PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

UNDERSTANDING NATURAL LANGUAGE DIRECTIONS FOR ROBOTIC INDOOR NAVIGATION

PATRICIO CERDA MARDINI

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
ÁLVARO SOTO ARRIAZA

Santiago de Chile, March 2021

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

UNDERSTANDING NATURAL LANGUAGE DIRECTIONS FOR ROBOTIC INDOOR NAVIGATION

PATRICIO CERDA MARDINI

Members of the Committee:
ÁLVARO SOTO ARRIAZA
JORGE BAIER ARANDA
CARLOS HERNÁNDEZ ULLOA
ÁNGEL ABUSLEME HOFFMANN

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, March 2021

Gratefully to my family and friends

ACKNOWLEDGEMENTS

There are many people whose help and support was key to see this thesis through to its completion, I am very grateful to them all, even if I forget to mention their name here.

First and foremost, I would like to thank my parents Patricio and Raquel for giving me the support and opportunity to live and share with the world, and for teaching me many virtues that shape me into the person I am today.

I would also like to thank my brothers Diego and Bruno for their encouragement and support in the most mundane ways. In the same vein, I thank my canine companions throughout these years: Buster, Vivi, Bari, and Tim.

To my grandparents Humberto, Patricio, Raquel and Rosario, thank you for sharing your advice, encouragement and experiences with me.

To all my friends, and in particular to Bastián, Santiago, Nicolás, Jeremy, Tomás, Dusan, Hugo, Felipe, JP, Max, Felipe, Vicente; thank you for the good times.

To the people that helped me develop an interest in robotics: Miguel Torres, Sergio Aguilera, Jorge Reyes, and Iván Lillo, thank you.

To the robotics group members Gabriel, JP, Paca, Felipe and Vladimir, thank you for the weekly meetings, interesting discussions, mentoring, helpful advice and feedback.

Finally, a huge thank you to the *IA Lab* students and faculty for the amazing times and opportunities. In particular, to my supervisors and advisors: Álvaro Soto, Hans Löbel and Denis Parra.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Smart robots in industrial, service and household environments are increasingly present in modern society. One necessary skill for their general usefulness is that of being able to navigate as instructed by a person through the same physical indoor spaces that humans use. However, this is a hard problem that is considered unsolved by the robotics community at large. With the advent of deep learning, current approaches use deep neural networks in various ways, leveraging multiple types of information that the robot either perceives or knows, such as vision, topological maps, or sound. Maps, in particular, are key when challenging a robot to interpret natural language directions for indoor navigation, as a map contains valuable insights with respect to the environment topology, something that humans normally consider.

This document explores augmentations to a supervised machine learning architecture that translates an unconstrained natural language direction "aligning" it with the topological information to a sequence of pre-defined, semantically meaningful behaviors that the robot will execute according to the directions. In particular, we analyze multi-head attention and graph attention networks to improve generalization in unseen maps as compared to a neural encoder-decoder baseline. We confirm that unseen map generalization can be improved, albeit with a performance hit for maps that are seen during training. This implies that an appropriate structural prior over the architecture can be helpful, and leaves room for future work to minimize the performance hit in seen environments.

Keywords: artificial intelligence, machine learning, natural language processing, human-robot interaction, cognitive robotics, social robotics, graph neural networks

RESUMEN

Los robots inteligentes son cada vez más necesarios en ambientes industriales, caseros, y de servicio. Una habilidad clave para su utilidad general es la de poder navegar en espacios de interior cotidianos según les sea instruido. Sin embargo, este es un problema aún no resuelto en la robótica. Gracias a la revolución del aprendizaje profundo, las soluciones actuales utilizan redes neuronales profundas para aprovechar la información multimodal que un robot conoce de antemano, o que percibe mediante sensores. Los mapas, en particular, sirven para que un robot interprete correctamente instrucciones en lenguaje natural para navegar por un ambiente de interior, pues el mapa posee información útil de la topología del ambiente.

Esta tesis propone modificaciones a una arquitectura de aprendizaje de máquina supervisado que traduce la instrucción de libre sintaxis –usando la topología del mapa– a una secuencia de comportamientos predefinidos de "alto nivel" semántico que el robot ejecutará. En particular, exploramos las técnicas *multi-head attention* y *graph attention networks* para lograr una mejor generalización sobre ambientes desconocidos comparado a la línea de base neuronal de tipo *encoder-decoder*. Nuestros experimentos demuestran que sí es posible mejorar las traducciones en ambientes nuevos, pero a expensas del desempeño en ambientes conocidos al entrenar. Se concluye que una regularización estructural puede ser útil como precedente, y se plantea como trabajo futuro mantener esta mejora sin disminuir el desempeño en ambientes conocidos.

Palabras Claves: inteligencia artificial, aprendizaje de máquina, interacción humano-robot, redes neuronales, procesamiento de lenguaje natural, robótica cognitiva

1. INTRODUCTION

1.1. Context and problem description

The field of robotics is a multidisciplinary one, where both computer science and engineering are required to achieve functional solutions. Computer scientists, in particular, have researched ways of developing smart agents inspired by cognitive science, baptizing this emerging field as "cognitive robotics" in 1993 (van Harmelen, Lifschitz, & Porter, 2008). Ever since the deep learning revolution begun –back when the 2012 ImageNet classification challenge was won by a wide margin using a convolutional neural network solution (Krizhevsky, Sutskever, & Hinton, 2012)– there has been quick progress towards computational systems that are able to complete tasks that were previously thought to be extremely hard or outright impossible for a computer agent, such as beating world champions in the game of Go (Silver et al., 2016), generating convincing synthetic free-form text (Brown et al., 2020), achieving super-human performance on Atari games (Badia et al., 2020), or building partially autonomous driving systems (Bojarski et al., 2016), among others.

In the context of cognitive robotics, industrial and household agents are very important for service roles in contemporary society. Some examples include a warehouse assistant that fetches items for more effective packaging (Enright & Wurman, 2011), a robotic arm that helps organizing the living space (Okuta et al., 2018), and a robot that learns to navigate long routes in drone delivery and indoor navigation tasks (Faust et al., 2018).

These examples show the importance and value of having agents that are able to navigate indoor environments. However, something that has not been explored with the same emphasis is how to instruct robots to navigate by interacting with them through unconstrained natural language.

Recently, there has been interesting work in the intersection of robotics and natural language processing. In (Hatori et al., 2018), natural language is explored as a way of instructing a robotic arm which item to pick from an assorted set of objects distributed in a series of container bins. Regarding mobile agents, in (Matuszek, Fox, & Koscher, 2010) a statistical model is used to tackle natural language translation to a motion plan for a robot, explicitly encoding metric information in the map that the agent can access.

In this thesis, we study the problem of successfully instructing a robotic agent to navigate indoor spaces through free-form natural language directions.

Based on the framework established by (Sepulveda, Niebles, & Soto, 2018), we frame the problem as a high-level planning task, where the robotic agent is capable of robustly executing behaviors such as "exit the room" or "cross the hall", coping with sensor noise, localization errors, and geometry variations, among other difficulties. In this formulation, we can think of any navigation plan as a sequence of high-level behaviors. Then, the problem of correctly interpreting directions is reduced to a translation problem: given the natural language instruction and information about the environment, we aim for translating to the correct sequence of behaviors that, when executed, let the robot navigate according to the intended directions.

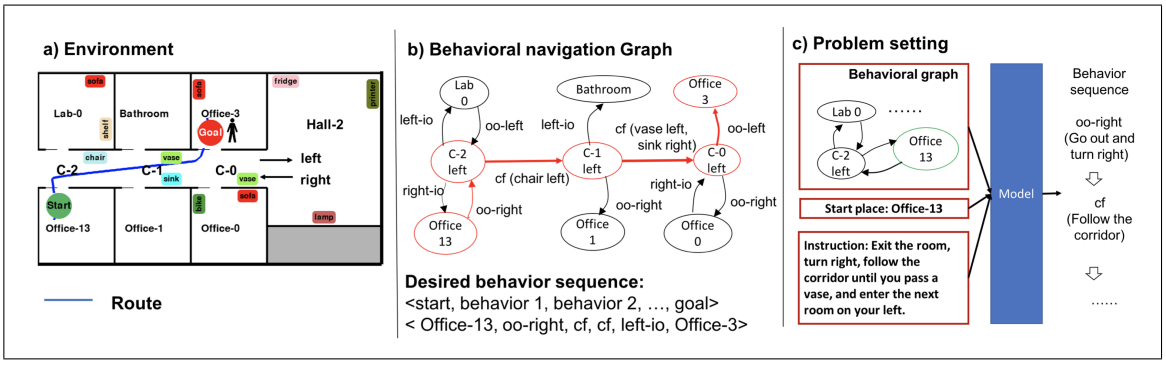

Figure 1.1. Problem definition. Map of an environment (a), its partial behavioral navigation graph (b), and the problem setting of interest (c). Figure from (Zang, Pokle, et al., 2018).

Figure 1.1 shows an example problem instance. We want the robot to navigate from the starting room "Office-13" to the goal room "Office-3" by following the route described in natural language directions, corresponding to the blue path in the environment map (a) and the red path in the graph (b). This route can be navigated by executing high-level behaviors such as `oo-right` ("go out and turn right") and `cf` ("follow the corridor"), which we assume the agent can successfully complete as in (Sepulveda et al., 2018). The framing of our problem as a translation task is depicted in (c), and formally defined in Section 4.1.

1.2. Hypotheses and objectives

Our work expands upon previous efforts that explore this problem formulation. In particular, we analyze two distinct neural network augmentations: 1) multi-head attention and 2) graph attention networks, as mechanisms that could further improve the correct translation rate.

Specifically, this thesis hypothesizes that:

(i) It is possible to develop language models that improve the translation of natural language instructions to a high-level sequential behavioral plan, using a topological map about the environment as a knowledge base.

(ii) Compared to a single-headed attention mechanism, multi-head attention (MHA) improves the alignment between natural language instructions and their associated indoor environments, boosting the performance of the model.

(iii) A graph attention network (GAT) can leverage information about the indoor environment, encoded in a topological map that the robot can access at any moment. As GATs can inductively handle unseen graphs, we expect an improved generalization capability compared to the encoder-decoder model proposed in (Zang, Pokle, et al., 2018) that we establish as a baseline.

The objective of this work is to determine whether these hypotheses hold true or not. To achieve this, we implement all modifications and integrate them with a reimplementation of the model introduced in (Zang, Pokle, et al., 2018).

We run experiments with each variant model (and hybrid combinations thereof) to compare performance and contributions towards solving the task, offering some insights into future work that could further improve these capabilities in modern robotics.

1.3. Thesis organization

The rest of this document is structured as follows: in Chapter 2, we explain concepts that are necessary for understanding the proposed approaches. Chapter 3 presents an overview of work related or relevant to this thesis, framing it with respect to the existing literature in the broader fields of machine learning, natural language processing, and robotics. Chapter 4 introduces the proposed modifications to the architecture as the main contribution of this work. In Chapter 5, we describe the conducted experiments and analyze our results. Finally, in Chapter 6 we conclude with our main findings, rejection or validation of the established hypotheses, and proposed lines of research for future work.

2. BACKGROUND INFORMATION

In this chapter we present background information that is necessary for understanding this thesis. Readers that seek more in-depth explanations are encouraged to refer to books such as (Goodfellow, Bengio, & Courville, 2016) and (Zhang, Lipton, Li, & Smola, 2020).

2.1. Machine learning

Machine learning (ML) is a subset of the computer science area known as artificial intelligence. Formally, ML is the study of computer programs that learn from experience E with respect to some class of tasks T and performance measure P. We say a program learns when the performance at tasks in T, as measured by P, improves with experience E (Mitchell, Carbonell, & Michalski, 1986). This discipline pursues the development of inductive learning in artificial agents, hoping to confer "intelligence" by correctly reasoning in novel situations through application of previously acquired knowledge.

Currently, ML application is widespread throughout academic and industrial contexts, with numerous and varied use cases, ranging from time series forecasting in financial contexts, meteorology, and recommendation systems; to vision understanding in smartphones or surveillance systems; and natural language understanding in translators, "chatbots" or text generators (LeCun, Bengio, & Hinton, 2015). Machine learning is a fundamental tool for the development and application of artificial intelligence.

The previous situation can be mostly attributed to "deep learning" techniques, which rose in popularity after increased computational power in specialized architectures such as graphics processing units (GPUs) and a bigger amount of available data made these algorithms feasible to use. Deep learning (DL) is a family of ML techniques that use "deep" neural networks as a main component. For challenging tasks, and when trained

over a large dataset, DL methods can comfortably surpass the performance of other classical ML algorithms (Krizhevsky et al., 2012; Hinton et al., 2012; Sutskever, Vinyals, & Le, 2014; Kaggle, 2014). In most of the previously mentioned examples, societies are already benefiting from DL through systems that outclass alternative approaches, sometimes even achieving super-human performance.

2.1.1. Neural networks

We can think of neural networks (NNs) in simple terms as a graph of interconnected neurons, where the topology of the graph defines different types of NNs. Each neuron is a function of an input vector x, and the output $n(x)$ can be interpreted roughly as letting the signal through, or blocking it either totally or partially. A neuron is defined by a weight vector w and a bias term b, with an activation function φ used to introduce non-linearity:

$$n(x) = \varphi(w \cdot x + b) \tag{2.1}$$

A "layer" is a set of grouped neurons, and sequentially chaining layers can enable the architecture to learn complex functions. In fact, the theorem of universal approximation states that a simple feed-forward network with a single layer can approximate any continuous function on compact subsets of $\mathbb{R}^n$, although it says nothing about the feasibility of learning weights that enable said approximation (Nielsen, 2015).

The optimization for an NN training is roughly as follows: we compute the output for a batch of training data, and calculate the error through a loss function. After this, we use the multivariate calculus chain rule to determine how much of each answer is attributed to each trainable parameter, to backpropagate this error in a proportional manner. After many iterations of this optimization process, the NN is going to find a local minimum (or maximum, depending on what we are optimizing) that is ideally close to the global extreme of the function.

2.1.2. Deep neural networks

A "deep" neural network (DNN) is an NN that has a high amount of layers, ranging from tens to hundreds of them. This architectural change promotes hierarchical compositional pattern learning, and further turns them into a tool that can capture highly complicated non-linear relationships from data.

Training these architectures normally requires huge datasets and computational resources, although usually DNN architectures are open-sourced along with pre-trained sets of weights so that anyone can use them. The training procedure is performed by optimizing with respect to a loss function that measures how far the NN is from accurately predicting data.

As of today, deep neural networks are a very popular choice for applying machine learning in industry, and are one of the most prominent research topics in ML academia (LeCun et al., 2015). The work we present in this thesis makes abundant use of three particular types of them: fully connected (Rosenblatt, 1958), recurrent (Rumelhart, Hinton, & Williams, 1986), and graph neural networks (Scarselli, Gori, Tsoi, Hagenbuchner, & Monfardini, 2009).

2.1.3. Fully connected layers

As the name implies, a pair of consecutive layers are fully connected (FC) when they are linked in such a way that every neuron in the first layer is connected with every neuron in the second layer. Due to the high connectivity that these layers have, they are very expensive in the number of trainable weights (and thus, computation) they add to a model architecture, as well as being more susceptible to overfitting the data (Zhang et al., 2020).

2.1.4. Recurrent neural networks

Recurrent neural networks (RNNs) are designed to operate over sequential inputs, where the sequence can represent a time series, or other signals through the time domain, such as language, speech, video or actions. To cope with this additional dimension, a "persistent" component is now considered: these networks evolve *hidden states* (or memories) that encode and keep track of the input seen up to a certain moment in time. To process each new item in a sequence, the input is analyzed in conjunction with the current hidden state. This temporal dynamic evolution means an RNN can handle sequential data of variable length.

There are many different RNN architectures, but two noteworthy variants are the long short term memory (LSTM) cell (Hochreiter & Schmidhuber, 1997) and the gated recurrent unit (GRU) cell (Chung, Gülçehre, Cho, & Bengio, 2014).

The LSTM cell was the first architecture to successfully alleviate the "vanishing gradient" issue, where standard RNNs could not keep track of arbitrarily long-termed dependencies in the input. This is because of the limited precision of numbers computed during the backpropagation process, so the derivative would end up being null as an approximation error, thus effectively not modifying the network weights during training. This, along with the similar "exploding gradient" issue, are serious problems that jeopardize an RNNs capability to effectively learn patterns from the data. The LSTM solved these issues by introducing a new arrangement inside the recurrent cell based on three "gates": the input, output and forget gates. With this, an LSTM could now hold states for arbitrarily long steps throughout a sequence.

The Gated Recurrent Unit (GRU) architecture, in turn, is a popular alternative to the LSTM cell, modifying the gate arrangement to omit the output gate, thus decreasing the number of trainable weights required for a similar if not equal performance under most tasks. A bidirectional variant that we use in our work, colloquially called "biGRU", is a

simple modification where the sequence is reasoned over both temporal directions by concatenating the output of two different GRU cells, where each one reads the input sequence in a different time direction.

2.2. Sequence-to-sequence architectures

Sequence-to-sequence architectures are a family of neural networks typically used for tasks like neural machine translation, image captioning, conversational models and text summarizing tasks. Their basic structure is based on a pair of RNNs: an encoder and a decoder (Sutskever et al., 2014).

The idea is to take the input sequence, which can be one-hot encodings[1] of each word or pre-trained word embeddings[2], and encode the sequence as a whole into a high-dimensional space using the encoder hidden state to capture meaning using the entire series of information. After this, said representation is decoded to generate tokens, one at a time, that will correspond to the output sequence. This model can handle variable input length as the encoder sequentially reads all tokens to compute the final descriptor. The decoder is given this last hidden state as the *only* information it has to generate the result from a special starting token `[START]`. It will sequentially output tokens until it emits a `[STOP]` token.

The fact that the decoder can only use the final encoded representation to reason about the entire input sequence is a pitfall of the first sequence-to-sequence architectures. It took the advent of attention mechanisms to infuse the models with an improved learning ability, thus promoting the widespread use of this architecture in the previously mentioned tasks.

[1] A vector filled with as many zeros as words in the vocabulary, except for a one in the index of the actual word it represents.

[2] Learned representations with rich semantic meaning. Variants include Word2Vec (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) and GloVe (Pennington, Socher, & Manning, 2014), among others.

2.3. Attention mechanisms

Attention mechanisms are mathematical constructs that can be seen as layers inside a neural network architecture. Their primary purpose is to selectively correlate one sequence of information with another by establishing relationships between the elements of each sequence.

In the context of deep learning, some of these mechanisms build an attention tensor by reasoning over a memory that the layer can access (Zhang et al., 2020). Following the terminology of (Vaswani et al., 2017), this memory can be encoded as a series of key-value pairs, and the general idea is to calculate the similarity between the keys and a query. With this, we construct the context tensor from the memory values, where the ones whose key has higher affinity to the query will have a greater contribution to the final tensor value.

Formally, let d_k, d_v, d_q denote dimensions of the key, value and query descriptors, and a memory $m = \{(k_1, v_1), ..., (k_n, v_n)\}$, with $k_i \in \mathbb{R}^{d_k},\ v_i \in \mathbb{R}^{d_v}$. An attention layer takes a query $q \in \mathbb{R}^{d_q}$ and gives back an output $o \in \mathbb{R}^{d_v}$ as follows. First, we compute the similarity scores $a_i, ..., a_n$ with a similarity function α as $a_i = \alpha(q, k_i)$. These scores are squished into a probability distribution with a softmax operation $s(a_i) = \frac{exp(a_i)}{\sum_j exp(a_j)}$ which will then provide the weight coefficients to build a final vector:

$$o = \sum_{i=1}^{n} s(a_i) v_i \tag{2.2}$$

As the softmax function yields a probability distribution, this attention variant (Bahdanau, Cho, & Bengio, 2015) can be classified as "soft". Among other types of attention that exist, "hard" attention is the use of similarity scores to select a single contributor to the output vector, effectively opting for an $argmax$ operation in lieu of softmax. "Self attention" is when all key, query and value vectors are equal, taking as input each timestep of the initial

sequence, and it is useful for discovering how each individual item of an input sequence relates to the rest of them.

In soft-attending sequence-to-sequence architectures, the decoder can attend over all encoder hidden states (one per each timestep of the input). This is important for long sequences and for tasks that may have additional difficulties, such as the problem considered in this work.

It is proposed in (Henderson, 2020) that attention mechanisms confer additional learning capabilities in sequence-to-sequence models because they fundamentally change what kind of generalizations can be made by introducing variable binding to these language models (i.e. the capacity of robustly learning sub-concepts that at first might be entangled in the training data).

2.3.1. Multi-head attention

The multi-head attention (MHA) mechanism consists of h parallel attention layers, called *heads*. The output of all heads is aggregated with some function f, which normally concatenates or averages them. The layer has to learn three different transformation matrices for the key, query and value tensors per head: $W_q^{(i)}, W_k^{(i)}, W_v^{(i)}$. With this, the MHA layer computes per each head:

$$o^{(i)} = attention(W_q^{(i)}q, W_k^{(i)}k, W_v^{(i)}v) \tag{2.3}$$

where $attention()$ is the soft mechanism from the previous section. The final operation aggregates the output from each head: $o = f(W_o(o^{(1)}, ..., o^{(h)}))$, where W_o is an output linear transformation.

2.4. Graph neural networks

Graph neural networks (GNN) are a family of neural networks that operate directly over data represented in graph format.

A graph is a widely used concept in discrete mathematics. It is a structure $\mathcal{G}$ formed by two sets: a node set $\mathcal{V}$, and an edge set $\mathcal{E}$ where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each node represents an entity or object, and each edge represents a link or relation between two nodes. In our case, nodes are indoor places and edges are high-level behaviors that the robot can execute to go from one place to the other.

Undirected graphs have no notion of orientation in their links: $(u, v) \in \mathcal{E} \iff (v, u) \in \mathcal{E}$. In this thesis, however, we work with graphs that have directed edges because high-level behaviors need not be symmetric with respect to the nodes they connect (e.g. if a is an office and b is not, then $[a,$ "exit the office", $b]$ is a valid behavior, but $[b,$ "exit the office", $a]$ is not).

According to (Wu et al., 2019), GNNs can be classified as recurrent, convolutional, graph autoencoders, or spatial-temporal. For this thesis, it is specially relevant to introduce convolutional GNNs ("ConvGNNs") in greater detail.

The family of ConvGNNs generalize the convolutional operator from a grid pattern (such as an image) to an arbitrary graph topology, where each node has its own feature. In essence, a convolution is a mathematical operation where, given two input functions f, g, it outputs a new function as follows:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{2.4}$$

The resulting convolution expresses how f changes with respect to g, as the operator reverses and then shifts g through the domain (time, for example), and for each new shift

displacement it computes the product between both f and g. The values are registered as a mapping that constitutes the output function.

There are two sub-families of ConvGNNs: spectral-based, and spatial-based. For this work, we consider spatial-based ConvGNNs. Their objective is to generate a modified node embedding v' for each original node $v \in \mathcal{V}$ in the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ by aggregating its own feature in conjunction with a local neighborhood as defined by the edges from which information is propagated, similar to a standard convolutional NN. This neighborhood can be immediately adjacent or further out, by considering more than one edge "hop". We can generate high-level node features that take into consideration increasingly larger local neighborhoods by using more than one ConvGNN layer.

One key distinction between ConvGNN architectures is whether they are capable of handling transductive or inductive tasks. Transductive tasks have a fixed graph node set, so the same nodes are used for both training and inference, like in (Kipf & Welling, 2017). Inductive tasks are those where the graph can have a variable topology, like in (Hamilton, Ying, & Leskovec, 2017).

2.4.1. Graph attention networks

In particular, Graph Attention Networks (GATs) are used in this thesis, as first proposed in (Velickovic et al., 2018). They are a type of inductive spatial-based ConvGNN, characterized by their ability to selectively attend to a variable number of neighbors. This is important because, in practice, it means that a trained GAT can operate in never-before-seen graphs, independently of their topology. For the purposes of our problem, this is a desirable feature as it lets us train over multiple environments as context to learn translations of instructions, and then at inference time we can handle environments that are new and unseen to the robotic agent in a robust way. Note that the GAT architecture internally uses the multi-headed attention mechanism.

Formally, a GAT expects as input a set of node features $h = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_N\}$ and produces a modified set of node features $h' = \{\vec{h'}_1, \vec{h'}_2, ..., \vec{h'}_N\}$, with N the number of nodes. Each original node feature is F dimensional: $\vec{h}_i \in \mathbb{R}^F$. The new features could have any dimensionality, i.e. $\vec{h'}_i \in \mathbb{R}^{F'}$, which requires a shared linear transformation as a first step, $W \in \mathbb{R}^{F' \times F}$. Next, a self-attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ determines how important each node feature is to every other node, considering the topology with a mask that computes the attention coefficients only between neighboring nodes (for the ith node, we seek e_{ij} where $j \in \mathcal{N}_i$ is the neighbor set of nodes). Normalized coefficients are obtained with the softmax function $s(\cdot)$:

$$\alpha_{ij} = s_j(e_{ij}) = s_j(a(W\vec{h}_i, W\vec{h}_j)) \tag{2.5}$$

For the mechanism a, we follow the original implementation and use a single-layered feed-forward neural network with the LeakyReLU nonlinearity. Finally, a linear combination that uses the attention coefficients produces the modified node features:

$$\vec{h'}_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} W\vec{h}_j \tag{2.6}$$

To stabilize the learning process, GAT uses MHA by defining K different attention *heads*, which means K independent mechanisms arrive at different representations as described by the previous equation, and the last step is to aggregate them with some function f. We take the average as f, which means the final representation is:

$$\vec{h'}_i = \frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k \vec{h}_j \tag{2.7}$$

Note that it is possible to attend over increasingly bigger neighborhoods for each node, either by modifying the definition of $\mathcal{N}_i$ to include nodes that are p "hops" or behaviors away, or by applying the GAT forward pass more than once to a set of descriptors (for instance, $\vec{h'_i} = GAT(GAT(\vec{h_i}))$ for $p = 2$). In this work, we consider $p = 1$.

3. RELATED WORK

There has been fast progress in the task of robotic interpretation of natural language during the last few years, in no small measure thanks to the deep learning revolution (LeCun et al., 2015), which has enabled researchers to enhance their toolboxes with techniques such as the ones presented in the last section. In particular, visual perception is now more advanced than ever thanks to convolutional neural networks, and big strides have also been made in sequence processing and understanding, thanks to recurrent neural networks (Lan et al., 2020).

A robotic navigational plan can be described through human language. Correctly interpreting and executing it requires the agent to understand and use information from the environment and other entities that move through it, while exhibiting socially acceptable behavior (Kosaraju et al., 2019). In this area, current datasets propose challenging tasks, some of which include a visual component built in various ways, for instance using point cloud technology as in Matterport3D (Chang et al., 2017).

The problem where the robot is given 1) a direction to navigate through an indoor environment following a (potentially implicit) path and 2) visual input after executing an action, is known as the Visual Language Navigation (VLN) task. Some datasets for VLN include R2R (Anderson et al., 2018), R4R (Jain et al., 2019), and ALFRED (Shridhar et al., 2020).

Although agents are increasingly able to handle more challenging scenarios, VLN tasks are still far from being solved: state-of-the-art approaches execute paths longer than most humans would deem acceptable (Majumdar et al., 2020), and for the most demanding datasets, such as Reverie (Qi et al., 2020) or RXR (Ku, Anderson, Patel, Ie, & Baldridge, 2020), the best success rates are barely a quarter than that of humans.

There is work that, instead of relying explicitly on the visual sensing of the robot, regards localization inside a metric map as in (Matuszek et al., 2010), or as a high-level construct by considering the indoor environment topology as a graph, where places are connected through the execution of high-level behaviors (Sepulveda et al., 2018). This makes sense, as most of the time humans intuitively know the semantics of each place, and make use of this information when navigating by following an instruction, sometimes even ignoring visual cues until a change is detected in the functionality or purpose of the current room.

The first work to propose this behavioral navigation approach by means of using deep neural networks is the aforementioned (Sepulveda et al., 2018), where several neural network modules are trained to learn from the visual input of the robotic agent to robustly execute high-level behaviors, such as "exit office", "follow corridor", etc. The immediate benefit of such an approach is twofold: on the one hand, robust behavior execution implies the robot can handle dynamic environments where objects or people can move about in unforeseen ways. On the other hand, planning paths for navigation in such a framework is simplified, as we just need to sequentially execute the appropriate high-level behaviors to navigate our way through the indoor environments.

In (Zang, Vázquez, Niebles, Soto, & Savarese, 2018) a proof-of-concept language model is successfully introduced to show the feasibility of this approach, and furthermore in the follow-up work (Zang, Pokle, et al., 2018) a dataset for high-level behavioral robotic navigation is introduced, along with a novel language model that is the starting point for our work.

Lastly, there is work done by (Shrestha, Pugdeethosapol, Fang, & Qiu, 2020) that closely resembles our problem definition, but crucially their translation model explicitly knows the destination room as additional information that our agent does not have. This

important difference in the task formulation complicates a straightforward result comparison to both (Zang, Pokle, et al., 2018) and our work.

4. METHODOLOGY

4.1. Problem formulation

We consider instructing a robot how to move through an indoor environment using unconstrained natural language. The robot can execute a pre-defined set of high-level behaviors, such as "follow the corridor" or "cross the hall", in a robust way (i.e. able to cope with localization errors, sensor noise, etc.) as proposed by (Sepulveda et al., 2018).

The task is to correctly translate any given instruction to a sequence of high-level behaviors that, when executed, make the robot navigate as it was asked. Following (Zang, Pokle, et al., 2018), we frame this as a supervised learning problem.

As additional information, we have $m = (\ V,\ E\ )$ the topology graph of the indoor environment, where vertices $v \in V$ represent places, and edges $e \in E$ represent connectivity between two distinct places by executing a specific behavior. Supplementary landmark information is encoded in each edge, representing specific objects the robot can see when executing a behavior (e.g. bookshelves, flowerpots, water coolers, etc). The graph is represented by a set of "triplets", where each triplet $(n_1,\ b\,[attr],\ n_2)$ associates two nodes $n_1, n_2 \in V$ through the execution of a behavior $b \in E$, observing attributes $[attr]$ along the way.

Formally, the inputs to the translation model are (1) a navigation graph m, (2) the starting node s of the robot in m, and (3) a set of free-form navigation instructions I in natural language, as seen in Figure 4.1. The instructions describe a path in the graph to reach from s to a destination node g, where both nodes might not be explicitly referenced. Table 4.1 includes a sample of instructions. The objective is to predict a suitable sequence of robot behaviors $b_1, ..., b_T$ to navigate from s to g according to I. The goal is to estimate:

$$\underset{b_1,...,b_T}{\operatorname{argmax}} P(b_1, ..., b_T \mid m, s, I) \tag{4.1}$$

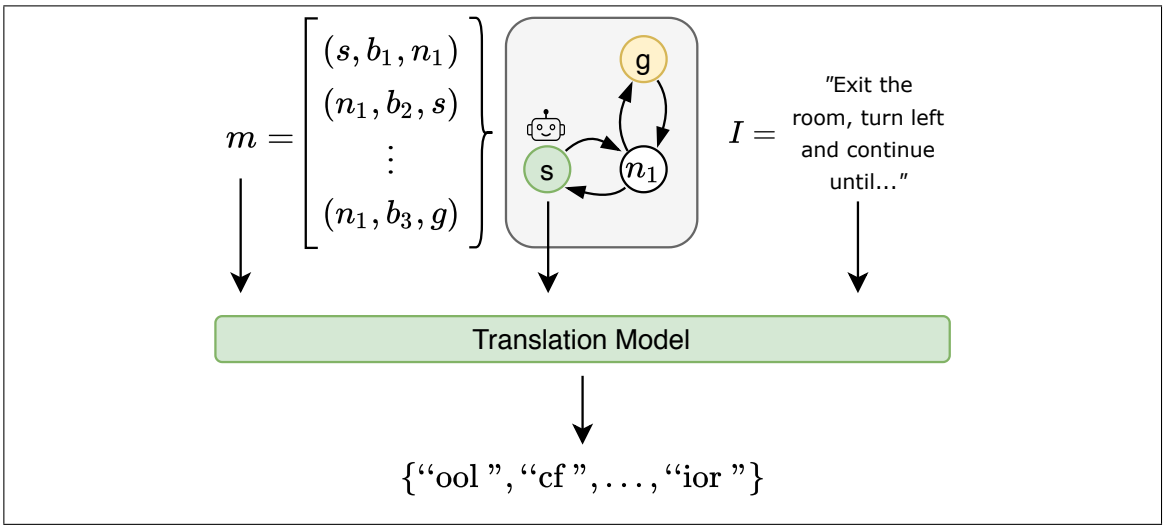

Figure 4.1. Problem definition. The translation model receives (m, s, I) as input.

The dataset consists of N input-target pairs $(x_i, y_i) \mid 0 \leq i \leq N$, with $x_i = (m, s, I)$ and $y_i = (b_1, ..., b_T)$. The sequential execution of the behaviors $b_1, ..., b_T$ should replicate the route intended by the instructions I. To train the model in a supervised manner, we define the loss function as the cross entropy over the output token set with respect to the ground truth sequence.

Table 4.1. Sampled instruction-plan pairs from the training dataset. Note that origin and destination rooms are not always explicitly referenced in the instruction.

Instruction	**Plan**
Go directly across the hall to room 0.	[oio]
Exit right. Enter the only door on your right after the turn.	[oor, rt, cf, ior]
Go out, turn right. Turn right just before you reach the locker to enter office-0.	[oor, cf, cf, cf, ior]
Leave Office-9 and take a right. Walk down the hallway and enter the second door on your right.	[oor, cf, cf, ior]
Exit room, turn right and walk to end of hall. At end of hall, turn right and enter second room on left.	[oor, cf, rt, cf, iol]

4.2. Baseline architectures

We consider two baseline models to compare against our work. The first one does not use neural networks in its formulation, but the second one does use neural networks and is the starting point for the variants that we propose.

The non-neural model we consider is proposed in (Shimizu & Haas, 2009), and establishes a baseline for minimum viable performance. Broadly speaking, the translation is done in two stages using classical machine learning techniques. First, a path candidate is generated from the natural language instruction with a standard hidden Markov model. Second, a path verification is done to ensure the path is compliant with the indoor environment, by using a depth-first search. If the route is found to be invalid, then the model might change up to three behavior tokens in the predicted sequence to turn it into a valid path.

On the other hand, the neural baseline against which we compare our results is the architecture proposed in (Zang, Pokle, et al., 2018), which constitutes a strong baseline compared to (Shimizu & Haas, 2009), particularly over environments that were seen at training time.

A summary of this model can be seen in Figure 4.2. In general terms, the architecture is a sequence-to-sequence neural network with soft attention that aligns the natural language instruction with the map topology to decode the translation, one token at a time.

We now briefly describe each of the five layers this model contemplates. The initial embedding layer takes the environment triplets and one-hot encodes them according to which behavior and nodes compose each triplet: every zero-filled vector of length[1] $|2\mathcal{V}| + |\mathcal{E}|$ has 1 as value on the indexes that correspond to each triplet component. On

[1]With $|\mathcal{V}|$ and $|\mathcal{E}|$ the number of nodes and edges in the environment graph.

the other hand, instructions are embedded per each word by means of a pre-trained 100-dimensional GLoVE mapping (Pennington et al., 2014). After this, an encoding layer takes both representations and encodes them by using two separate bi-directional GRU cell arrays to capture patterns in the time dimension for the instructions, and the triplet set span for the map. Further, an attention layer refines the descriptors by fusing information from the graph to the instructions, aligning both sources for the rest of the translation process. A fully connected layer then reduces the dimensionality of the tensors. Finally, a decoding layer takes as input the initial robot place, the FC layer output as context, and its own hidden state to emit translated behavior outputs using a GRU cell, until a `[STOP]` token is emitted. A mask is used to penalize the scores of behaviors that would not comply with the topology of each map, if executed.

To combat overfitting, dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is applied on the weights of both GRU encoders and the FC layer with probability $p = 0.5$.

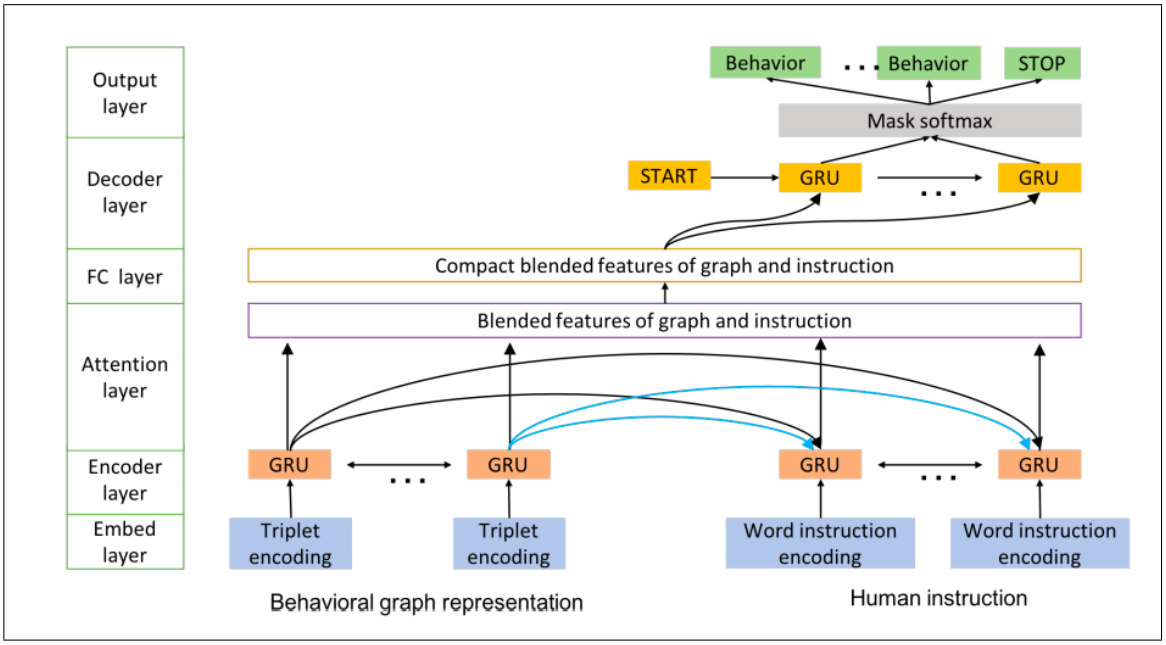

Figure 4.2. Neural baseline architecture. Figure from (Zang, Pokle, et al., 2018).

4.3. Proposed architectures

In this section, novel neural variants are introduced as alternative translators that aim to improve the neural baseline performance for our problem formulation.

4.3.1. Multi-head attention

The first modification we propose is an incremental change with respect to the neural baseline architecture. The standard attentional mechanism that blends the two source information modalities is replaced by a multi-headed attention mechanism, inspired by the success of the technique introduced in (Vaswani et al., 2017), as seen in Figure 4.3. In what remains of this document, we refer to this proposal as the "MHA" model variant.

The main intuition for this change is that with multiple heads, the blending process can specialize each head in attending to different patterns that might be more difficult to capture with just one single head, and thus even after aggregating the different attentional distributions, the embedding would provide better information in further layers.

Following the notation used in the neural baseline work, matrices $\bar{I} \in \mathbb{R}^{T \times 2H}$ and $\bar{G} \in \mathbb{R}^{L \times 2H}$ generated by the encoding layer are fused using K distinct attentional matrices $W_1, ..., W_K \in \mathbb{R}^{2H \times 2H}$, where T is the number of words in the instruction I, H the size of each GRU hidden state and L the number of triplets in the graph m. We compute the attentional distribution a_i for each encoded node $\bar{G}_i$ by aggregating all K alignments as follows:

$$a_i = softmax((\coprod_{k=0}^{K} e_i^k) \cdot W_o) \tag{4.2}$$

$$e_i^k = [\bar{G}_i W_k \bar{I}_1^\intercal, ..., \bar{G}_i W_k \bar{I}_T^\intercal] \tag{4.3}$$

with $\coprod$ the tensor concatenation operator, and W_o a learned matrix linear transform.

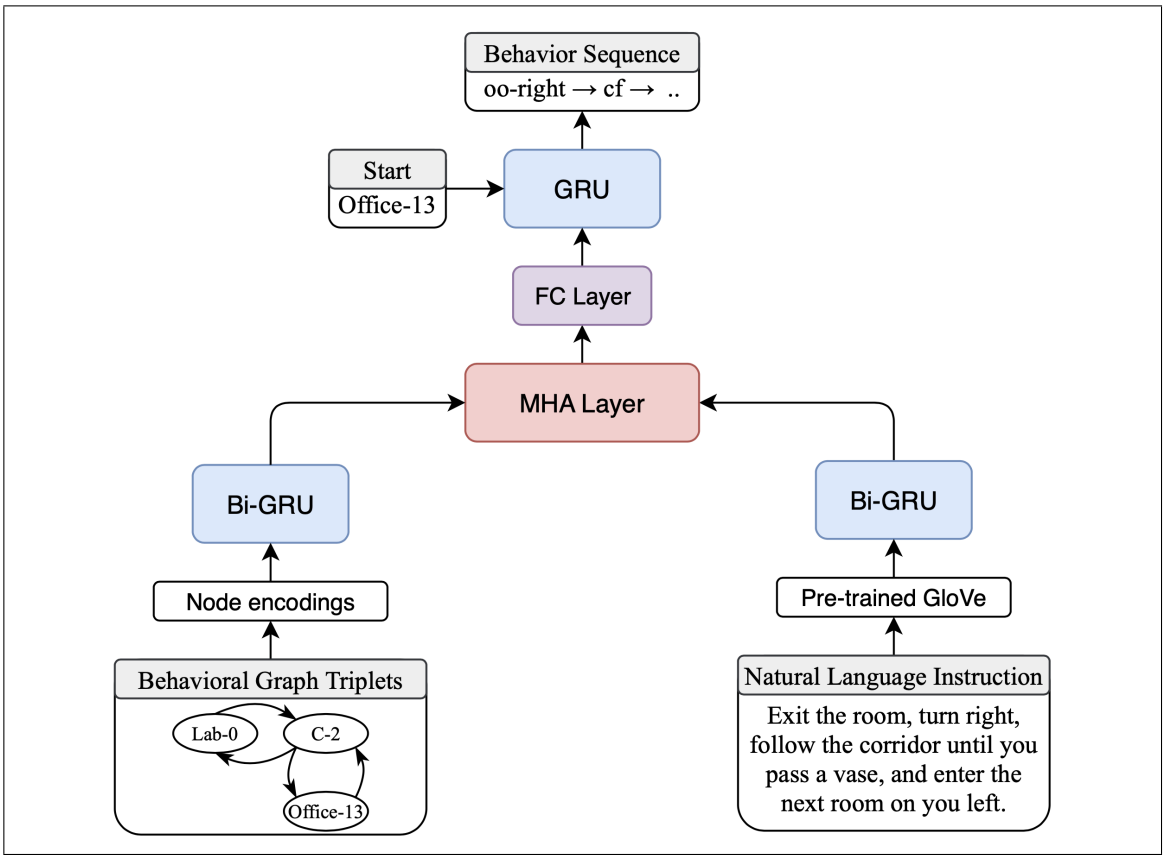

Figure 4.3. Multi-head attention architecture.

Attention vectors $R_i \in \mathbb{R}^{2H}$ are then defined as $R_i = \sum_{j=1}^{T} a_{ij} I_j$ and passed as input to the fully connected layer downstream as in the neural baseline architecture, concatenated with the original node encodings $\bar{G}$.

4.3.2. Graph attention networks

An alternative modification consists of using a graph attention network (GAT), which modifies the map descriptors by attending the environment topology with a module that was designed for this purpose in (Velickovic et al., 2018). Figure 4.4 illustrates the architecture of the network. In what remains of this document, we refer to this proposal as the "GAT" model variant.

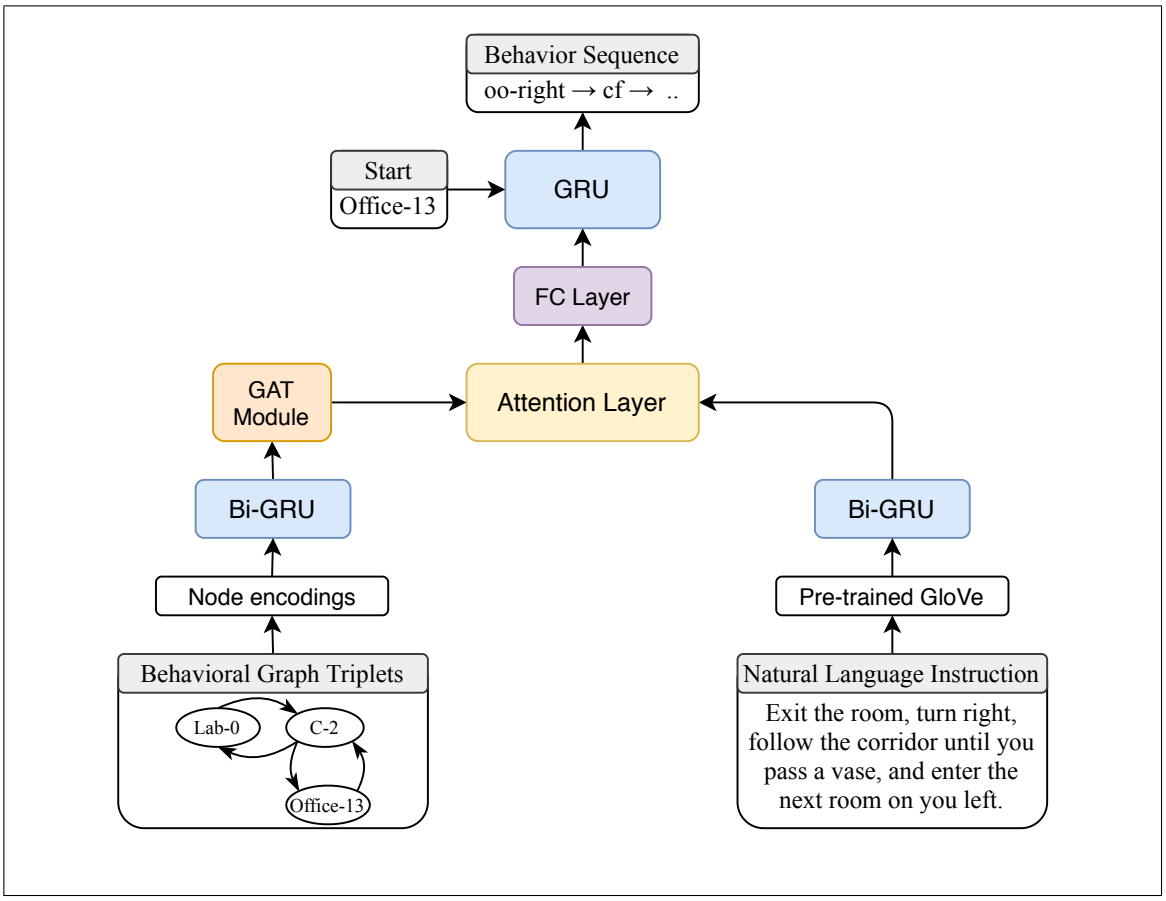

Figure 4.4. Graph attention network architecture.

The first important change with respect to the neural baseline is that the graph must be represented not as a list of triplets, but rather as a node set with an adjacency matrix. This implies that descriptors now work at node level, instead of triplet level. The one-hot encoding is similar to the previous one, with the difference that the size is modified from $(|2\mathcal{V}| + |\mathcal{E}|)$ to $(|\mathcal{V}| + |\mathcal{E}|)$ per each node, encoding the adjacency vector along with the node index.

In practice, the GAT layer receives the output from the graph biGRU encoding layer as a context matrix $C \in \mathbb{R}^{H \times L}$, and it generates a new node descriptor matrix $C' \in \mathbb{R}^{H' \times L}$ by fusing its own representation with its immediate neighbor descriptors[2].

[2] H' need not be equal to H, although our experiments show performance is better when this is the case.

Representations are blended through a weighted product learned by an internal MHA layer that handles a dynamic amount of neighbors, which means the GAT layer can process never-before-seen topologies.

4.3.3. Hybrid model

In our ablation analysis, we test a hybrid architecture that uses a GAT and a MHA module at the same time to understand the impact of both components working together. For a visual description, see Figure 4.5. In what remains of this document, we refer to this proposal as the "GAT-MHA" model variant.

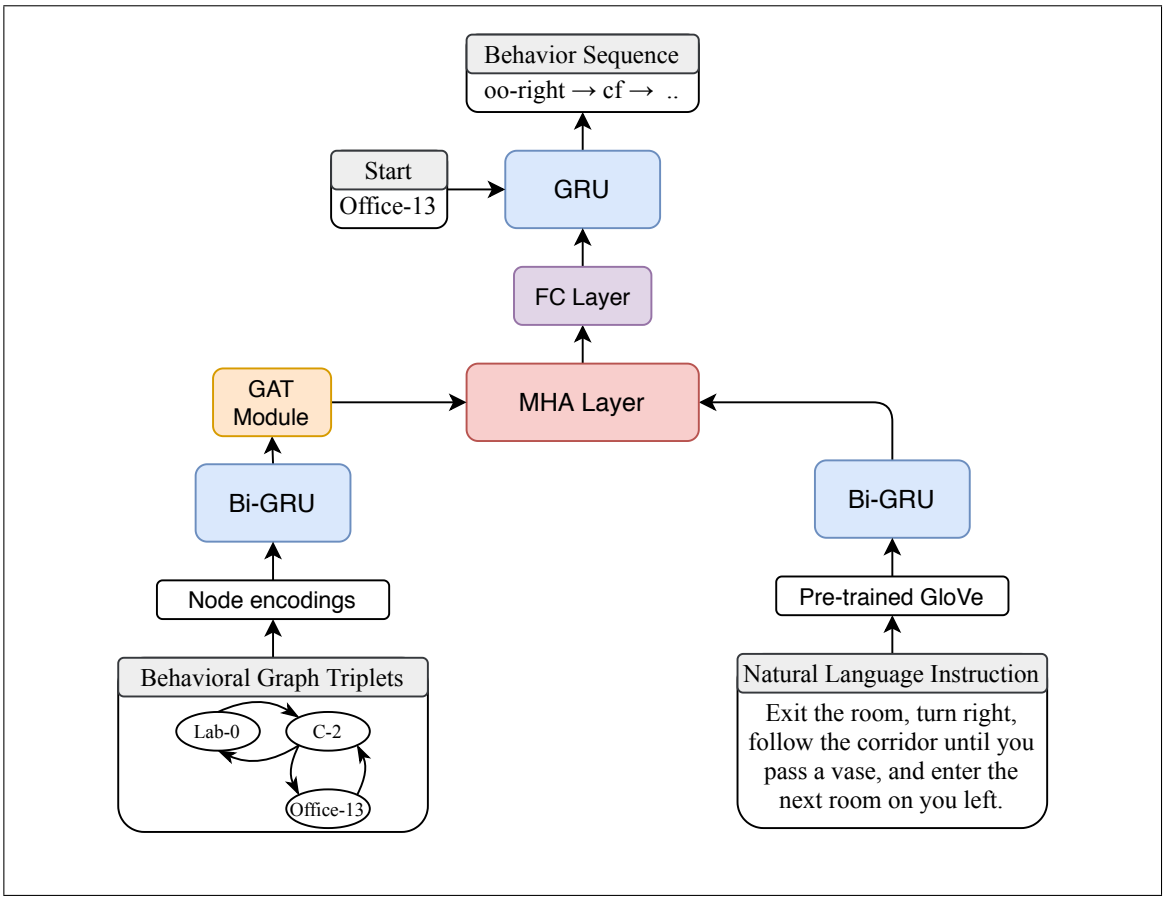

Figure 4.5. Graph attention network with multi-head attention architecture.

5. EXPERIMENTS AND DISCUSSION

5.1. Dataset

The dataset used for this work is the one introduced in (Zang, Pokle, et al., 2018), which consists of approximately ten thousand instructions in natural language along with the environment and path they correspond to. We now detail the data gathering process and other relevant aspects of the dataset, as described in the appendix of (Zang, Pokle, et al., 2018).

5.1.1. Data generation

One of the main contributions of (Zang, Pokle, et al., 2018) is the generation of a dataset for the problem, by crowd sourcing annotations using the Amazon Mechanical Turk platform (Crowston, 2012).

First, 100 distinct layouts of indoor environments were generated, with each room, hall, or corridor being correctly labeled. Each map has between 6 and 65 rooms. The types of rooms considered for the environments are: bathroom, office, kitchen, hall, corridor, laboratory, and room. There are 12 distinct behaviors to move between places, listed in Table 5.1.

Table 5.1. Navigation behaviors. Directions (dir) can be either left or right.

Symbol	Description
oo (dir)	Go out of the current place and turn (dir)
io (dir)	Turn (dir) and enter a new place
oio	Cross the corridor and enter the opposite room
(dir) t	Turn (dir)
sp	Enter a new place
cf	Follow the corridor
ch (dir)	Cross the hall and turn (dir)
chs	Cross the hall and keep straight

Landmarks were added to the map as additional information that the robot might use for navigation. The set of considered environmental landmarks is: chair, table, vase, clock, lamp, printer, computer, fridge, window, sofa, dustbin, bed, shoes, television, shelf, bike, bookshelf, sink, photo, and locker.

Second, pairs of start-goal points are generated and then connected to each other using standard planning techniques offered by the OMPL library (Sucan, Moll, & Kavraki, 2012). A random route subset was selected for humans to generate appropriate natural language instructions.

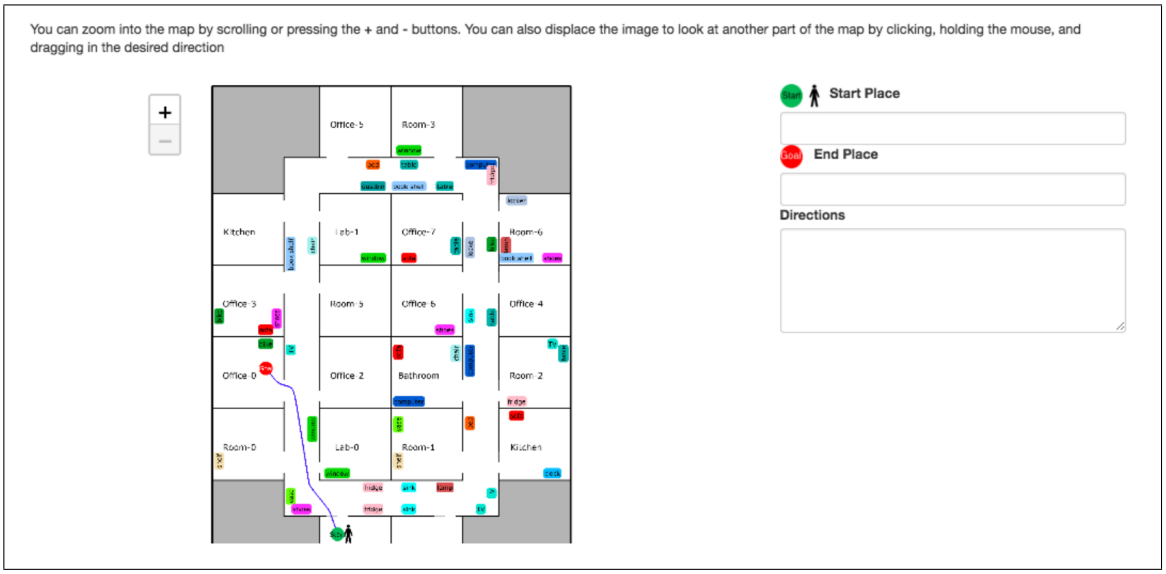

Figure 5.1. Amazon mechanical turk interface for dataset generation. Figure from Appendix A in (Zang, Pokle, et al., 2018).

Third, the directions in free-form unconstrained natural language are collected through the Amazon Mechanical Turk crowd sourcing platform, in which random human "workers" are exposed to an interface that shows the 2D map and a plan, and are subsequently asked to 1) identify the origin and destination of the route, and 2) describe free-form directions to another imaginary person, in sufficient detail to complete the path as needed.

A total of 822 workers generated route descriptions. A picture of the interface presented to workers can be seen in Figure 5.1, and sample map-route pairs are shown in Figure 5.2.

Finally, to ensure the quality of the gathered data, a protocol was implemented to verify the generated instructions. Two additional workers would be shown a route and instruction pair, and would confirm whether the origin-destination pair was correct and whether the provided instruction was unambiguous and if it matches the route. Mismatch reporting was performed to detect incorrect instructions, correct them and re-integrate them into the final dataset, which would only happen when the two workers verified and approved the route description.

Figure 5.2. From left to right, examples of small ($\leq$ 20 rooms), medium ($20 <$ rooms ≤ 40 rooms) and big (> 40 rooms) indoor environments, along with a sample route drawn in blue. Figure from Appendix A in (Zang, Pokle, et al., 2018).

5.1.2. Dataset split

The dataset is partitioned into three subsets for training, validation, and testing purposes. In particular, two distinct test sets are generated: the first one, Test-Repeated, consists of directions to be executed in indoor environments that the robotic agent has previously seen during training. On the other hand, Test-New considers directions only on maps that the agent has never seen before, which in theory constitutes a bigger challenge

for the generalization capabilities of the model. This is consistent with the results presented in (Zang, Pokle, et al., 2018), and further establishes diminishing the performance hit over novel maps as one of the main objectives of this work.

5.1.3. Dataset characteristics

Table 5.2 shows the main statistics of the dataset. We can see that graphs vary in size as measured by the triplet quantity in each set, ranging from 27 to 379. As the neural network requires a fixed tensor size, we set the maximum triplet set size to 300, truncating the few instances that exceed this limit[1]. Instructions are also diverse in the number of words, with concise examples of less than 10 words and long cases with upwards of 200 words, as well as the quantity of behaviors they describe. For the same reason as before, we set the maximum amount of instruction words to 150.

Table 5.2. Dataset statistics - summary of distribution of the number of triplets in graph, number of words in instructions, and number of behaviors in the predicted routes for training and testing (Test-New) dataset.

Data Statistics	**No. of Graph Triplets (Train)**	**No. of Graph Triplets (Test-New)**	**Instruction Length (Train)**	**Instruction Length (Test-New)**	**Behavior Number (Train)**	**Behavior Number (Test-New)**
Min	27	51	2	7	1	1
Max	379	379	239	145	22	19
Mean	176.5	209.32	33.71	30.99	7.15	7.07
Std Dev	72.02	104.5	17.95	16.94	3.67	3.67

The route length histogram is presented in Figure 5.3. The shortest paths have a single behavior executed, and the longest consists of 22 behaviors sequentially executed to navigate as the instruction requests. This, along with an average length of 7.1 ± 3.7 behaviors, implies the routes are not trivial to correctly interpret from potentially long route descriptions. However, short behavior sequences are available for the agent to use during the initial learning process.

[1]Only 6.4% and 5.4% of the maps exceed the limit in the training and validation set, respectively.

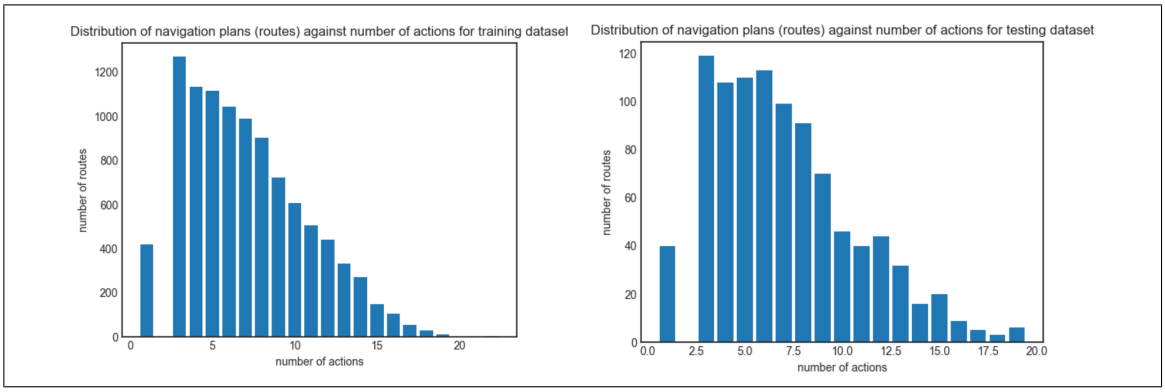

Figure 5.3. Distribution of route length in a) training set, b) testing set (both "new" and "repeated" splits). Figure from Appendix A in (Zang, Pokle, et al., 2018).

5.2. Evaluation metrics

To measure the performance of different approaches towards solving the problem, we consider six metrics: F_1 score, edit distance, match at 0, 1 and 2 moves away from ground truth, and goal match. We explain each of these metrics below. The input to the evaluation metrics are 1) a ground truth token sequence $gt = [b_1, ..., b_n]$ with b_i the i-th behavior to be executed by an oracle agent that perfectly interprets the directions, and 2) the predicted token sequence $pt = [b'_1, ..., b'_k]$, with b'_i the i-th behavior to be executed by the agent we are evaluating. Note the possibility of cases where the robot would arrive to the desired destination even when $gt \neq pt$ or $k \neq n$.

5.2.1. F_1 score

The F_1 score is the harmonic average between the precision p and the recall r of the model with respect to the predicted tokens pt and the ground truth tokens gt of the translated sequence. Let $c = pt \cap gt$ be the tokens present on both sequences. We can define $p = \frac{|c|}{|pt|}$ and $r = \frac{|c|}{|gt|}$. Then, the F_1 score is:

$$F_1 = \frac{2 \cdot p \cdot r}{p + r} \tag{5.1}$$

5.2.2. Edit distance

The edit distance (ED) is the number of atomic actions that the predicted sequence is away from becoming the ground truth sequence. An atomic action can be (1) deleting a single token, (2) adding a single token or (3) swapping a given token with another one. This is known as the Levenshtein distance in the linguistics and natural language processing literature (Levenshtein, 1966).

5.2.3. Match at k

The match at k metric, with $k \in \{0, 1, 2\}$, represents whether or not the edit distance is equal to or under k moves away for a prediction-truth sequence pair. If $k = 0$, we talk about an exact match. Ideally, we want the model to be able to exactly translate as many problem instances as possible.

5.2.4. Goal match

This metric indicates whether the predicted sequence arrives at the target node destination or not, even if the translated plan might not exactly match what the instructor meant. Failing an exact match, a goal match (GM) indicates that the translation model at least knows where it needs to go.

5.3. Setup and implementation details

All experiments were performed on a machine with a six-core 2.8 Ghz Intel Core i5-8400 CPU, 32 GB of RAM, and a Nvidia Geforce GTX1070 GPU with 8GB VRAM. The code was written using the PyCharm IDE, with git version control integration, on the Ubuntu 16.04 operating system.

All neural network models were implemented using the PyTorch 1.4 deep learning framework (Paszke et al., 2019). This includes a re-implementation of the neural baseline –originally written in TensorFlow 1.4– for reproducibility analysis[2]. The DGL 0.4.2 library was used to handle graph data structures (Wang et al., 2019). Agile development and faster prototyping was achieved by using the aforementioned stack of tools, which emphasized a "define-by-run" philosophy.

Define-by-run means a deep neural network backpropagation graph is defined just in time, and not on a previous compilation step as in TensorFlow (Abadi et al., 2015), for instance. Adhering to this philosophy results in shorter code and simpler forward passes when defining a model architecture.

Table 5.3 shows a comparison of the number of trainable parameters for each model. As for the hyperparameters, ADAM (Kingma & Ba, 2015) optimizer with a learning rate of $1e-3$ is used for all models, with a batch size of 256, four heads for the MHA mechanisms, hidden size of 128 for RNNs, and pre-trained GloVe word vectors of size 100.

Table 5.3. Millions of trainable parameters for each model.

Architecture	**Trainable parameters (millions)**
Zang Baseline (Ours)	0.785
MHA Model	1.011
GAT Model	0.975
GAT-MHA Model	1.266

All models were trained for 300 epochs. Normally, an early stopping mechanism would be used to avoid overfitting, meaning that once every n epochs we check the performance metrics on the validation dataset and stop training if the exact match metric diminishes with respect to the last checkpoint. To better understand the training dynamics of every variant we opt to train for the fixed amount of epochs, and at testing time we

[2]Special thanks to Gabriel Sepúlveda.

select the best performer as indicated by the validation exact match rate that is checked every $n = 10$ epochs.

We include a teacher forcing (Bengio, Vinyals, Jaitly, & Shazeer, 2015) scheme in the training loop for improved stability, where we replace a fraction f of the translation tokens emitted by the model with ground truth token at training time. We initially set $f = 0.5$ with a linear decrease until $f = 0$ is reached at 100 epochs.

Both baseline performance metrics are considered as reported in their respective documents. We also provide and report our own implementation of the (Zang, Pokle, et al., 2018) baseline for reproducibility purposes: ideally, the same performance would be observed.

5.4. Main results

The main results of this work are presented in Tables 5.4 and 5.5, where all proposed methods are compared against the baselines in both test datasets.

Table 5.4. Main results for Test-Repeated dataset. The symbol ↑ indicates that higher results are better in the corresponding column; ↓ indicates that lower results are better.

Architecture	**Metric (Test Repeated)**					
	↑ F1	↑ M@0	↑ M@1	↑ M@2	↑ GM	↓ ED
Shimizu Baseline	79.83	25.30	-	-	26.28	2.53
Zang Baseline	**93.54**	**61.17**	**83.30**	**92.19**	**61.36**	**0.75**
Zang Baseline (Ours)	91.67	44.43	76.93	89.16	44.43	1.01
MHA Model	93.07	55.96	81.31	90.16	55.96	0.84
GAT Model	91.87	48.41	78.53	89.66	48.41	0.94
GAT-MHA Model	90.67	42.45	72.46	86.58	42.45	1.14

In the Test-Repeated dataset, the best performer –as determined by the exact match rate– is the neural baseline introduced in (Zang, Pokle, et al., 2018), although it should be noted that our own implementation of the model is not able to reproduce the reported

performance in this set, as seen in the second and third rows of Table 5.4. The MHA model is able to improve upon the latter, and beats all other proposed variants. The GAT module also helps for the task when compared to our implementation of the neural baseline, although by a smaller margin than MHA. Surprisingly, the hybrid GAT-MHA variant exhibits a performance inferior to the the simpler alternatives.

Table 5.5. Main results for Test-New dataset. The symbol ↑ indicates that higher results are better in the corresponding column; ↓ indicates that lower results are better.

Architecture	**Metric (Test New)**					
	↑ F1	↑ M@0	↑ M@1	↑ M@2	↑ GM	↓ ED
Shimizu Baseline	81.38	25.44	-	-	25.44	2.39
Zang Baseline	90.22	41.71	69.82	82.08	41.81	1.22
Zang Baseline (Ours)	90.89	43.41	72.64	87.25	43.41	1.09
MHA Model	**92.57**	**51.40**	**79.06**	89.43	**51.40**	**0.91**
GAT Model	91.77	47.36	77.51	**90.26**	47.36	0.96
GAT-MHA Model	90.43	42.28	72.95	85.08	42.28	1.18

In the Test-New split, we can observe that the best performer for unseen environments is the newly introduced MHA model, outperforming (Zang, Pokle, et al., 2018) and thus obtaining state of the art performance (Cerda-Mardini, Araujo, & Soto, 2020). Note that in this case our own implementation does match (and even slightly improve upon) the reported results, which stands in contrast with the test-repeated results. The GAT module presents a similar behavior as in test-repeated: it does contribute to the model learning capacity, but not as much as the MHA mechanism. Once again, the hybrid model fails to perform at an acceptable level when compared to the rest of the neural architectures.

In Figure 5.4, training procedures for the reported models are presented. In general, models converged at around 100 epochs, which for our test environment happened after 6 hours of training time. In the case of GAT variants, batching the graph data structures imposes an additional CPU load that increases training time by a factor of 2.1 (an average of 109 versus 227 seconds per epoch).

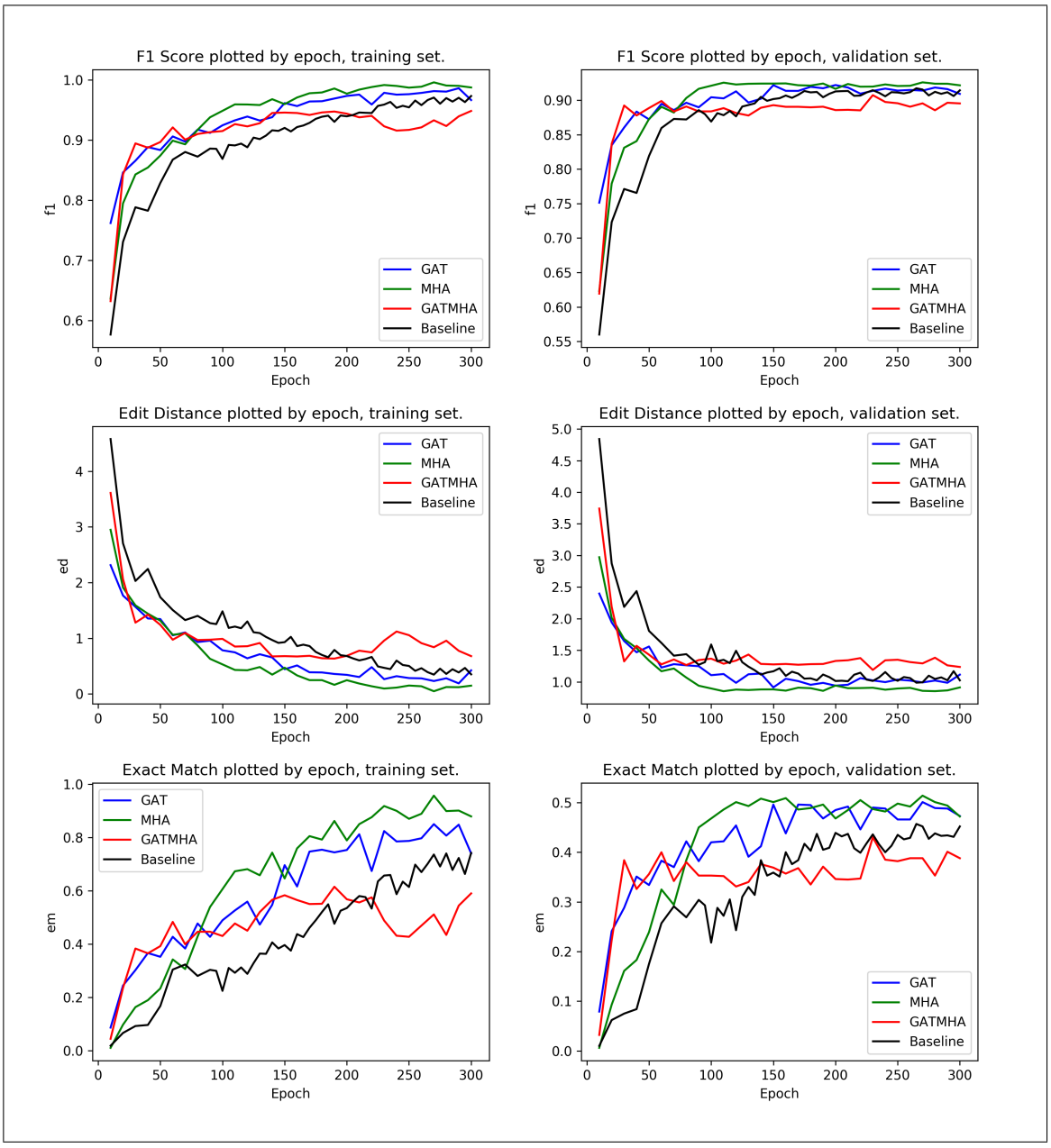

Figure 5.4. Metric evolution throughout the training process.

A noteworthy observation is that, in our experiments, the goal match metric always equals the exact match metric, which indicates that our variants are averse to completing a route by following a path other than the one intended in the instructions. It is not clear

what could be the cause for this behavior, specially when compared to the baselines that do present a slightly higher goal match rate than they do an exact match rate.

5.5. Discussion

In what follows, we will further analyze our results, and detail the behavior of each model variant in more depth to gain insights as to what aspects of the problem each model is able to solve.

5.5.1. Multi-head attention

As seen in the main results, the MHA model performs better than every other variant in the test-new partition, setting the state of the art performance with a 23.23% improvement over the neural baseline, although at the cost of a 8.52% decrease in the test-repeated partition, and a 28.79% increase in the number of trainable parameters.

To get a better intuition as to how multiple heads can help to generate different attention alignments, Figures 5.5 and 5.6 show a case where each head is blending the multimodal information in a different relevant pattern for translating the instruction "`Go out of office 3 and turn left past the clock on the left. Enter the kitchen on your left.`" to the corresponding translation ["O-3", "ool", "cf", "iol"].

In Figure 5.5, it is shown that head #1 focuses on the relation between corridor C-1 and kitchen K-2 (which is the correct triplet the robot needs to execute as the last behavior) along with a relevant part of the instructions that precedes a kitchen mention, while head #2 is not particularly useful in this case.

However, Figure 5.6 shows how heads #3 and #4 align very similar sub-segments of the instruction in a different way. Head #3 aligns "`past the clock on the left`"

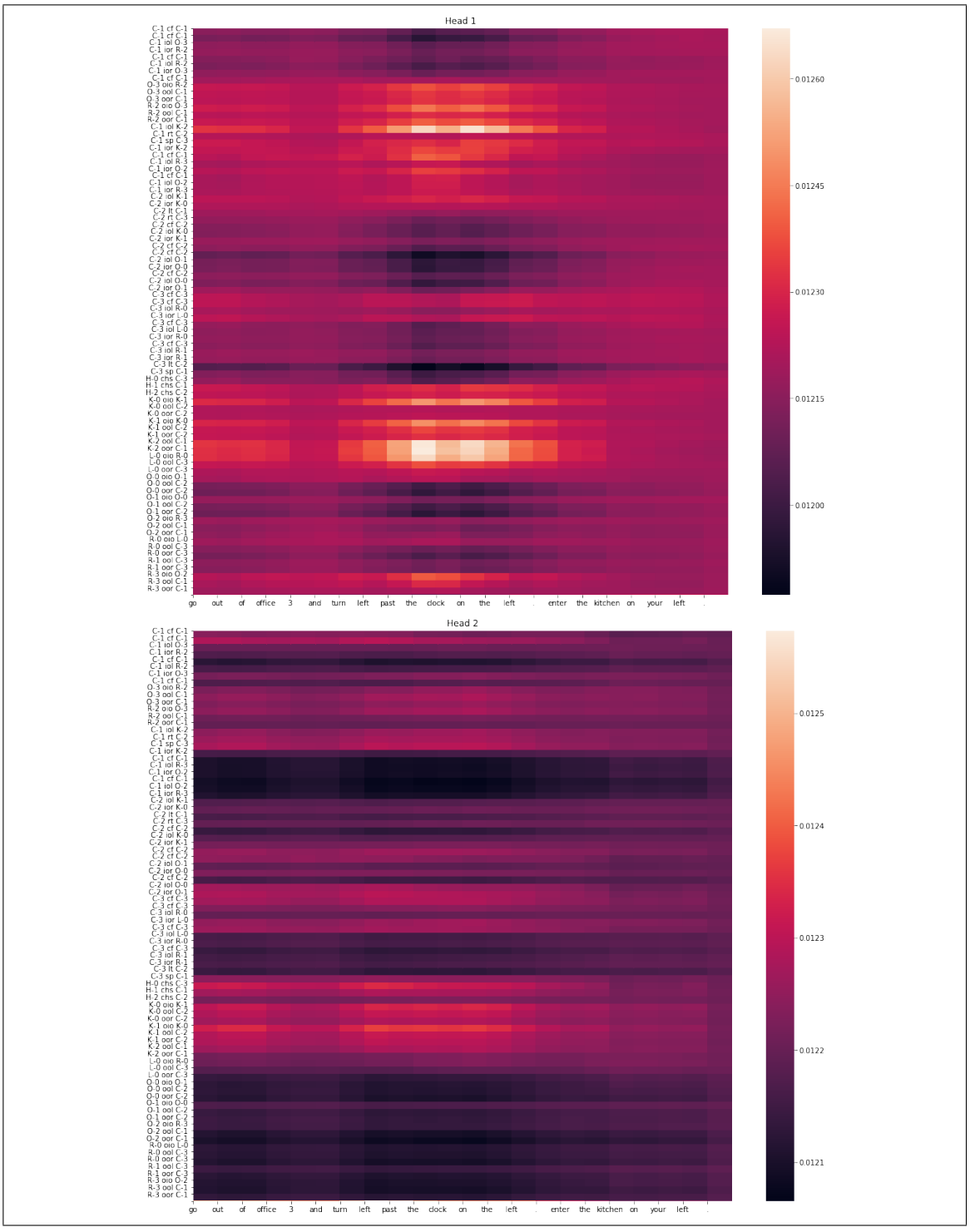

Figure 5.5. Example attention maps of heads 1 & 2. Head 1 correctly attends to nodes C-1 and K-2, while head 2 does not show a useful alignment.

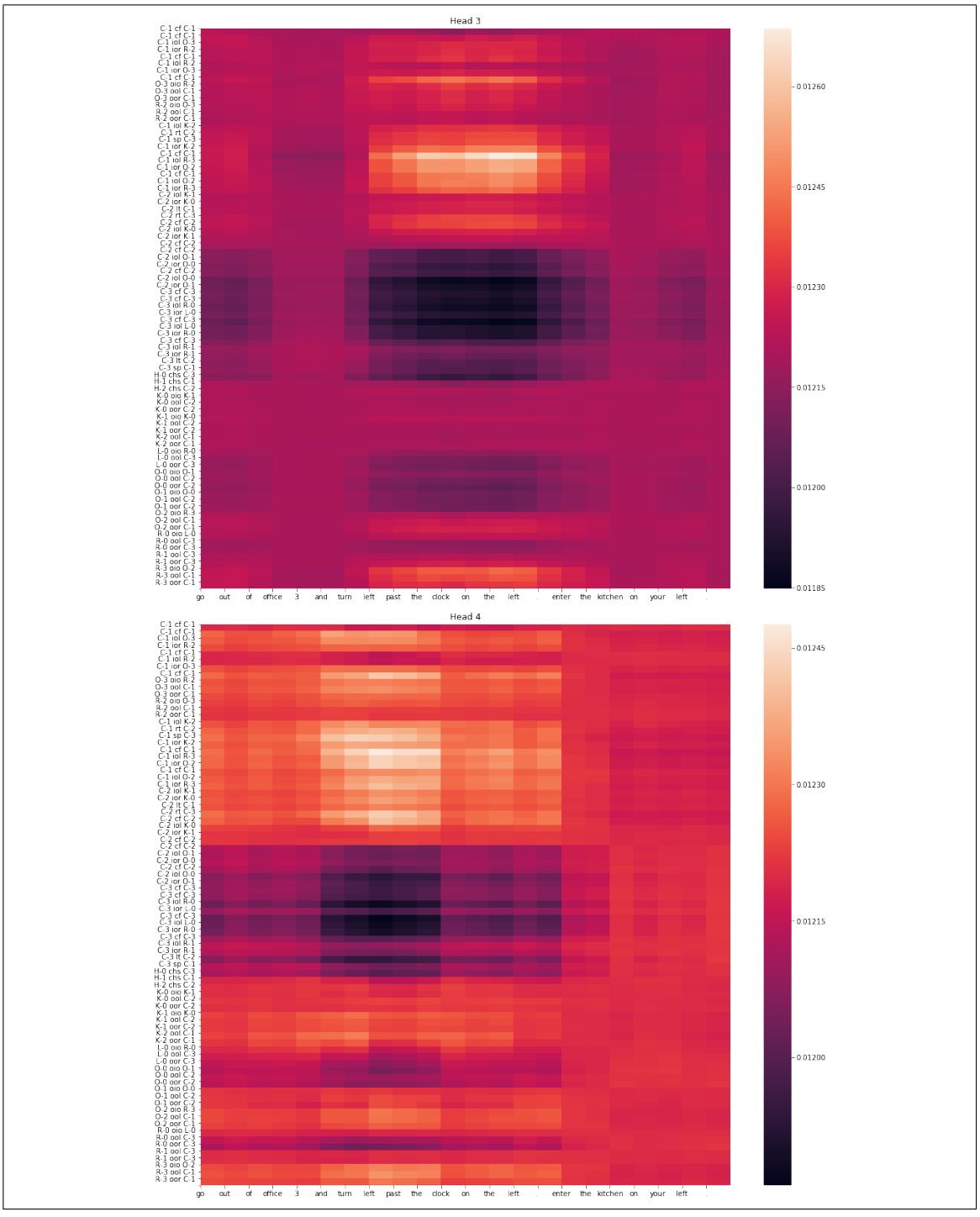

Figure 5.6. Example attention maps of heads 3 & 4. Similar parts of the instruction show different alignments in each case, which would be less likely to happen with a single headed scheme.

with two triplets that describe the corridor C-1, where one of the visible attributes corresponds to the clock referenced in the instruction. Head #4 reinforces alignment of "`and turn left past the clock`" to these triplets, but also correlates this segment to the "iol" and "ior" (turn and enter) behaviors originating from C-1, which is the correct node where the robot would be at this point, highlighting the potential places where it could wrongly enter when following the instructions. A visualization of relevant map areas for heads #1, #3 and #4 is presented in figure 5.7.

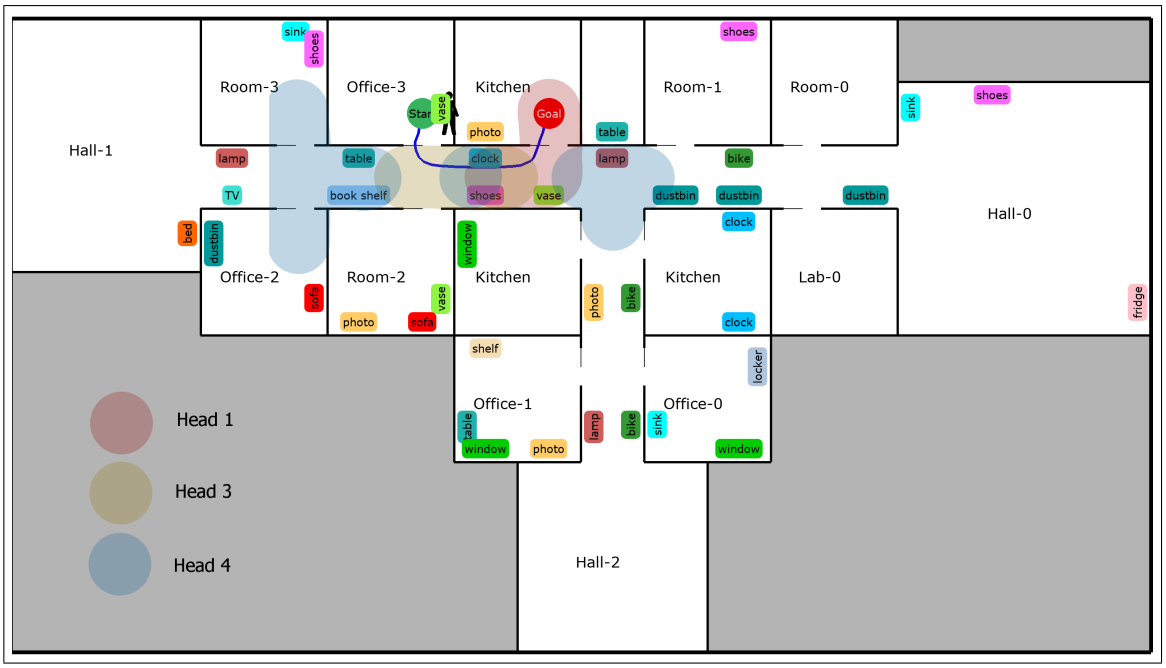

Figure 5.7. Example visualization of most relevant map areas for heads #1, #3 and #4. The model correctly translated the instructions to the path marked by the blue line.

In Table 5.6, we present the results of an ablation test where we train MHA models with 1, 2, 4 and 8 heads. The single-headed case corresponds to our implementation of the neural baseline. There is a clear benefit from using more heads, but the limit on what this mechanism can leverage out of the data happens with $k = 4$ heads, showing diminishing returns after this.

Table 5.6. Ablation analysis for number of attentional heads. Optimal performance is achieved at $k = 4$.

#	Test Repeated						Test New					
Heads	**F1**	**M@0**	**M@1**	**M@2**	**GM**	**ED**	**F1**	**M@0**	**M@1**	**M@2**	**GM**	**ED**
1	91.67	44.43	76.93	89.16	44.43	1.01	90.89	43.41	72.64	87.25	43.41	1.09
2	90.94	49.70	74.45	86.18	49.70	1.08	90.30	43.62	72.95	86.73	43.62	1.13
4	**93.07**	**55.96**	**81.31**	**90.16**	**55.96**	**0.84**	**92.57**	**51.40**	**79.06**	**89.43**	**51.40**	**0.91**
8	90.44	43.44	74.35	87.57	43.44	1.09	89.78	38.14	71.61	85.18	38.14	1.24

Summarizing, we can conclude that a more specialized blending of multi-modal information is valuable for building a better translator, in particular, for unseen environments.

5.5.2. Graph attention networks

The GAT architecture performs slightly better than the neural baseline for new environments, but still below what MHA achieves.

An ablation analysis in Table 5.7 shows results of the GAT layer working with graph context representations at different stages of refinement: immediately after the encoding layer ("GAT-Enc", reported in main results as "GAT"), after the attention layer ("GAT-Attn"), and after the dimensionality reduction FC layer ("GAT-FC"). In this case, results indicate that the best performing descriptors are upstream, suggesting that topology information is more valuable when added earlier into the graph representation.

Table 5.7. Ablation analysis for graph attention network layer positioning.

	Test Repeated						Test New					
Model	F1	M@0	M@1	M@2	GM	ED	F1	M@0	M@1	M@2	GM	ED
GAT-Enc	**91.86**	**48.41**	**78.53**	**89.66**	**48.41**	**0.94**	91.77	**47.36**	**77.51**	**90.26**	**47.36**	**0.96**
GAT-Attn	90.94	43.04	75.15	87.38	43.04	1.11	**91.78**	44.15	73.37	88.08	44.15	1.06
GAT-FC	87.81	40.56	69.28	83.00	40.56	1.32	88.68	41.45	70.88	84.04	41.45	1.31

In Tables 5.4 and 5.5, an ablation test can be established between two variants: one where the attention mechanism is a multi-headed one (GAT-MHA variant), and one where the mechanism is as presented in the neural baseline (GAT variant). The results indicate

that the GAT and MHA modules, as defined, are not able to jointly perform at an acceptable level, although the network as a whole seems to achieve a faster convergence, as shown in Figure 5.4.

To improve the low performance of the hybrid model, we test an additional variant ("GAT-MHA FC") where the GAT module operates after the FC layer, instead of operating after the biGRU graph encoder. Results in Table 5.8 suggest that it is possible to further optimize the hybrid approach architecture, although the performance is still inferior to that of the MHA model.

Table 5.8. Graph attention network module position analysis for hybrid model.

	Test Repeated						Test New					
Model	F1	M@0	M@1	M@2	GM	ED	F1	M@0	M@1	M@2	GM	ED
GAT MHA	90.67	42.45	72.46	86.58	42.45	1.14	90.43	42.28	72.95	85.08	42.28	1.18
GAT MHA FC	**92.07**	**53.88**	**80.22**	**87.87**	**53.88**	**0.93**	**92.01**	**51.30**	**78.65**	**87.88**	**51.30**	**0.99**

5.5.3. General remarks

We can think of the GAT and MHA variants as incremental changes with respect to the neural baseline, each adding or replacing one single new component in the original architecture. The GAT-MHA variant mixes both approaches into the same architecture. By comparing the performance between them, our analysis suggests that:

(i) The MHA mechanism is highly useful for the task.

(ii) The GAT mechanism does help, but in a smaller measure.

(iii) Using both mechanisms in the same architecture does not translate into an improved model. In fact, the standard hybrid model performs worse than both simpler counterparts.

Taking into consideration the positive impact that the MHA mechanism has, a good question for motivating future work would be whether a transformer architecture (Vaswani et al., 2017) could further increase performance.

Summarizing, our results show that multi-headed attention does lead to improved performance in previously unseen environments. Graph attention networks also have a net positive impact, but not as notorious. Our implementation of the neural baseline failed to perform as reported in (Zang, Pokle, et al., 2018) for seen environments, which suggests that the MHA variant would be superior for Test-Repeated instances as well, if used in the original implementation.

Lastly, and rather disappointingly, the hybrid GAT-MHA model results indicate that both techniques are not immediately compatible with each other, as shown by not being able to surpass the neural baseline performance in new maps, even when the standalone variants manage to do so on their own. Future efforts could explore alternative ways in which to use both layers at the same time to obtain a performance increase, in the spirit of the analysis presented in Table 5.8.

6. CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

In this work we have introduced mechanisms to improve the performance of a deep neural network that translates natural language directions to a high-level sequential language understandable by a robotic agent.

Both mechanisms, multi-head attention and graph attention networks, aim to achieve better translations by refining the alignment between the instruction and the knowledge about the environment that the robot has to navigate.

Results show that our modifications are not equally well suited to contribute to the problem. In particular, multi-head attention mechanisms can achieve state-of-the-art performance for unseen environments, improving the generalization capabilities of the model. On the other hand, graph attention networks do help but their impact is not as noticeable, even when combined with a multi-head attention module.

The small impact of graph attention networks might be partially explained because only immediate neighbors are considered for refining the descriptors of each node. As the average path length is of approximately eight nodes, the GAT module would require a span of seven hops for the start and goal nodes to affect the descriptor of each other.

In other words, one single hop might not provide enough topological information to significantly improve the node descriptors before being aligned in the attentional layer. In fact, considering the results of the GAT-MHA variant, a $1-$hop GAT module is detrimental to the MHA mechanism.

In a more general sense, the results in section 5.5.2 suggest that any node descriptor refinement based on the map topology should be made after language alignment is done.

This makes sense, as in a translation task the bulk of the information is in the source sentence itself.

To answer the questions posed in our main hypothesis, we can say that:

(i) It is indeed possible to develop language models that perform better in this task than previous efforts, even if only for the Test-New dataset partition. In fact, we argue that Test-New is a more useful partition than Test-Repeated, because a robotic agent that uses our translation model would probably be trained once and then deployed into the real world, to navigate environments that are not the same as those in the training dataset.

(ii) Yes, a multi-headed attention mechanism is able to produce better multi-modal alignments than the single-headed equivalent layer, although at the cost of an increased number of trainable parameters.

(iii) Indeed, graph attention networks can slightly improve the generalization capabilities of the agent. Our results and analysis suggest that GAT modules should refine node descriptors only after aligning them with the instruction.

Considering our findings, we conclude this work contributes towards the pursuit of intelligent agents that are capable of interacting with humans by understanding natural language directions to navigate through indoor environments.

6.2. Future work

There are various research lines that constitute promising avenues for future work in this problem:

(i) Studying the effect of the neighborhood size (or, similarly, the number of hops), in a GAT-MHA FC architecture. It is possible that further performance improvements could be achieved this way, and it would confirm the hypothesis outlined

in our conclusions as a cause for the disappointing performance of the GAT-MHA model variant.

(ii) Using multi-head attention in the decoder that outputs instruction tokens while attending to the context vector that merges both sources of information. In this sense, a natural candidate is the decoder proposed in (Vaswani et al., 2017).

(iii) It could also prove interesting to try an end-to-end generic Transformer model to contrast its performance against our architecture, which was designed specifically for this task.

(iv) Using state-of-the-art pre-trained contextual word embeddings such as BERT (Devlin, Chang, Lee, & Toutanova, 2019), which should help to cope with both polysemy and ambiguity throughout the instructions. These improved descriptors could, in turn, help the attentional layer to produce a better context vector for the decoder.

(v) Lastly, a valuable extension to this work would be to implement the translator in a physical robot, to understand the operational constraints and limitations of our work in the real world.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from https://www.tensorflow.org/

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., ... van den Hengel, A. (2018). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3674-3683.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., & Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. *CoRR, abs/2003.13350*. Retrieved from https://arxiv.org/abs/2003.13350

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Retrieved from http://arxiv.org/abs/1409.0473

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (pp. 1171–1179). Retrieved from http://papers.nips.cc/paper/5956-scheduled-sampling-for-sequence-prediction-with-recurrent-neural-networks

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... Zieba, K. (2016). End to end learning for self-driving cars. *CoRR*, *abs/1604.07316*. Retrieved from http://arxiv.org/abs/1604.07316

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. *CoRR*, *abs/2005.14165*. Retrieved from https://arxiv.org/abs/2005.14165

Cerda-Mardini, P., Araujo, V., & Soto, Á. (2020). Translating Natural Language Instructions for Behavioral Robot Navigation with a Multi-Head Attention Mechanism. In *Proceedings of the The Fourth Widening Natural Language Processing Workshop* (pp. 96–98). Seattle, USA: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/2020.winlp-1.24

Chang, A. X., Dai, A., Funkhouser, T. A., Halber, M., Nießner, M., Savva, M., ... Zhang, Y. (2017). Matterport3d: Learning from RGB-D data in indoor environments. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017* (pp. 667–676). IEEE Computer Society. Retrieved from https://doi.org/10.1109/3DV.2017.00081

Chung, J., Gülçehre, Ç., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*. Retrieved from http://arxiv.org/abs/1412.3555

Crowston, K. (2012). Amazon mechanical turk: A research tool for organizations and information systems scholars. In A. Bhattacherjee & B. Fitzgerald (Eds.), *Shaping the Future of ICT Research. Methods and Approaches - IFIP WG 8.2, Working Conference, Tampa, FL, USA, December 13-14, 2012. Proceedings* (Vol. 389, pp. 210–221). Springer.

Retrieved from https://doi.org/10.1007/978-3-642-35142-6_14

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Association for Computational Linguistics. Retrieved from https://doi.org/10.18653/v1/n19-1423

Enright, J., & Wurman, P. R. (2011). Optimization and coordinated autonomy in mobile fulfillment systems. In *Automated Action Planning for Autonomous Mobile Robots, Papers from the 2011 AAAI Workshop, San Francisco, California, USA, August 7, 2011* (Vol. WS-11-09). AAAI. Retrieved from http://www.aaai.org/ocs/index.php/WS/AAAIW11/paper/view/3917

Faust, A., Oslund, K., Ramirez, O., Francis, A. G., Tapia, L., Fiser, M., & Davidson, J. (2018). PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018* (pp. 5113–5120). IEEE. Retrieved from https://doi.org/10.1109/ICRA.2018.8461096

Goodfellow, I. J., Bengio, Y., & Courville, A. C. (2016). *Deep learning*. MIT Press. Retrieved from http://www.deeplearningbook.org/

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive Representation

Learning on Large Graphs. In I. Guyon et al. (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA* (pp. 1024–1034). Retrieved from http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs

Hatori, J., Kikuchi, Y., Kobayashi, S., Takahashi, K., Tsuboi, Y., Unno, Y., ... Tan, J. (2018). Interactively Picking Real-World Objects with Unconstrained Spoken Language Instructions. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018* (pp. 3774–3781). IEEE. Retrieved from https://doi.org/10.1109/ICRA.2018.8460699

Henderson, J. (2020). The unstoppable rise of computational linguistics in deep learning. In D. Jurafsky, J. Chai, N. Schluter, & J. R. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020* (pp. 6294–6306). Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/2020.acl-main.561/

Hinton, G. E., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., ... Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*, 82-97. Retrieved from https://doi.org/10.1109/MSP.2012.2205597

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780. Retrieved from https://doi.org/10.1162/neco.1997.9.8.1735

Jain, V., Magalhães, G., Ku, A., Vaswani, A., Ie, E., & Baldridge, J. (2019). Stay

on the Path: Instruction Fidelity in Vision-and-Language Navigation. In A. Korhonen, D. R. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers* (pp. 1862–1872). Association for Computational Linguistics. Retrieved from https://doi.org/10.18653/v1/p19-1181

Kaggle. (2014). *Higgs boson machine learning challenge.* Retrieved from https://www.kaggle.com/c/higgs-boson

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Retrieved from http://arxiv.org/abs/1412.6980

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.* Retrieved from https://openreview.net/forum?id=SJU4ayYgl

Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I. D., Rezatofighi, H., & Savarese, S. (2019). Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada* (pp. 137–146). Retrieved from http://papers.nips.cc/paper/8308-social-bigat-multimodal-trajectory-forecasting-using-bicycle-gan-and-graph-attention-networks

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States* (pp. 1106–1114). Retrieved from http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

Ku, A., Anderson, P., Patel, R., Ie, E., & Baldridge, J. (2020). Room-Across-Room: Multilingual Vision-and-Language Navigation with Dense Spatiotemporal Grounding. *CoRR*, *abs/2010.07954*. Retrieved from https://arxiv.org/abs/2010.07954

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. Retrieved from https://openreview.net/forum?id=H1eA7AEtvS

LeCun, Y., Bengio, Y., & Hinton, G. (2015, 05). Deep learning. *Nature*, *521*(7553), 436–444. Retrieved from https://doi.org/10.1038/nature14539

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics. Doklady*, *10*, 707-710.

Majumdar, A., Shrivastava, A., Lee, S., Anderson, P., Parikh, D., & Batra, D. (2020). Improving vision-and-language navigation with image-text pairs from the web. *CoRR*, *abs/2004.14973*. Retrieved from https://arxiv.org/abs/2004.14973

Matuszek, C., Fox, D., & Koscher, K. (2010). Following directions using statistical machine translation. In P. J. Hinds, H. Ishiguro, T. Kanda, & P. H. K. Jr. (Eds.), *Proceedings of the 5th ACM/IEEE International Conference on Human Robot Interaction, HRI 2010, Osaka, Japan, March 2-5, 2010* (pp. 251–258). ACM. Retrieved from https://doi.org/10.1145/1734454.1734552

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *ArXiv*, *abs/1310.4546*.

Mitchell, T. M., Carbonell, J. G., & Michalski, R. S. (Eds.). (1986). *Machine learning: A guide to current research.* USA: Kluwer Academic Publishers.

Nielsen, M. A. (2015). *Neural networks and deep learning*. Determination Press. Retrieved from http://neuralnetworksanddeeplearning.com

Okuta, R., et al. (2018). *Preferred networks project blog.* Preferred Networks Inc. Retrieved from https://projects.preferred.jp/tidying-up-robot/

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada* (pp. 8024–8035). Retrieved from http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word

Representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL* (pp. 1532–1543). ACL. Retrieved from https://doi.org/10.3115/v1/d14-1162

Qi, Y., Wu, Q., Anderson, P., Wang, X., Wang, W. Y., Shen, C., & van den Hengel, A. (2020). REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020* (pp. 9979–9988). IEEE. Retrieved from https://doi.org/10.1109/CVPR42600.2020.01000

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65 6*, 386-408.

Rumelhart, D., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533-536.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80. Retrieved from https://doi.org/10.1109/TNN.2008.2005605

Sepulveda, G., Niebles, J. C., & Soto, A. (2018). A Deep Learning Based Behavioral Approach to Indoor Autonomous Navigation. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018* (pp. 4646–4653). IEEE. Retrieved from https://doi.org/10.1109/ICRA.2018.8460646

Shimizu, N., & Haas, A. R. (2009). Learning to Follow Navigational Route Instructions. In C. Boutilier (Ed.), *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009* (pp. 1488–1493). Retrieved from http://ijcai.org/Proceedings/09/Papers/249.pdf

Shrestha, A., Pugdeethosapol, K., Fang, H., & Qiu, Q. (2020). High-Level Plan for Behavioral Robot Navigation with Natural Language Directions and R-NET. *CoRR*, *abs/2001.02330*. Retrieved from http://arxiv.org/abs/2001.02330

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., ... Fox, D. (2020). ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020* (pp. 10737–10746). IEEE. Retrieved from https://doi.org/10.1109/CVPR42600.2020.01075

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., ... Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484-489. Retrieved from https://doi.org/10.1038/nature16961

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, *15*(1), 1929–1958. Retrieved from http://dl.acm.org/citation.cfm?id=2670313

Sucan, I. A., Moll, M., & Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics Automation Magazine*, *19*, 72–82. Retrieved from https://doi.org/10.1109/MRA.2012.2205651

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (pp. 3104–3112). Retrieved from http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks

van Harmelen, F., Lifschitz, V., & Porter, B. W. (2008). Handbook of knowledge representation. In *Handbook of knowledge representation* (Vol. 3, chap. 24). Elsevier. Retrieved from http://www.sciencedirect.com/science/bookseries/15746526/3

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention Is All You Need. In I. Guyon et al. (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA* (pp. 5998–6008). Retrieved from http://papers.nips.cc/paper/7181-attention-is-all-you-need

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. Retrieved from https://openreview.net/forum?id=rJXMpikCZ

Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., ... Zhang, Z. (2019). Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. *CoRR*, *abs/1909.01315*. Retrieved from http://arxiv.org/abs/1909.01315

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2019). A Comprehensive Survey on Graph Neural Networks. *CoRR*, *abs/1901.00596*. Retrieved from http://arxiv.org/abs/1901.00596

Zang, X., Pokle, A., Vázquez, M., Chen, K., Niebles, J. C., Soto, A., & Savarese, S. (2018). Translating Navigation Instructions in Natural Language to a High-Level Plan for Behavioral Robot Navigation. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018* (pp. 2657–2666). Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/D18-1286

Zang, X., Vázquez, M., Niebles, J. C., Soto, A., & Savarese, S. (2018). Behavioral Indoor Navigation With Natural Language Directions. In T. Kanda, S. Sabanovic, G. Hoffman, & A. Tapus (Eds.), *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, HRI 2018, Chicago, IL, USA, March 05-08, 2018* (pp. 283–284). ACM. Retrieved from https://doi.org/10.1145/3173386.3177001

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2020). *Dive into deep learning*. Retrieved from https://d2l.ai