TensorFlow

Student Developer, TensorFlow Hub Team *2020.05 - 2020.08*

- Built command line tools for fine-tuning machine learning models based on TensorFlow Hub.
- Optimized the image classification fine-tuning tool by supporting multi-GPU training, updating to TF 2.0 Dataset pipeline and adding training tricks.
- Created an object-detection tool from scratch, enabling users to fine-tune a pretrained object detector without writing code.
- Extended an embedding indexing tool to the image domain, enabling similarity match for images based on Apache Beam and Spotify Annoy.

Amazon | AWS AI Lab

Applied Scientist Intern, DGL Team *2019.06 - 2019.08*

- Built a complete knowledge graph embedding (KGE) module for Deep Graph Library (DGL) using PyTorch and DGL.
- Implemented eight KGE models, and matched five evaluation metrics with respective reported results on all models.
- Built an efficient negative sampling method and gained 20x speedup on sampling using C++.
- Scaled the module to large knowledge graphs with millions of nodes and billions of edges, by utilizing a K-V store server.

Quebec AI Institute (Mila) & RL Lab, McGill

Research Assistant, Advisor: Prof. William Hamilton *2019.01 - 2020.04*

- Constructed a Q&A benchmark dataset focusing on relational reasoning and systematic generalization, by Parl AI Mechanical Turk.
- Built both non-structured and structured baseline models for this dataset, including LSTM, MAC network, and graph attention network by PyTorch.
- Evaluated models' generalization ability by incorporating relations with different types and length into training set and test set, respectively.
- Published two research paper on EMNLP conference as first-author and co-author respectively.

Tencent | WeChat Group

Software Engineer Intern *2016.12 - 2017.03*

- Built an attention-based bi-directional LSTM model for joint intent detection and slot filling.
- Created a sequence-to-sequence dialog system with LSTM and trained on Weibo Dialog dataset using TensorFlow.
- Incorporated an attention mechanism that could generate various and logical responses.

Project Experiences

Distributed Key-Value Storage with Fault-Tolerant Guarantee *2020.04 - 2020.07*

- Built a distributed Key-Value Storage using GoLang.
- Implemented the Raft consensus algorithm to guarantee fault-tolerance, passed 1000 testcases.
- Created the key-value storage application by adding client-end implementations.
- Distributed the application by implementing a shard master and shard servers.

Distributed Travel Reservation System *2018.09 - 2018.11*

- Constructed a multi-client, middleware, multi-server architecture using Java remote method invocation (RMI) and TCP sockets.
- Implemented transaction management and concurrency control using 2-phase locking.
- Achieved fault tolerance using 2-phase commit and crash recovery with data shadowing.
- Distributed the complete system on 8 machines and passed all tests on concurrent request, dead lock, crashing, and data recovery.