Jiseung Hong

Seoul, Republic of Korea

+82-10-9658-8058
jiseung.hong@kaist.ac.kr
Personal Website
GitHub Profile
LinkedIn Profile

Education

•**Korea Advanced Institute of Science and Technology, Republic of Korea** *03.2017 - 02.2024*
BSc, School of Computing GPA/4.3: (CS) 3.67; (Overall) 3.50

•**Sejong Science High School, Republic of Korea** *03.2015 - 02.2017*
High School Diploma

Experience

•**NCSOFT** *03.2021 - 09.2021*
Internship as a software developer, Natural Language Processing Narrative Laboratory Seongnam, Republic of Korea
- I participated in a project cooperated with Yonhap News, one of South Korea's largest broadcasting companies. In this project, we clustered news articles and utilized **Learning-To-Rank (LTR)** techniques to organize clusters in order of relevance when given a query.

•**SK hynix** *12.2019 - 03.2020*
Internship as a software developer, Mobile Design and Operations Department Seongnam, Republic of Korea
- I designed a **Flash Memory Translation Layer (FTL)**, responsible for mapping between the operating system and NAND flash memory. I developed techniques to enhance mapping performance, including a page mapping algorithm, garbage collection, and dynamic wear leveling.
- This internship marked my first opportunity to apply my algorithmic skills in a real-world context. Throughout the research and development progress, I further honed my ability to create algorithms that precisely meet host interface requirements.

•**Military Service** *10.2021 - 07.2023*
Republic of Korea Air Force (ROKAF); cooperated with United States Air Force (USAF) Osan Air Base, Republic of Korea
- I served as an Information Systems Management Specialist at Osan Air Base, where I was responsible for managing the software systems used by both the U.S. and South Korean Air Force. During my 21-month assignment, I participated in **three joint U.S.-South Korea exercises**. In every exercise, I was recognized for the excellence of my work in the information systems department and was awarded a coin by a U.S. Colonel.
- I had the opportunity to engage in daily conversations with U.S. military personnel and collaborate on troubleshooting software issues. This experience significantly enhanced my communication skills with Americans.

•**Concurrency and Parallelism Laboratory** *06.2019 - 12.2019*
Lab internship as an undergraduate student. Daejeon, Republic of Korea
- I involved in a project on improving the security of a hypervisor by porting it over from C to RUST.

Researches and Projects

•**SIn-NeRF2NeRF** *09.2023 - 12.2023*
Editing 3D Scenes with Instructions through Segmentation and Inpainting
- Language: Python 100%
- [ON GOING PROJECT] Our team is currently working on an enhanced version of Instruct-NeRF2NeRF(). This project is aimed at achieving superior results, especially on the case of reduction/ decrease in object geometry. We are incorporating SPIn-NeRF, a state-of-the-art 3D inpainting model.
- Tools & Techniques: NeRFStudio (nerfacto, in2n), OpenCV, LaMa, Colmap

•**Gravisu; GradCam Visualization Web Hosting** *09.2023 - 12.2023*
Building a user-friendly web which provides explanation of machine learning models (GradCam).
- Language: Python 41.2%, Jupyter Notebook 25.2%, JavaScript 16.5%, CSS 9.2%, HTML 7.9%
- [ON GOING PROJECT] Our team is currently working on an AI model explanation platform, which utilizes GradCam as a visualization tool. We primarily focus on a user-friendly interface (usability and accessibility) design and versatile server connection.
- Tools & Techniques: FastAPI, Uvicorn, LAVIS

•**Korean Bio-Medical Corpus (KBMC) for NER with Span Representation Analysis** *01.2023 - 06.2023*
Publicly released Korean Bio-Medical Corpus and its span representation analysis.
- Language: Python 100%, *<submitted to LREC COLING 2024, waiting for acceptance>*

- Our team introduced the Korean Bio-Medical Corpus (KBMC), the first open-source dataset for Korean medical Named Entity Recognition (NER). Furthermore, we developed KR-SpanBERT, a pre-trained model using KBMC with span representation, which demonstrates extensive coverage of medical terminology and precise boundary detection.
  This project took place at the Computational Linguistics Laboratory of Seoul National University (SNU). Collaborating with linguistics majors, I learned how to handle the complexities of the Korean language, known for its rich morphology, and built the language model with a linguistic approach, focusing on terminology preservation and boundary detection.
- Tools & Techniques: PyTorch, MedSpaCy, Open-source Korean Text Processor

•**Ranking Clustered News Articles** *03.2021 - 09.2021*

NCSOFT Internship: Building a prototype that ranks clustered news articles based on their relevance.

- During my internship, I developed a prototype for ranking clustered news articles by their relevance, given a query. I employed a tool called Learning-To-Rank and utilized a pairwise dataset to train the model in a supervised manner. This was a significant improvement over the previous method employed by clients, which relied on human-annotated heuristics. Despite using a simple SVM model, I managed to enhance its performance significantly. The major things I found important from the internship are three-fold: (1) proactive client communication for establishing well-defined objectives, (2) the substantial impact of feature engineering on model performance, and (3) the effectiveness of SVM, despite its simplicity and compact size.
- Tools & Techniques: Learning to Rank, SVM, feature engineering

•**Abstract Text Summarizing Model for Title Generation** *09.2020 - 12.2020*

Transformer based abstract text summarization model that automatically generates headlines from news articles.

- Language: Python 88.8%, Jupyter Notebook 8.2%, C++ 1.9%
- Our team developed a Transformer-based model for generating article titles through abstract summarization. The model comprises four key steps: data crawling, sentence extraction, model training, and style transfer. These processes allowed us to fine-tune the Transformer model effectively, leading to superior BLEU score.
- Tools & Techniques: Tensorflow, Transformer, Newspaper3k, NLTK/KoNLPy, Canrevan, RegEx

•**Any Questions?** *09.2020 - 12.2020*

Motion detector for interactive online real-time class.

- Language: Jupyter Notebook 53.5%, Python 46.5%
- Our team developed a project that aims to improve online education platform (Zoom) by creating a prototype with gesture recognition using OpenPose technology. The prototype has two functions: Recognizing hand gestures and detecting big movements. It consequently seeks to encourage student engagement and help instructors in monitoring classes and exams.
  We employed a predefined metric to detect significant movements, but for hand gesture recognition, we trained a basic Multi Layer Perceptron (MLP) using an open-source dataset. Despite its simplicity, training the model to perform a function was a meaningful achievement. Moreover, although we couldn't integrate the prototype with the Zoom Software Development Kit (SDK) due to a few deprecated functions at the time, our goal was to ensure human interaction with the task.
- Tools & Techniques: PyTorch, Openpose, Scikit-learn

•**Translation of Scripts with Automatic Lip Synchronization** *03.2020 - 06.2020*

Translating a Korean script into English while considering viseme and timing simultaneously.

- Language: Python 100%
- Our team devised a novel method for translating Korean movie scripts into English while ensuring that the translated sentences closely matched the original visemes and maintained similar sentence lengths. Our approach translates a Korean sentence into several plausible English sentences. Subsequently, we break down these translated sentences into visemes, aiming to identify the best-fit sentence that harmonized both in terms of visemes and sentence length.
- Tools & Techniques: KoNLPy, Mecab, G2P

•**Building Mini-C Interpreter** *03.2020 - 06.2020*

Mini-C interpreter with CLI debugger, lexer, parser, Visitor method, and syntax/ compile time error handling.

- Language: Python 100%
- Our team developed a mini-C language interpreter by implementing code execution and return value capture. Additionally, we created a Command Line Interface (CLI) debugger and a tool for handling syntax and compile errors. The compiler included key components: a lexer for code tokenization, a parser constructing an Abstract Syntax Tree (AST), a Visitor method for error checks and variable storage, and a terminal-based debugger for line-by-line code analysis.
- Tools & Techniques: Lex & Yacc, Visitor Design Pattern, Threading

Achievements

•**SK Hynix Scholarship for Excellence**$1,000/month *03.2020 - 02.2021*

•**KOSAF National Scholarship**$100/month + $27,000/year *03.2017 - 02.2024*

•**ROKAF Excellence Training Performance**4th/1700 *11.2021*