Instituto Tecnológico de Costa Rica

Mechatronics Engineering Department

Licentiate Degree Program in Mechatronics Engineering

Learning semantic representations through multimodal graph neural networks

Final report submitted in partial fulfillment of the requirements for the degree of

Licentiate in Mechatronics Engineering

Jose Manuel Lainer Alfaro

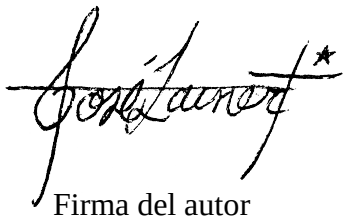
Cartago, June 21, 2021

Declaración de Autenticidad

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.



Cartago, 21 de junio de 2021

Firma del autor

Jose Manuel Lainer Alfaro

Céd: 2-0756-0239

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

El profesor asesor del presente trabajo final de graduación, indica que el documento presentado por el estudiante cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica del Instituto Tecnológico de Costa Rica para ser defendido ante el jurado evaluador, como requisito final para aprobar el curso Proyecto Final de Graduación y optar así por el título de Ingeniero(a) en Mecatrónica, con el grado académico de Licenciatura.

Estudiante: José Manuel Lainer Alfaro

Proyecto: Learning semantic representations through multimodal graph neural networks.

JUAN LUIS CRESPO MARIÑO (FIRMA)

Firmado digitalmente por JUAN LUIS CRESPO MARIÑO (FIRMA)
Fecha: 2021.06.14 15:48:19 -06'00'

Dr. -Ing. Juan Luis Crespo Mariño

Asesor

Cartago, 21 de junio 2021

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

Proyecto final de graduación defendido ante el presente jurado evaluador como requisito para optar por el título de Ingeniero(a) en Mecatrónica con el grado académico de Licenciatura, según lo establecido por el programa de Licenciatura en Ingeniería Mecatrónica, del Instituto Tecnológico de Costa Rica.

Estudiante: José Manuel Lainer Alfaro

Proyecto: Learning semantic representations through multimodal graph neural networks.

Miembros del jurado evaluador

JOHANNA VANESSA MUÑOZ PEREZ (FIRMA)
Firmado digitalmente por JOHANNA VANESSA MUÑOZ PEREZ (FIRMA)
Fecha: 2021.06.21 14:11:38 -06'00'

MSc. -Ing. Johanna Muñoz Pérez

Jurado

IVAN ARAYA MENESES (FIRMA)
Firmado digitalmente por IVAN ARAYA MENESES (FIRMA)
Fecha: 2021.06.21 21:12:16 -06'00'

MSc. -Ing. Iván Araya Meneses

Jurado

FELIPE GERARDO MEZA OBANDO (FIRMA)
Digitally signed by FELIPE GERARDO MEZA OBANDO (FIRMA)
Date: 2021.06.22 08:11:10 -06'00'

MSc. -Ing. Felipe Meza Obando

Jurado

Los miembros de este jurado dan fe de que el presente proyecto final de graduación ha sido aprobado y cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica.

Cartago, 21 de junio 2021

Resumen

Para proporcionar del conocimiento semántico sobre los objetos con los que van a interactuar los sistemas robóticos, se debe abordar el problema del aprendizaje de las representaciones semánticas a partir de las modalidades del lenguaje y la visión. El conocimiento semántico se refiere a la información conceptual, incluida la información semántica (significado) y léxica (palabra), y que proporciona la base para muchos de nuestros comportamientos no verbales cotidianos. Por lo tanto, es necesario desarrollar métodos que permitan a los robots procesar oraciones en un entorno del mundo real, por lo que este proyecto presenta un enfoque novedoso que utiliza Redes Convolucionales Gráficas para aprender representaciones de palabras basadas en el significado. El modelo propuesto consta de una primera capa que codifica representaciones unimodales y una segunda capa que integra estas representaciones unimodales en una para aprender una representación desde ambas modalidades. Los resultados experimentales muestran que el modelo propuesto supera al estado del arte en similitud semántica y que tiene la capacidad de simular juicios de similitud humana. Hasta donde sabemos, este enfoque es novedoso en el uso de Redes Convolucionales Gráficas para mejorar la calidad de las representaciones de palabras.

Palabras clave: Procesamiento Natural del Lenguaje, Redes Convolucionales Gráficas, Redes Neuronales Gráficas, Representaciones semánticas, Robótica Cognitiva

Abstract

To provide semantic knowledge about the objects that robotic systems are going to interact with, you must address the problem of learning semantic representations from modalities of language and vision. Semantic knowledge refers to conceptual information, including semantic (meaning) and lexical (word) information, and that provides the basis for many of our everyday non-verbal behaviors. Therefore, it is necessary to develop methods that enable robots to process sentences in a real-world environment, so this project introduces a novel approach that uses Graph Convolutional Networks to learn grounded meaning representations of words. The proposed model consists of a first layer that encodes unimodal representations, and a second layer that integrates these unimodal representations into one to learn a representation from both modalities. Experimental results show that the proposed model outperforms that state-of-the-art in semantic similarity and that can simulate human similarity judgments. To the best of our knowledge, this approach is novel in its use of Graph Convolutional Networks to enhance the quality of word representations.

Keywords: Cognitive robotics, Graph Convolutional Networks, Graph Neural Networks, Natural Language Processing, Semantic representations.

Dedicatory

It makes me nostalgic to think of all the things that have changed in my life since I started the process of becoming an engineer. I have learned from all the experiences with professors, classmates, friends, family, and people who are no longer part of my life today. Therefore, I am eternally grateful, they shared with me, taught me, and motivated me to be a better person, or at least I hope so.

For that reason, I want to dedicate to all those people, not this document, but rather everything that the title implies. To my brother Alvaro who always helped me in difficult economic situations, to my mother for her unconditional love and words of support, and to Mayra Vega and Guillermo Fernández for becoming my "new parents" while I lived in Cartago and all the love they gave me as if I were his son. To all the familiars and friends that I did not mention, but who know that this is possible thanks to them: This is also for you!

Despite not being the best student with incredible grades, passion and effort have made me dream big and I hope this project is the beginning of a new stage to learn new things and enjoy the simple details of life.

Acknowledgments

There is a lot of people that I would like to express my deep sense of thanks and gratitude to. Concluding this academic process is possible thanks to them and the words below fall short to thank you for everything.

I would like to thank my supervisor Mariella Dimiccoli, for trusting me and allowing me to carry out the project in the Perception and Manipulation Group of the Institut de Robòtica i Informàtica Industrial. Also, for always being aware and for all the crucial feedback and guidance, without her completing this project would not have been possible. I am heartily grateful, and I look forward to the opportunity to work together again.

I would also like to thank Juan Luis Crespo, for not hesitating to be my advisor professor and guide me throughout this process. For all that he taught me in several courses and that was essential to carry out this project. In addition to the feedback that he gave me to develop adequately this project.

To Iván Araya, Felipe Meza, and Johanna Muñoz for all the recommendations that they gave me in the previous defense. Besides, thank you to all professors that are part of the major of Mechatronics Engineering for share their knowledge and dedication like the amazing people that they are.

Finally, thank you to the Costa Rica Institute of Technology for the incredible education, for allowing me to have the scholarship to study and make me grow both personally and professionally.

Jose Lainer Alfaro

Barcelona, June 2021

Contents

List of Figures

List of Tables

List of Abbreviations

An alphabetically ordered list of the abbreviations used in this project is given next.

AI	Artificial Intelligence
CBOW	Continuous Bag of Words
CNNs	Convolutional Neural Networks
CSIC	Spanish Council for Scientific Research
DGL	Deep Graph Library
GCN	Graph Convolutional Network
GCNs	Graph Convolutional Networks
GNN	Graph Neural Network
GNNs	Graph Neural Networks
GPUs	Graphics Processing Units
IRI	Institut de Robòtica i Informàtica Industrial
ML	Machine Learning
NLP	Natural Language Processing
PyG	PyTorch Geometric
TEC	Instituto Tecnológico de Costa Rica
UPC	Universitat Politècnica de Catalunya

Chapter 1 **Introduction**

Project Environment

The project has been carried out in the Institut de Robòtica i Informàtica Industrial (IRI), which is a Joint University Research Institute located in Barcelona and participated by the Spanish Council for Scientific Research (CSIC) and the Universitat Politècnica de Catalunya (UPC) that conducts basic and applied research in human-centered robotics and automatic control [1]. The institute, founded in 1995, is a key player in the Spanish robotics and automatic control scenes, and a valued participant in many international collaborations.

According to [1], the IRI has three main objectives: to carry out excellent research in Robotics and Automatic Control, to cooperate with the community in industrial-technological projects, and to offer scientific education through graduate courses. The research activities are organized in four research lines:

- Automatic Control.
- Kinematics & Robot Design.
- Mobile Robotics & Intelligent Systems.
- Perception & Manipulation (the project belongs to this research line).

Furthermore, IRI has received the María de Maeztu scientific excellence seal for the period July 2017 to June 2021. In this period, a strategic research program in human-centered robotics will be developed. This is the main accreditation given by the Spanish Government to research units that stand out for the impact and international relevance of their results. Also, IRI received the TECNIO accreditation in November 2018 and November 2020. This seal is granted by the Catalan Government to recognized and differential technology providers which are involved in the tech transfer process [1].

Problem Description

Cognitive robotics is an approach to creating artificial intelligence in robots by allowing them to learn and respond to real-world situations, rather than pre-programming the robot with specific responses to every imaginable stimulus. Objective robotic cognitive abilities include

perceptual processing, attention allocation, anticipation, planning, complex motor coordination, reasoning about other agents, and perhaps even about your mental states [2].

It is necessary to develop methods that enable robots to process sentences in a real-world environment if we want them can communicate naturally with people. However, the language is a dynamic symbol system in which the meaning of a sign depends on the context and is understood subjectively [3]. The meaning of phrases and words is something that robots must ground, but this is not only determined by what they represent, but also by their relationship to other words. This is called distributional semantics and learning it is also crucial.

Semantic knowledge refers to conceptual information, including semantic (meaning) and lexical (word) information, and that provides the basis for many of our everyday non-verbal behaviors. For example, if you want to crack an egg on a bowl, you must first recognize them and later infer their unobserved qualities (the bowl is rigid, the egg is hard-shell and liquid inside, and so forth), and employ the appropriate praxis to achieve the current goal. All tasks require semantic knowledge about both the actions and the objects.

As part of the project and in relation to cognitive robotics, if you want to provide semantic knowledge about the objects that robotic systems are going to interact with, you must address the problem of learning semantic representations. As demonstrated in [4], new types of distributive models based on perception conceptualize the problem of the meaning representation as one of learning from multiple points of view corresponding to different modalities, so that attention will focus mainly on the modalities of vision and language. These modalities together ensure both the semantic relatedness and the perceptual grounding of the representations of the objects.

Problem Synthesis

The project will focus on investigating the suitability of a new model that uses Graph Convolutional Networks (GCNs) for learning semantic representations of objects from modalities of language and vision, to provide robotic systems with the necessary knowledge to interact with them. The knowledge can be organized in graphs and GCNs have proved to be successful in dealing with graph-structured data.

General Objective

Design a system that learns semantic representations of objects by simultaneously combining language and vision modalities so that IRI robotic systems can interact with them.

Specific Objectives

1. Determine the influential characteristics of existing solutions that allow the problem to be adequately fragmented.
2. Automate the process of acquiring information and preparing the data set for better exposure of these to the model.
3. Establish the programming logic necessary for the generation of the model that addresses the problem of learning semantic representations.
4. Validate the model developed with the execution of tests that allow the verification of the established criteria.

Document Structure

In Chapter 2 will be presented the theoretical framework, which introduces and describes the theory that explains why the research problem under study exists, besides the concepts that are relevant to the topic and that relate to the areas of knowledge being considered. Chapter 3 describes the methodology used in the project, and which is based on the engineering design process. The details of technical specifications, scope, and limitations of the design proposal for the project appear in Chapter 4. Experimental results and their analysis will be shown in Chapter 5, including the different strategies of verification and validation that were applied to the proposed design, in addition to the economic analysis. Chapter 6 includes conclusions and recommendations obtained from the project and in Chapter 7 appears the references to the bibliographic sources from which information necessary to develop the project was taken. Lastly, Chapter 8 and Chapter 9 corresponds to Appendix and Annex, respectively.

Main Contribution Engineering

To provide semantic knowledge about the objects that robotic systems are going to interact with, this project introduces a novel approach that uses Graph Convolutional Networks to learn grounded meaning representations of words, and that consist of a first layer that encodes unimodal representations, and a second layer that integrates these unimodal representations into one to learn a representation from both modalities. The proposed model validates the ability to simulate human similarity judgments and outperform the state-of-the-art [4] in semantic similarity.

Chapter 2 **Theoretical Framework**

This chapter helps to clarify the implicit theory in a manner that is more clearly defined, besides introducing and describe the key concepts that are relevant to the topic. Evaluate definitions, theories, and models to explain why the research problem under study exists. The problem of learning semantic representations of objects in robots is related to the area of Cognitive Robotics, so this area and the meaning of semantic representations will be explained. Then, the role of Natural Language Processing in the project and several algorithms to generate the desired model that involves Graph Neural Networks.

2.1 Cognitive Robotics

The research in the fields of Artificial Intelligence (AI) and robotics are strongly connected and in recent years more and more complex robots have been developed. However, programming these systems has become more difficult, because there is a clear need for such robots to be able to adapt and perform certain tasks autonomously, or even learn by themselves how to act [2]. Such systems offer unprecedented potential for AI to help in a variety of human-centric applications such as elder care and household maintenance. According to [5], the importance of allowing non-expert users to interact with them naturally and comfortably increases, and natural language is an excellent modality for end-users to give instructions and teach robots about their environments.

Object manipulation in real-world settings is a very hard problem in robotics, yet it is one of the most important skills for robots to possess. Through manipulation, they can interact with the world and therefore become useful and helpful to humans. Yet to produce even the simplest human-like behaviors, a humanoid robot must be able to see, act, and react continuously. Even more so for object manipulation tasks, which require precise and coordinated movements of the arm and hand [2].

In Cognitive Robotics the aim is to provide robots with cognitive processes, like humans and animals, creating artificial intelligence in robots by allowing them to learn and respond to real-world situations, rather than pre-programming with specific responses to every imaginable stimulus. Within the abilities includes perceptual processing, attention allocation, anticipation, planning, complex motor coordination, reasoning about other agents, and perhaps even about your

mental states [2], i.e., an integrated view of the body is taken, including the motor system, the perceptual system, and the body's interactions with the environment.

The acquisition of knowledge, whether through actions or perception, is a big part of cognitive robotics research [2]. Most social interactions in dialogue among humans require a substantial domain and context-specific knowledge to interpret the intents, emotions, and social relations among the agents involved. Lack of the kind of knowledge and models, it is hard for a robot or virtual agent to infer the social contexts and generate proper responses.

Robots using language must learn how words are grounded on the perceptual and noisy world in which a robot operates, and natural language systems can benefit from the rich contextual information provided by sensor data about the world. Despite an extensive and growing body of research, some challenges still need to be addressed before we see language-controlled robotic assistants deployed in human spaces:

- Learned models of language grounding should, as much as possible, be shared among all robots in the environment [5]. Similarly, robots that are not physically co-located should be able to share and combine learned models when applicable. Sharing data among robots with different sensors further increases complexity since learned models cannot be directly transferred.
- Moving beyond descriptive language to more general vocabulary and higher-level representations of meanings, including more abstract, pragmatic, and intent-driven language interpretations [5]. This will require richer representations of contextual world information, including models of the time, people, and intent.
- Asking questions about specific topics often leads to faster learning than receiving data sequentially, in part because queries can be selected using a variety of information-theoretic methods [5]. To go beyond learning passively from a human teacher, it is necessary for a robot to ask questions and otherwise direct its learning, which will ultimately require the incorporation of models of dialog and discourse.

2.2 Semantic Representations

Human language does not exist in isolation, because it is learned, understood, and applied in the physical world in which people exist, as seen in [5]. Finding the connection between symbols and their underlying meanings is the grounded language acquisition problem: taking linguistic

tokens and learning to interpret them by connecting them to real-world percepts and actions [6]. A true understanding of natural language requires capturing the connection between linguistic concepts and the world as experienced through perception, and even most abstract concepts are based on metaphorical references to more concrete physical concepts.

According to [6], ideally, an AI system would be able to learn the language like a human child, by being exposed to utterances in a rich perceptual environment. The perceptual context would provide the necessary supervisory information and learning the connection between language and perception would ground the semantic representations of the system on its perception of the world. However, such supervisory information is highly ambiguous and noisy since the language could be referring to numerous aspects of the environment or may not even refer to anything in the current perceivable context [6].

In general, there has been an increasing interest in deriving semantic representations from text corpora in terms of word vector space [7], and different models specify a mechanism for constructing semantic representations based on the *Distributional Hypothesis* [8], which states that *words that occur in similar contexts should have similar meanings*. However, word meanings are tied to sensory experience as shown in [9], at least for concrete nouns. Indeed, psychological studies have given pieces of evidence that the meaning of words is grounded in perception and report a bias between what is said in texts and what can be seen in images [10].

To further improve the quality of word representations, leveraging multimodal information is crucial. Zablocki et al. [10] stated that exploiting the visual surroundings and context of objects might be useful to grasp the semantics of words, e.g., from an image of an apple, we can infer its shape, color, and texture. Nevertheless, from an image of an apple growing on a tree (context), we can infer the relative size of apples, and that apples are fruits that grow on trees. If there is someone that is eating the apple, we can infer that apples are edible, and so on.

A widely adopted way to test vector-based models is to measure how well model-generated scores approximate human similarity judgments about pairs of words [11]. More specifically, measure how well the model predictions of word similarity correlate with human semantic and visual similarity ratings [12]. Several benchmarks provide gold labels (i.e., human judgment scores) for word pairs, then the Spearman's correlation is computed between the list of similarity scores given by the model (cosine-similarity between multimodal vectors) and the gold labels. The higher the correlation is, the more semantic is captured in the embeddings [10].

As demonstrated in [4], the ability to judge similarity underlies many cognitive tasks such as semantic priming and practical applications such as document retrieval.

2.3 Natural Language Processing

Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret, and manipulate human language. NLP borrows elements from many disciplines, including computer science and computational linguistics, in its quest to bridge the gap between human communication and understanding computers [13]. Although that is not a new science, technology is advancing rapidly thanks to an increased interest in human-machine communication, as well as the availability of big data, powerful computers, and improved algorithms.

Machines can analyze more language-based data than humans, without fatigue and in a consistent and unbiased way. Considering the staggering amount of unstructured data that is generated every day, from medical records to social media, automation will be instrumental in fully analyzing text and speech data efficiently. Basic NLP tasks include symbolization and parsing, lemmatization/derivation, part-of-speech labeling, language detection, and identification of semantic relationships.

NLP has led to many achievements, of course, as many language problems and phenomena could be addressed using only written text. Nevertheless, many open questions, which cannot be solved with written text alone, remain in NLP. Now a new academic challenge is being introduced regarding NLP that uses sensor-motor information in real-world settings, namely language and robotics [5].

2.4 Word Embeddings

In NLP, words are often mapped into vectors that contain numeric values so that the machine can understand them. Word embedding is a type of mapping that allows words with similar meanings to have similar representation, i.e., the texts converted into numbers and there may be different numerical representations of the same text [14].

But why do we need Word Embeddings? According to [14], many Machine Learning (ML) algorithms and almost all Deep Learning Architectures are incapable of processing strings or plain text in their raw form, because they require numbers as inputs to perform any sort of job (classification, regression, and so forth). With the vast amount of data that is present in the text, it is imperative to extract knowledge out of it and build applications (sentiment analysis, news classification, clustering by Google). Using word embeddings, you can create representations for words that capture their semantic relationships, meanings, and the different types of contexts they are used in.

Different types of word embeddings can be broadly classified into two categories: Frequency-based Embedding and Prediction based Embedding. Due to the limitations of the first one, the project will focus on the second one. The Prediction based Embedding provides probabilities to the words and proves to be state of the art for tasks like word analogies and word similarities [14]. Word embeddings (Word2Vec) leverages the context of the target words, using the surrounding words to represent the target words with a Neural Network whose hidden layer encodes the word representation [15]. There are two types of word embeddings: Continuous Bag of Words (CBOW) and Skip-gram.

2.4.1 Continuous Bag of Words

The idea of the Continuous Bag of Words is to predict the word given a context. Then the inputs are the words surrounding, while the output is the target word, as illustrated in [14, Figure 1]. For this type of word embedding, all the examples with the target word as the target are fed into the networks and taking the average of the extracted hidden layer.

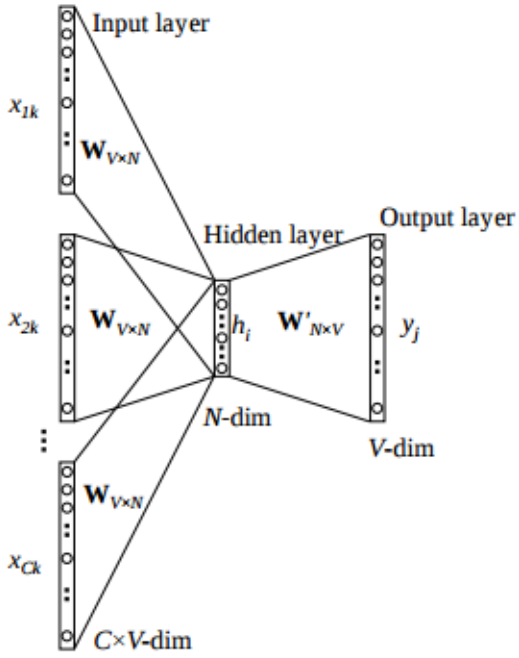


Figure 1. The continuous bag of words architecture. Obtained from [14]

Assuming the sentence “He is a nice guy”, to compute the word representation for the word “a”, we need to feed “He”, “is”, “nice”, and “guy” into the Neural Network and take the average of the value in the hidden layer. According to [15], it is claimed that Skip-gram tends to do better in rare words. Nevertheless, the performance of Skip-gram and CBOW are generally similar.

2.4.2 Skip-gram

For skip-gram, the idea is that the input is the target word, while the outputs are the words surrounding the target words, i.e., skip-gram aims to predict the context given a word, as illustrated in [14, Figure 2]. The model loops through the words in each sentence and tries to use the current word to predict its context, and the limit on the number of words in each context is determined by a parameter called *window size* [16]. In the sentence “I have a tabby cat”, for example, the input would be “a”, whereas the output is “I”, “have”, “tabby”, and “cat”, assuming that the parameter window size is 2.

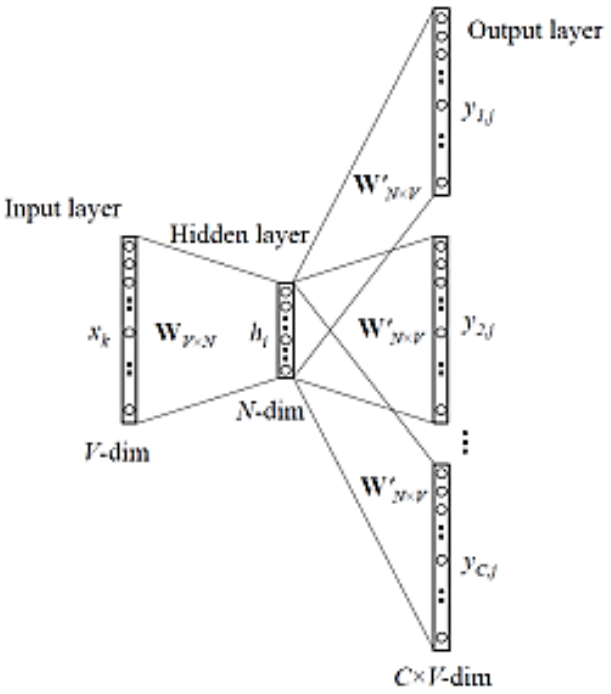


Figure 2. The skip-gram architecture. Obtained from [14]

For the target words, the word embedding is obtained by extracting the hidden layer after feeding the one-hot representations of that word into the network. Using skip-gram, the length of the hidden layer (that is a parameter called *vector size*) determines the new representation dimension of the vocabulary, and with this, the vectors will be more "meaningful" in terms of describing the relationship between words, as demonstrated in [15].

Mikolov et al. [7] presented several extensions of the original Skip-gram model as subsampling frequent words that accelerate learning (around 2x – 10x) and even significantly improves the accuracy of the learned vectors of the rare words or negative sampling that results in faster training and better vector representations for frequent words, compared to more complex hierarchical softmax.

- **Negative Sampling**: this parameter addresses the problem that the skip-gram model has a large number of weights, modifying a small percentage of the weights in each training sample, rather than all of them.
- **Subsampling of Frequent Words**: words such as "in", "the", "my", and so on, do not bring much information and can easily occur hundreds of millions of times (too

many samples containing these words). Then, the authors propose a sampling rate which states whether we should keep a word or not.

If the word embeddings are reduced to n-dimensions, it is possible to see relationships between certain words. As seen in [16, Figure 3], there are different examples of semantic relationship (male and female designations, and countries and their capital cities) and syntactic relationships (present vs past tense). The skip-gram model can learn implicitly the relationships between concepts and automatically organize them and considering that during the training we did not provide any supervised information. These relationships are represented by vectors of similar magnitude and mapped close to each other.

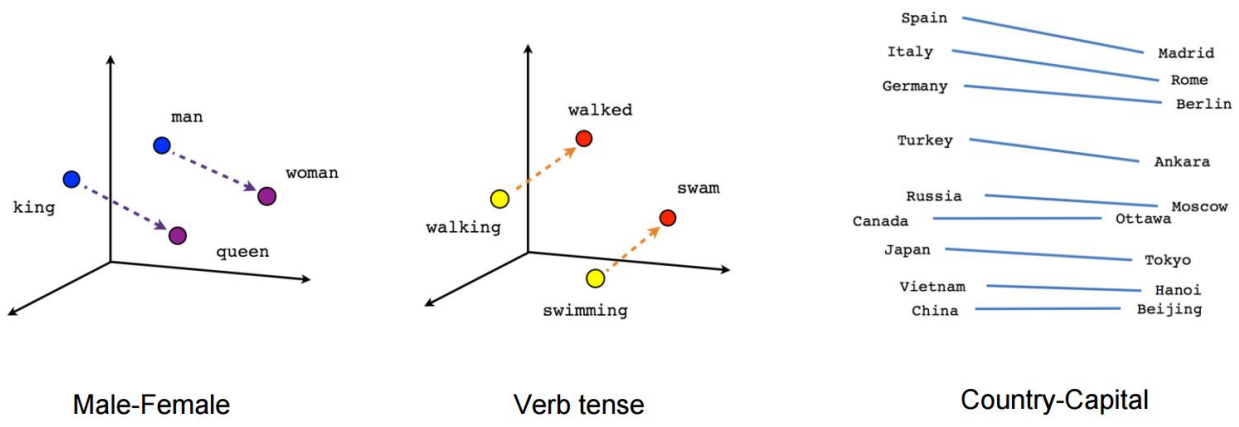


Figure 3. Examples of the semantic and syntactic relationships between words. Obtained from [16]

2.5 Graph Neural Networks

Generalizing well-established neural models like RNNs or CNNs to work on arbitrarily structured graphs is a challenging problem. Molecular graph structures, biological protein-protein networks, recommender systems, social networks—all these fields can be modeled as graphs, which capture interactions (i.e., edges) between individual units (i.e., nodes). A graph is a data structure consisting of nodes (vertices) and edges connected to represent information with no definite beginning or end. In contrast to [17], edges can be either directed or undirected, depending on whether there exist directional dependencies between vertices, and directed graphs can be unidirectional or bidirectional.

According to [18], a convenient way to represent graphs is through an *adjacency matrix* $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $\mathcal{V}$ is set of nodes. To represent a graph with an adjacency matrix, order the nodes in the graph so that every node indexes a particular row and column in the adjacency matrix, and then represent the presence of edges as entries in this matrix: $A[u,v] = 1$ if $(u,v) \in \mathcal{E}$ (set of edges) and $A[u,v] = 0$ otherwise. Each node has a set of features defining it and each edge may connect nodes that have similar features (it shows interaction or relationship between them).

As seen in [19], graphs are not only useful as structured knowledge repositories: they also play a key role in modern machine learning. Many ML applications look to discover new patterns using graph-structured data as feature information or make predictions, but the central problem is discovering a way to incorporate information about graph structure into a machine learning model. The most important and well-studied ML tasks on graph data are node classification, relation prediction, clustering, and community detection, and graph classification, regression, and clustering. For more details, see [18].

Recently, there has been an increase of approaches that seek to learn representations that encode structural information about the graph and to learn a mapping that embeddings nodes, or entire (sub)graphs, as points in a low-dimensional vector space $\mathbb{R}^d$. Hamilton [19] stated that the goal is to optimize this mapping so that geometric relationships in the embedding space reflect the structure of the original graph. Once the learned embeddings are optimized, they can be used as feature inputs for downstream machine learning tasks.

These low-dimensional embeddings can be viewed as encoding or projecting, nodes into a latent space [19], where geometric relations in this latent space correspond to interactions in the original graph. An example embedding can be visualized in [19, Figure 4], with the Zachary Karate Club social network and where two-dimensional node embeddings capture the community structure implicit in the network. In the embedding space, the distances between nodes reflect similarity in the original graph, and the node embeddings are spatially clustered according to the different color-coded communities.

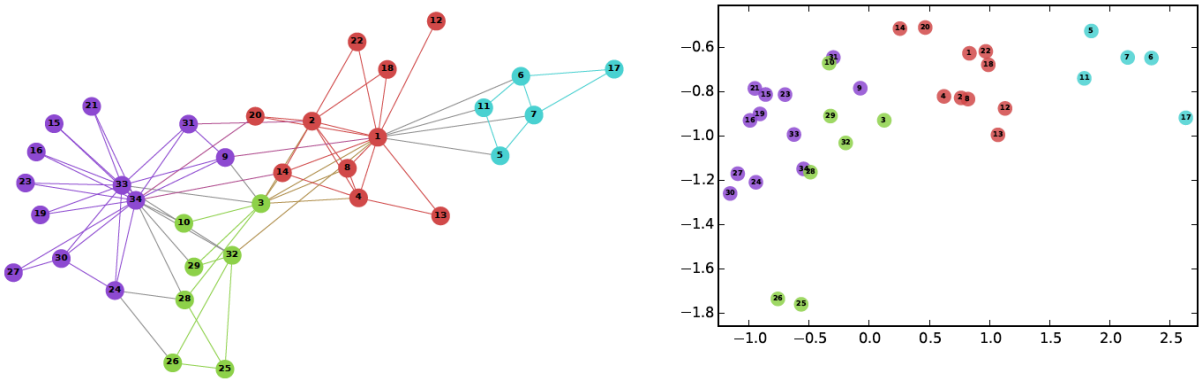


Figure 4. Zachary Karate Club social network and his two-dimensional visualization of node embeddings. Obtained from [19]

Hamilton [18] sees the graph representation learning problem as involving two key operations. First, an encoder model maps each node in the graph into a low-dimensional vector or embedding. Next, a decoder model takes the low-dimensional node embeddings and uses them to reconstruct information about each node's neighborhood in the original graph. This idea is an encoder-decoder perspective and summarized in [19, Figure 5]. Learn to decode high-dimensional graph information from encoded low-dimensional embeddings, it will produce embeddings that should contain all information necessary for downstream ML tasks.

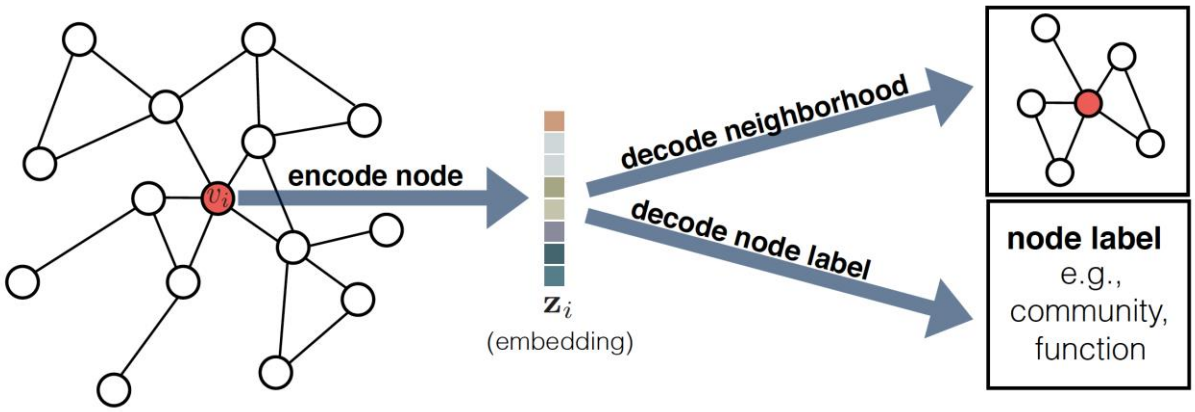


Figure 5. Overview of the encoder-decoder approach. Obtained from [19]

The node embedding approaches generate representations of nodes, where we simply optimized a unique embedding vector for each node. However, there are more complex encoder models, such as the Graph Neural Network (GNN), which is a general framework for defining

deep neural networks on graph data. According to [19], the key idea is that instead of aggregating information from neighbors, the graphs can be viewed as specifying scaffolding for a *message passing* algorithm between nodes.

The idea behind the GNN message passing framework is straightforward: every node aggregates information from its local neighborhood at each iteration, and as these iterations progress, each node embedding contains more information from the farthest reaches of the graph [18]. For instance, after the first iteration (k = 1) every node embedding contains information about the features of its immediate graph neighbors; after the second iteration (k = 2) every node embedding contains information from its 2-hop neighborhood; and generally, after k iterations, each node embedding contains information about its k-hop neighborhood.

The information encoded in these node embeddings comes in two forms:

1. **Structural information:** For instance, after k iterations of GNN message passing, the embedding of each node can encode information about the degrees of all the nodes in the k-hop neighborhood [18].
2. **Feature-based:** After k iterations, the embeddings for each node also encode information about all the features in their k-hop neighborhood [18]. This local feature-aggregation behavior of GNNs is analogous to the behavior of the convolutional kernels in Convolutional Neural Networks (CNNs). However, whereas CNNs aggregate feature information from spatially-defined patches in an image, GNNs aggregate information based on local graph neighborhoods.

2.6 Graph Convolutional Networks

Machine learning on graphs is a difficult task due to the highly complex graph structure, but also informative. Most graph neural network models have a somewhat universal architecture in common, and Kipf [20] refers to these models as Graph Convolutional Networks (GCNs). Convolutional because filter parameters are typically shared over all locations in the graph. GCNs is a very powerful neural network architecture for ML, and according to [21], they are so powerful that even a randomly initiated 2-layer GCN can produce useful feature representations of nodes in networks.

More formally, GCN is a neural network that operates on graphs. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a GCN takes as input:

- A features description x_i for every node i; summarized in a $N \times F$ feature matrix X, where N is the number of nodes and F is the number of input features for each node.
- An $N \times N$ matrix representation of the graph structure such as the adjacency matrix A of $\mathcal{G}$.

Generalizing the convolution operator to irregular domains is typically expressed as a *neighborhood aggregation* or *message passing* scheme. In terms of GNNs, for a single reference node, the neighboring nodes pass their messages embeddings through the edge neural networks [17]. The new embedding of the node is updated by applying a function on the current embedding and a summation of the edge neural network outputs of the neighboring node embeddings. With $x_i^{(k-1)} \in \mathbb{R}^F$ denoting node features of node i in layer $(k-1)$ and $e_{j,i} \in \mathbb{R}^D$ denoting (optional) edge features from node j to node i.

Message passing can be described as:

$$x_i^{(k)} = \gamma^{(k)}\left(x_i^{(k-1)}, \mathcal{H}_{j \in \mathcal{N}(i)} \phi^{(k)}\left(x_i^{(k-1)}, x_j^{(k-1)}, e_{j,i}\right)\right) \tag{2.1}$$

where $\mathcal{H}$ denotes a differentiable, permutation invariant function (e.g., sum, mean, or max), and γ and ϕ denote differentiable functions such as MLPs (Multilayer Perceptron). This process is performed, in parallel, on all nodes in the network as embeddings in layer $H+1$ depend on embeddings in layer H. In practice, that is why we do not need to move from one node to another to carry out message passing [17].

The GCN layer [22] is mathematically defined as:

$$x_i^{(k)} = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(i)}\sqrt{\deg(i)}} \cdot \left(\Theta \cdot x_j^{(k-1)}\right) \tag{2.2}$$

where neighboring node features are first transformed by a weight matrix Θ, normalized by their degree, and finally summed up. This formula can be divided into the following steps:

1. Add self-loops to the adjacency matrix.
2. Linearly transform node feature matrix.
3. Compute normalization coefficients.
4. Normalize node features in ϕ.
5. Sum up neighboring node features.

Each layer corresponds to a feature matrix, and where each row is a feature representation of a node. These features are aggregated to form the next layer's features (at each layer). In this way, at each consecutive layer, the features become increasingly more abstract. A multi-layer GCN is schematically depicted in [20, Figure 6].

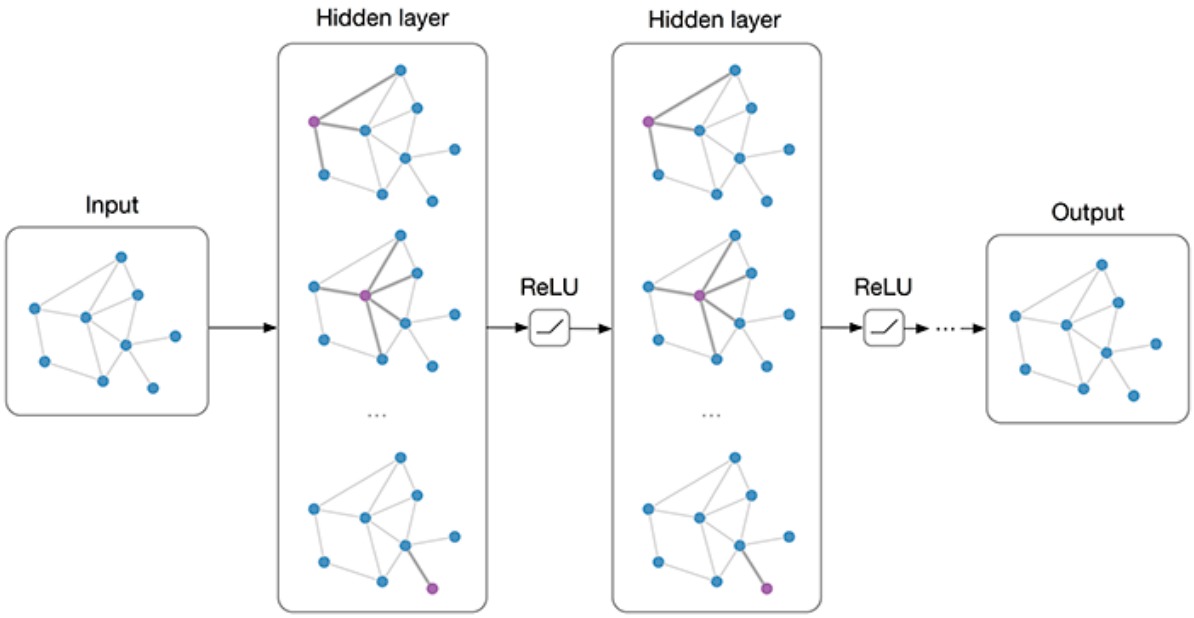


Figure 6. Multi-layer Graph Convolutional Network (GCN) with first-order filters. Obtained from [20]

Kipf [22] realized experiments to investigate the influence of the number of layers (model depth) on classification performance. The datasets used were Cora, Citeseer, and Pubmed using all labels, on 5-fold cross-validation, in which each cross-validation split was trained for 400 epochs using the Adam optimizer with a learning rate of 0.01, dropout rate vale of 0.5 (first and last layer), L2 regularization value of $5x10^{-4}$, and 16 units for each hidden layer. The results are illustrated in [22, Figure 7].

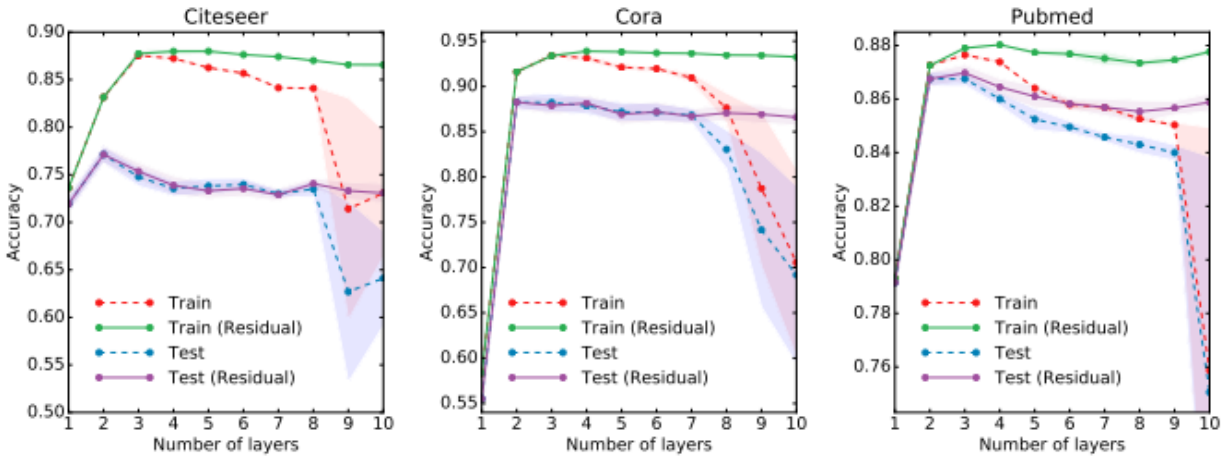


Figure 7. Influence of the number of layers on classification performance

The best results are obtained with a model that has 2 or 3 layers (for the datasets considered). Besides, training without the use of residual connections can become difficult for models deeper than 7 layers since the effective context size for each node increase by the size of its neighborhood for a model with K layers (with each additional layer). Also, overfitting can become a problem as the number of parameters increases with the depth of the model.

Chapter 3 **Methodology**

The methodology used during the project is based on the Engineering Design Process. As seen in [23] this process consists of five steps for solving the design problem as illustrated in [23, Figure 8]. Furthermore, it is important to remember that it is an iterative process and involves continually refining the design to get a solution that accomplishes the requirements of the project.

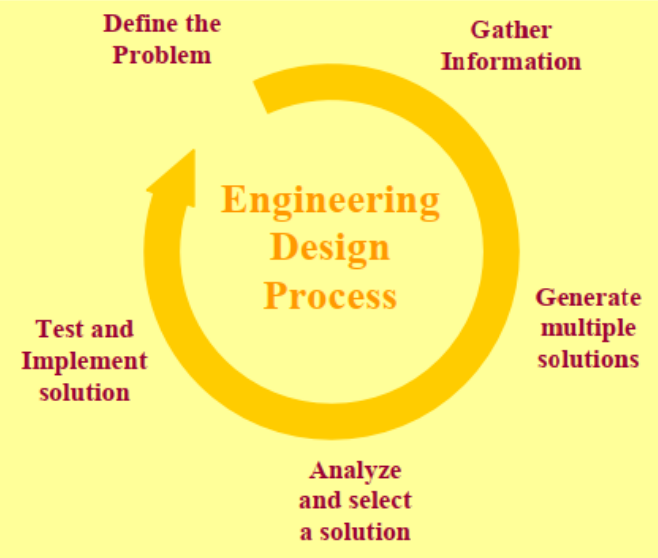


Figure 8. Engineering Design Process. Obtained from [23]

In the next subsections, a description of each step and details involved with the implemented solution will be mentioned and how they are related to the Engineering Design Process. Also, the Machine Learning Project Checklist [24] was used because the main area of the project is Artificial Intelligence, and it is helpful to have a guide to structure the right combination of elements to reliably deliver a good solution to any Machine Learning (ML) problem.

3.1 Define the problem

According to [23], the solution to a problem should begin with an unambiguous definition of the problem. Subsequently, the need must be identified and established (always occurs in response to a human need). In contrast to [24], knowing this is important because it will help to

determine which algorithms to select, which tests to use to evaluate and validate the proposed solution, and how much effort to spend tweaking it, i.e., this will help to frame the problem.

The problem and the need of the project are to investigate the suitability of GCNs for learning semantic representations of objects (from modalities of language and vision), to provide robotic systems with the necessary knowledge to interact with them.

As seen in [23], once the need has been established, it must define that need in terms of an engineering design problem statement, and that requires collecting data, run experiments, and perform computations that allow that need to be expressed as part of an engineering problem-solving process. In this project.

Finally, this step also involved establishing selection criteria (specifications that a design solution must have or the attributes that it must possess to be considered successful), and at this point, it will most likely need to be redefined or modified. Furthermore, these criteria should not be too specific to allow flexibility throughout the design process [23]. The following is a list of preliminary criteria for the project:

- The solution must employ Graph Neural Networks.
- The solution must employ modalities of language and vision.
- The solution has to consider the use of libraries such as PyTorch Geometric (PyG) or Deep Graph Library (DGL).
- The solution should apply the theoretical knowledge of word embeddings.
- The solution must show studies of the influence of the hyperparameters of the model.
- The solution must be validated with a benchmark of word similarity.
- The solution should be compared against the state-of-the-art.

3.2 Gather pertinent information

Once the problem has been defined, it proceeded to gather pertinent information (can reveal facts about the problem that results in a redefinition of the problem [23]). Also, it is good practice to list and verify the assumptions that have been made so far; this can help to catch serious issues early on [24].

Some main resources to gather information were the TEC bases to compile scientific papers of interest, textbooks used in different courses of the major (e.g., Artificial Intelligence, Computer

Vision, Mechatronics Systems Design, Robotics), internet articles, and so forth. For this project, it was primordial to receive information and feedback from the project supervisor Mariella Dimiccoli, a researcher in the Perception and Manipulation Group at IRI, and Juan Luis Crespo, professor in the Mechatronics Engineering Area at TEC.

Besides, as part of this step, the data for the model was obtained. Two datasets were used: 1) the first one is the attribute-based representations proposed by [3], and the second one is the skip-gram-based representations used by [10], where for textual data use a dump of the Wikipedia database (a large collection of English texts) and for visual data use the Visual Genome dataset [25]. It is recommended to automate as much as possible, to obtain new data easily as illustrated in [5]. It should create a workspace, check the size and type of the data, how much space they require, exploration and visualization, document what you have learned, among others.

3.3 Generate multiple solutions

The best practice is to start with the existing solutions to the problem and then see what is wrong with those solutions and focus on how to improve your weaknesses. Consciously combining new ideas, tools, and methods to produce a unique solution to the problem, and this process is called synthesis [23]. The problem of the project addresses a need that is not new, but there are no existing solutions about learning semantic representations through multimodal graph neural networks, and for that reason, is an innovative project and research.

This step also involves preparing the data: work on copies of the data to keep the original dataset intact, convert the data to a format easily manipulate, data cleaning, feature scaling, and so forth. It is important to write functions for all data transformations because you can easily prepare the data the next time you get a fresh dataset, apply these transformations in future projects, and make it easy to treat your preparation choices as hyperparameters [24]. In the next section, all implemented functions will be explained.

With the information collected in the previous step, different models of graph neural networks were trained using standard parameters and their performances were measured and compared. Also, the most significant variables of each algorithm were analyzed, and the types of errors that each model generated.

Lastly, it must remember that this is an iterative process, where hyperparameters must be adjusted, performance measured, errors analyzed, realize different combinations between

algorithms, among others. Combining the best models will often produce better performance than running them individually [24].

3.4 Analyze and select a solution

At this phase of the design process, each alternative solution must be analyzed and review the defined scope, time, and costs. This is a very subjective step and should be done by a group of experienced people, and consider the results of a design analysis [23].

Each design problem is unique and requires different types of analysis, such as functional analysis, product safety and liability, ergonomics, and so forth. How is a project focused on software, selection considered different analyses such as accuracy or interpretability of the output, speed or training time, the scalability, among others.

As seen in [23], after analyzing the alternative solutions, you must decide and document which design solution is the best. Finally, the best solution will be refined and developed in more detail during the later stages of the design process.

3.5 Test and implement the solution

The final step of this process refers to the testing, construction, and fabrication of the solution to the design problem and where various implementation methods such as prototyping and concurrent engineering should be considered as illustrated in [23], in addition to documenting the design solution. Due to the nature of the problem, neither construction nor fabrication of the solution takes place, but testing and validation are primordial in the design process.

The solution was tested and validated in a benchmark that realizes a semantic task known as word similarity, where Spearman's correlation is computed between the list of similarity scores given by the model and the gold labels (i.e., human judgment scores). The benchmark used was the one created by [4], which consists exclusively of concrete nouns and it consists of 7576-word pairs, in which the similarity ratings were obtained using Amazon Mechanical Turk (AMT).

Finally, at every step, you may find that your potential solution is flawed and must go back to a previous step to get a viable solution. Without proper testing at all stages of the process, you may find yourself making costly mistakes later.

Chapter 4 **Design Proposal**

This chapter clearly and precisely demonstrates the follow-up of the methodology proposed in the previous chapter. Based on the Engineering Design Process [23] the following block diagram in Figure 9 was defined to respond to all objectives of the project and explain how the select solution was designed and implemented.

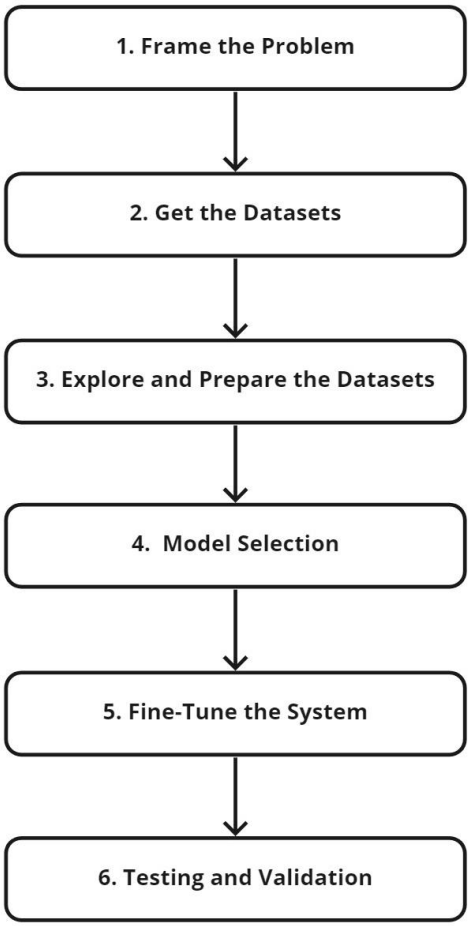


Figure 9. Project workflow block diagram

4.1 Frame the Problem

The first step was to define the objective of the project, which is to investigate the suitability of a new model that uses GCNs for learning semantic representations of objects from modalities of language and vision.

The idea of use GCNs instead of another algorithm comes from [12], where the modeling is also based on a graph. If you think of the famous WordNet [26], all knowledge is organized in graphs. For example, from the domain of computer science, can be considered community structures [27] on an underlying graph structure. Therefore, it is of interest to test the suitability of GCNs, since they have proved to be successful in dealing with graph-structured data.

As argued in the previous chapter, knowing the objective helps determine how to frame the problem, which algorithms to select, which tests to use to evaluate and validate the proposed solution, and so forth. In Table 1 are presented some characteristics (or questions) based on Machine Learning Project Checklist [24] and that need to be considered to frame adequately the problem. Considering this information before starting to develop the project can help to catch serious issues early on.

Table 1. The influential characteristic that allows frame adequately the problem

Define the objective.
What are the currents solutions (if any)?
Is it supervised, unsupervised, or reinforcement learning?
Is it a classification task, regression task, or something else?
How should performance be measured?
List the data you need.
Get the data.
Explore the data to get insights.
The data is ready or need to be prepared?
Write functions for all data transformations.
Research promising models.
Fine-tune the system.
Validate the results.

In this phase, some of the main articles were obtained to understand the problem of learning semantic representations and solutions proposed by different authors. Since that, the solution involves the use of GCNs, and according to [22], we are talking about semi-supervised learning and classification tasks.

The basic idea of the project is: use different data (textual and visual) to generate word representations (through word embeddings) that are more "meaningful" in terms of describing the relationship between words [14] and enhance the quality of these vectors with the help of GCNs. The data that was used, functions for all data transformations, models, and how the performance was measured, will be explained in the next steps.

4.2 Get the Datasets

In chapter 2 was defined that several studies reveal that the meaning of words is grounded in perception and the interest of learning word representations from multimodal approaches. This approach requires both textual and visual data to later feed the model based on GCNs with input feature vectors from different modalities. For that reason in this step, it was important to list the data needed, check how much space it will take, and check legal obligations (and get authorization if necessary), as demonstrated in [24]. The following representations of the dataset were used:

4.2.1 First Set (attribute-based)

The first set used in the project is the attribute-based representations proposed by [4]. These representations are for induvial words and are unique in their use of attributes as a means of representing the visual modality. Silberer et al. [4] created a large-scale dataset consisting of nearly 700K images and taxonomy of 636 visual attributes to train attribute classifier and extract attribute predictions for new images. Besides, analogously to representing the visual modality through attributes extracted from images, they represent the textual modality through attributes from text data (directly extracted from the corpus).

These representations in both modalities were scaled to the [−1, 1] range and it was already prepared to feed the model. Also, the goal of using these representations is to compare the results obtained by [4] with their model SAE. The words used in the dataset are illustrated in Figure 10.

- **Textual data**: attributes were extracted by running Strudel [28] on the WaCkypedia corpus [29], and by retaining only the ten attributes with the highest log-likelihood ratio scores for each target word. The union of the selected attributes leads to 2,362 dimensions for the textual vectors.
- **Visual data**: obtained automatically from images by training an SVM-based classifier for each attribute on the VISA dataset [4] and encoded via 414-dimensional vectors of attributes. More specifically, we used initial meaning representations of words for the McRae nouns [30] covered by the VISA dataset [4], which consists of a wide range of concrete concepts including animate and inanimate things (e.g., fruits, animals, vegetables, clothing, utensils, and vehicles).

According to [4], the attributes are suitable for describing visual phenomena (e.g., actions, objects, scenes), beyond providing a cognitive and natural way of expressing the salient properties of word meaning. This is crucial because they allow generalization to new instances when there are no training examples available (it is possible to say something about new objects even though we cannot identify them), i.e., attributes transcend category and task boundaries while providing a generic description of visual data [4].

```
['accordion' 'airplane' 'alligator' 'ambulance' 'anchor' 'ant' 'apartment'
 'apple' 'apron' 'ashtray' 'asparagus' 'avocado' 'bag' 'bagpipe' 'ball'
 'balloon' 'banana' 'banjo' 'banner' 'barn' 'barrel' 'basement' 'basket'
 'bathtub' 'baton' 'bazooka' 'beans' 'beaver' 'bed' 'bedroom' 'beehive'
 'beetle' 'beets' 'belt' 'bench' 'bike' 'birch' 'biscuit' 'bison'
 'blender' 'blouse' 'blueberry' 'bluejay' 'boat' 'bolts' 'bomb' 'book'
 'bookcase' 'boots' 'bottle' 'bouquet' 'bowl' 'box' 'bra' 'bracelet'
 'bread' 'brick' 'bridge' 'broccoli' 'broom' 'brush' 'bucket' 'buckle'
 'buffalo' 'buggy' 'building' 'bull' 'bullet' 'bungalow' 'bureau' 'bus'
 'butterfly' 'cabbage' 'cabin' 'cabinet' 'cage' 'cake' 'calf' 'camel'
 'camisole' 'canary' 'candle' 'cannon' 'canoe' 'cantaloupe' 'cape' 'car'
 'caribou' 'carpet' 'carrot' 'cart' 'cat' 'caterpillar' 'catfish'
 'cathedral' 'cauliflower' 'celery' 'cellar' 'cello' 'chain' 'chair'
 'chandelier' 'chapel' 'cheese' 'cheetah' 'cherry' 'chickadee' 'chicken'
 'chimp' 'chipmunk' 'church' 'clamp' 'clarinet' 'cloak' 'clock' 'closet'
 'coat' 'cockroach' 'cod' 'colander' 'comb' 'cork' 'corkscrew' 'corn'
 'cottage' 'couch' 'cougar' 'cow' 'coyote' 'crab' 'cranberry' 'crayon'
 'crocodile' 'crossbow' 'crow' 'crown' 'cucumber' 'cup' 'cupboard'
 'curtains' 'dagger' 'dandelion' 'deer' 'desk' 'dish' 'dishwasher' 'dog'
 'doll' 'dolphin' 'donkey' 'door' 'dove' 'dress' 'dresser' 'drill' 'drum'
 'duck' 'eagle' 'earmuffs' 'eel' 'eggplant' 'elephant' 'elevator' 'elk'
 'emu' 'envelope' 'falcon' 'faucet' 'fawn' 'fence' 'finch' 'flamingo'
 'flea' 'flute' 'football' 'freezer' 'fridge' 'frog' 'garage' 'garlic'
 'gate' 'giraffe' 'gloves' 'goat' 'goldfish' 'goose' 'gopher' 'gorilla'
 'gown' 'grape' 'grapefruit' 'grasshopper' 'grater' 'grenade' 'groundhog'
 'guitar' 'gun' 'guppy' 'hammer' 'hamster' 'hare' 'harmonica' 'hatchet'
 'hawk' 'helicopter' 'helmet' 'hoe' 'hook' 'hornet' 'horse' 'house' 'hut'
 'iguana' 'inn' 'jacket' 'jar' 'jeans' 'jeep' 'jet' 'kettle' 'kite'
 'knife' 'ladle' 'lamb' 'lamp' 'lantern' 'leopard' 'lettuce' 'level'
 'lime' 'lion' 'lobster' 'machete' 'magazine' 'mandarin' 'marble' 'mat'
 'menu' 'microscope' 'microwave' 'minnow' 'mirror' 'missile' 'mittens'
 'mixer' 'moose' 'moth' 'motorcycle' 'mouse' 'mug' 'mushroom' 'muzzle'
 'napkin' 'necklace' 'nightgown' 'nylons' 'oak' 'octopus' 'onions'
 'ostrich' 'otter' 'oven' 'owl' 'ox' 'paintbrush' 'pajamas' 'pan'
 'panther' 'pants' 'parakeet' 'parka' 'parsley' 'partridge' 'peach'
 'peacock' 'pear' 'pelican' 'pen' 'pencil' 'penguin' 'pepper' 'piano'
 'pickle' 'pier' 'pig' 'pigeon' 'pillow' 'pin' 'pine' 'pineapple' 'pistol'
 'plate' 'platypus' 'plum' 'pony' 'porcupine' 'pot' 'potato' 'projector'
 'pumpkin' 'pyramid' 'python' 'rabbit' 'raccoon' 'racquet' 'radio'
 'radish' 'raft' 'raisin' 'rake' 'raspberry' 'rat' 'rattlesnake' 'raven'
 'razor' 'revolver' 'rice' 'rifle' 'robe' 'robin' 'rock' 'rocker' 'rocket'
 'rooster' 'rope' 'ruler' 'sack' 'saddle' 'sailboat' 'salamander' 'salmon'
 'sandals' 'sardine' 'saxophone' 'scarf' 'scissors' 'scooter'
 'screwdriver' 'screws' 'seagull' 'seal' 'shack' 'shawl' 'sheep' 'shelves'
 'shield' 'ship' 'shirt' 'shoes' 'shotgun' 'shovel' 'shrimp' 'skillet'
 'skirt' 'skis' 'skunk' 'skyscraper' 'sled' 'sleigh' 'slippers' 'snail'
 'socks' 'sofa' 'spade' 'sparrow' 'spatula' 'spear' 'spider' 'spinach'
 'spoon' 'squid' 'squirrel' 'stereo' 'stone' 'stork' 'stove' 'strainer'
 'strawberry' 'submarine' 'subway' 'swan' 'sweater' 'swimsuit' 'sword'
 'table' 'tack' 'tangerine' 'taxi' 'telephone' 'tent' 'thermometer'
 'tiger' 'toad' 'toaster' 'toilet' 'tomahawk' 'tomato' 'tortoise'
 'tractor' 'trailer' 'train' 'tray' 'tricycle' 'tripod' 'trolley'
 'trombone' 'trout' 'truck' 'trumpet' 'tuba' 'tuna' 'turkey' 'turnip'
 'turtle' 'typewriter' 'umbrella' 'unicycle' 'van' 'veil' 'vine' 'violin'
 'vulture' 'wagon' 'wall' 'walnut' 'walrus' 'wand' 'wasp' 'whale' 'wheel'
 'wheelbarrow' 'whip' 'whistle' 'willow' 'woodpecker' 'worm' 'wrench'
 'yacht' 'zebra']
```

Figure 10. Word keys (439) of the dataset that was used in the project

4.2.2 Second Set (skip-gram based)

This data is the same used by [10], and initially, the idea was to compare the obtained results of this project against the state-of-the-art of this paper. But the final decision was to train a standard skip-gram model [7] with this data and generate skip-gram-based representations with the same words used in the First Set, but this will be explained in the next step.

- **Textual data**: it was used a dump of the Wikipedia database (a large collection of English texts) with a size of 17.3 GB and that provides us with 4 million articles and a vocabulary of approximately 2.5 million unique words.
- **Visual data**: it was used the Visual Genome dataset [25] that contains over 100K images with many different objects, 4842 unique entities with more than 10 occurrences, where each image has an average of 21 objects, 18 attributes, and 18 pairwise relationships between objects.

Once the representations were obtained, a workspace with enough memory storage space was created. They were stored in Google Drive because all transformations, functions, and models were developed using Colaboratory, which allows writing and execution of arbitrary python code through the browser, and is especially well suited to ML, education, and data analysis [31]. Specifically, is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including Graphics Processing Units (GPUs).

4.3 Explore and Prepare the Datasets

The next step was taking a glance at the data to get a general understanding of the kind of data since the goal is to go into a little more depth. As noted in the previous step, the First Set (formed by 439 words, in which textual data has 2368-dimensional vectors of attributes, visual data has 414-dimensional vectors) is ready to be processed in the model that will be explained in the next step. Nevertheless, the Second Set needs to be prepared before being able to use this information in the model, and this process is illustrated in Figure 11.

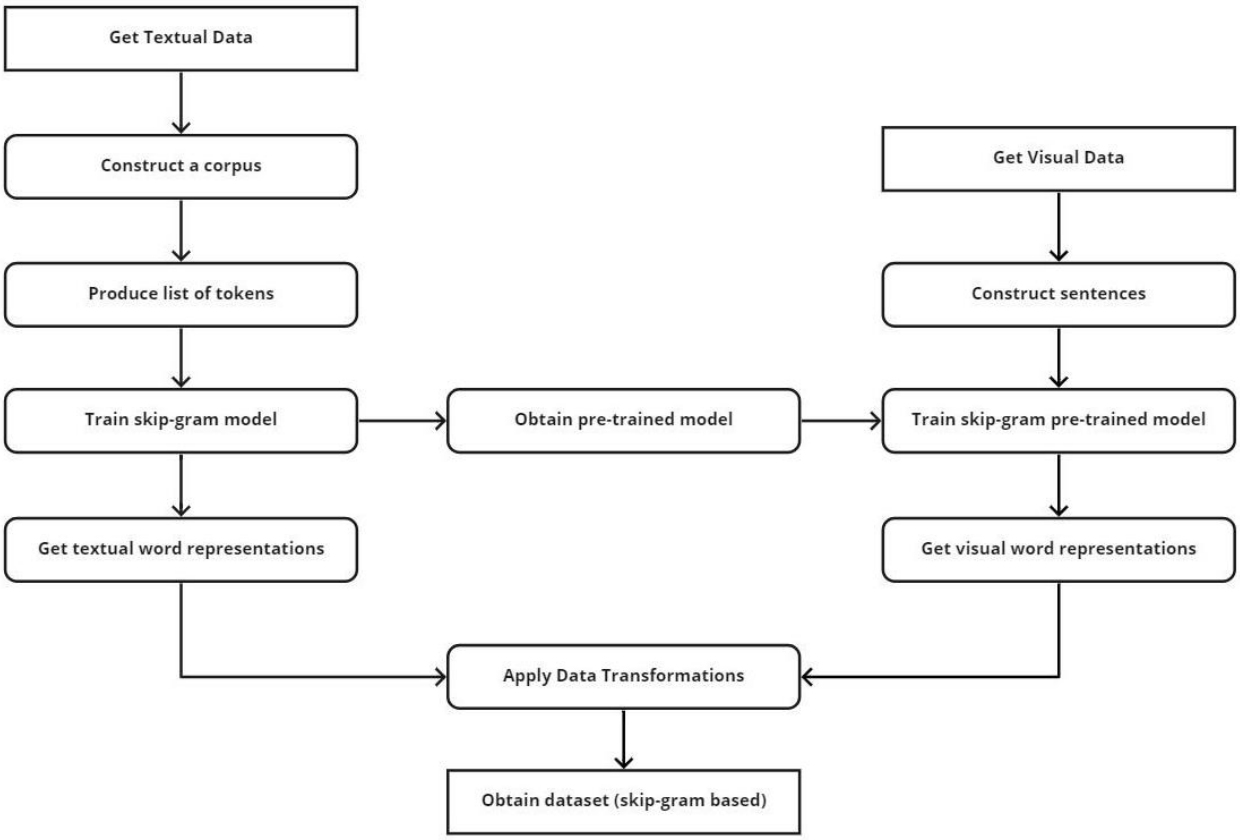


Figure 11. Block diagram of the process of preparing the Second Set

For the textual data (dump of Wikipedia database), we have plain text, and many ML algorithms are incapable of processing this. Then the solution was generating the same 439 words of the First Set through word embeddings, i.e., vectors that contain numeric values so that the machine can understand them, as argued in chapter 2. To realize these word embeddings, the data was cleaned and tokenized with Gemsin [32], which is a free open-source Python library for representing documents as semantic vectors and can train Word2Vec models. The skip-gram was selected because as demonstrated in [7], outperforms all the other models in the quality of the learned representations.

In the case of the visual data (Visual Genome dataset), we have a JSON file format that includes all object instances, the web address to download each image, x-y coordinates of each object bounding box, height-width of each object bounding box, and so forth. The solution for generating the word embeddings for this data is used in [10], where an image I can be seen as a bag of objects: $I = \{o_1, o_2, \ldots\}$, and this simple view gives high-level information about the

environment in which objects occur. This generates 108077 sentences (one for each image) which each sentence is a list with total objects that appears in that image, and that will be trained with the skip-gram model generated for textual data.

4.3.1 Skip-gram model for textual data of Second Set

Given a text corpus (dump of Wikipedia), the skip-gram model aims at inducing word representations that are good at predicting the context words surrounding a target word [14]. The following hyperparameters (that were explained in chapter 2) in Table 2 were tuned based on the paper of Mikolov et al. [7], where we have:

Table 2. Skip-gram model hyperparameters for training the textual data of Second Set

Vector size	300
Window size	5
Subsampling	10^{-5}
Negative sampling	5
Epochs	10
Alpha	0.025

The algorithm first constructs a vocabulary from the textual data and then learns the vector representation of words. Train this model took around 48 hours and generate a vocabulary of 2514312 words (that will be cleaned to the 439 words later).

The word embeddings generated by the skip-gram can be tested thanks to different functions that offer Gensim. One of these functions is *most_similar* that finds the top-N most similar words for a given word, and some examples are illustrated in Figure 12. As argued in chapter 2, the skip-gram model can learn implicitly the relationships between words, without providing any supervised information during training. Indeed, the skip-gram model is so powerful that there are different proposals to learn semantic representations through this model.

```
w2v.most_similar('car')
```

```
[('cars', 0.8139374256134033),
 ('automobile', 0.7039523124694824),
 ('truck', 0.6975124478340149),
 ('vehicle', 0.6853656768798828),
 ('roadster', 0.6779319643974304),
 ('SUV', 0.673419177532196),
 ('motorbike', 0.667402446269989),
 ('motorcycle', 0.6509369611740112),
 ('motorcar', 0.6486092805862427),
 ('driver', 0.6451826095581055)]
```

```
w2v.most_similar('robe')
```

```
[('robes', 0.8565139174461365),
 ('cloak', 0.763272762298584),
 ('tunic', 0.7261566519737244),
 ('gown', 0.7151514291763306),
 ('scarf', 0.7033424377441406),
 ('chlamys', 0.6899319291114807),
 ('wig', 0.6896828413009644),
 ('head-dress', 0.680381178855896),
 ('dress', 0.6654574275016785),
 ('garb', 0.6621318459510803)]
```

Figure 12. Some examples for top-N most similar words for a given word

4.3.2 Skip-gram pre-trained model for visual data of Second Set

The reason of use a pre-trained model for visual data is simple: the data (number of sentences) is not large enough so that the skip-gram model cannot generalize well enough, i.e., the word representations will not be good enough at predicting the context words surrounding a target word. Thanks to the Gensim library, we can save the model generated in the training for textual data, and later use that model to train the visual data.

The model also generates 2514312 words, but since that was re-trained for 10 epochs with visual data, the weights were updated, and the vectors changed slightly. As with textual data, these vectors will be cleaned to the 439 words of the First Set.

4.3.3 Data Transformations for Second Set

A crucial point before feeding the model with the generated word representations is to realize all data transformations since it is very critical for success. The algorithms that you may use can be powerful, but without the relevant or right data training, your system may fail to yield ideal results. This process is illustrated in Figure 13.

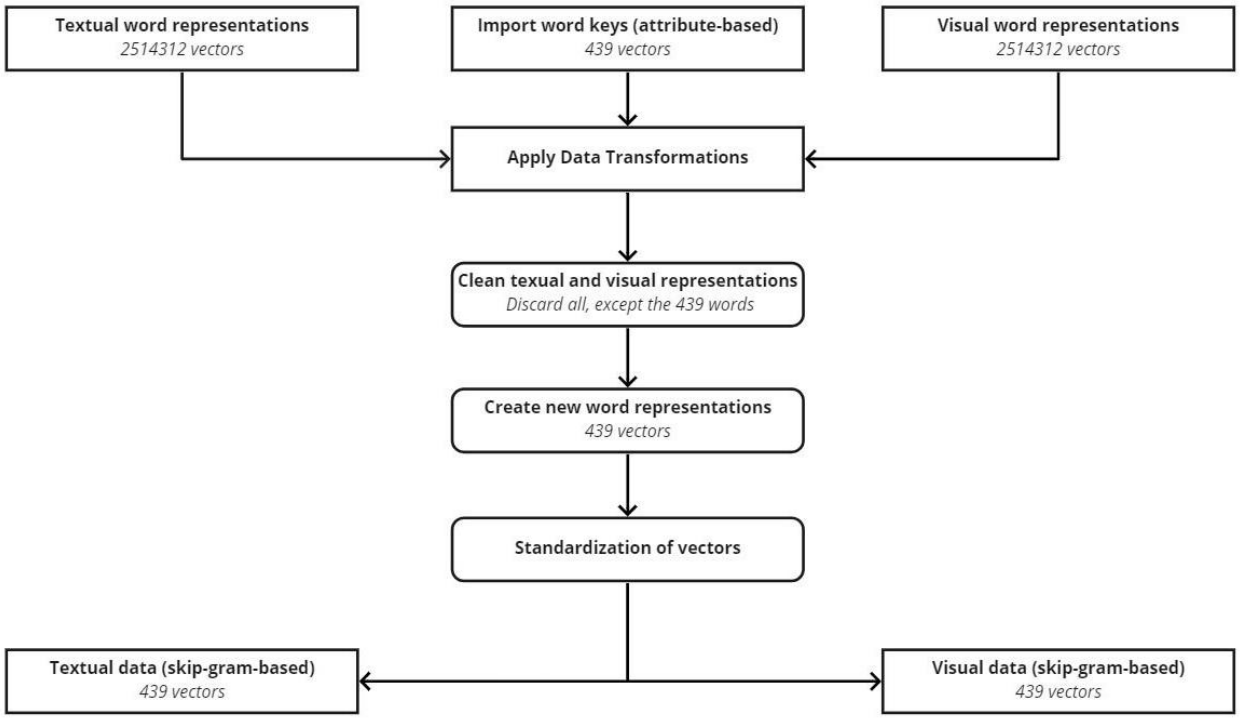


Figure 13. Block diagram of the process of applying the data transformations

The applied transformations were:

1. **Data Cleaning**: The main aim is to identify and remove errors and duplicate data, to create a reliable dataset. This improves the quality of the training data for analytics and enables accurate decision-making. Both textual and visual data were cleaned to have the same 439 words used in the First Dataset.
2. **Feature scaling**: One of the most important transformations, due that ML algorithms do not perform well when the input numerical attributes have very different scales. Both textual and visual were scaled through standardization (gave better results than normalization).

It is important to highlight that all algorithms used in this step were automated to prepare the data for the next time we get new datasets. Also, these algorithms can be applied in future projects, where it will be much easier to extract the word representations and realize the necessary transformations.

4.4 Model Selection

According to classification introduced in [33], they consider two families of strategies to integrate language and perceptual modalities: 1) a *posteriori combination*, where each modality is learned separately and they are integrated afterward, and 2) *simultaneous learning*, where a single representation is learned from raw input data enriched with both modalities.

The proposed model falls in the category of a *posteriori combination* since the model takes as input textual and visual vector representations (which were generated in the previous step), and that each of them is the input of a dedicated two-layer GCN trained to classify words. The word embeddings generated for each GCN are concatenated and feed one last two-layer GCN that generates the final multimodal embeddings. This model is based on HM-DGE [12], and a schematic overview is provided in Figure 14.

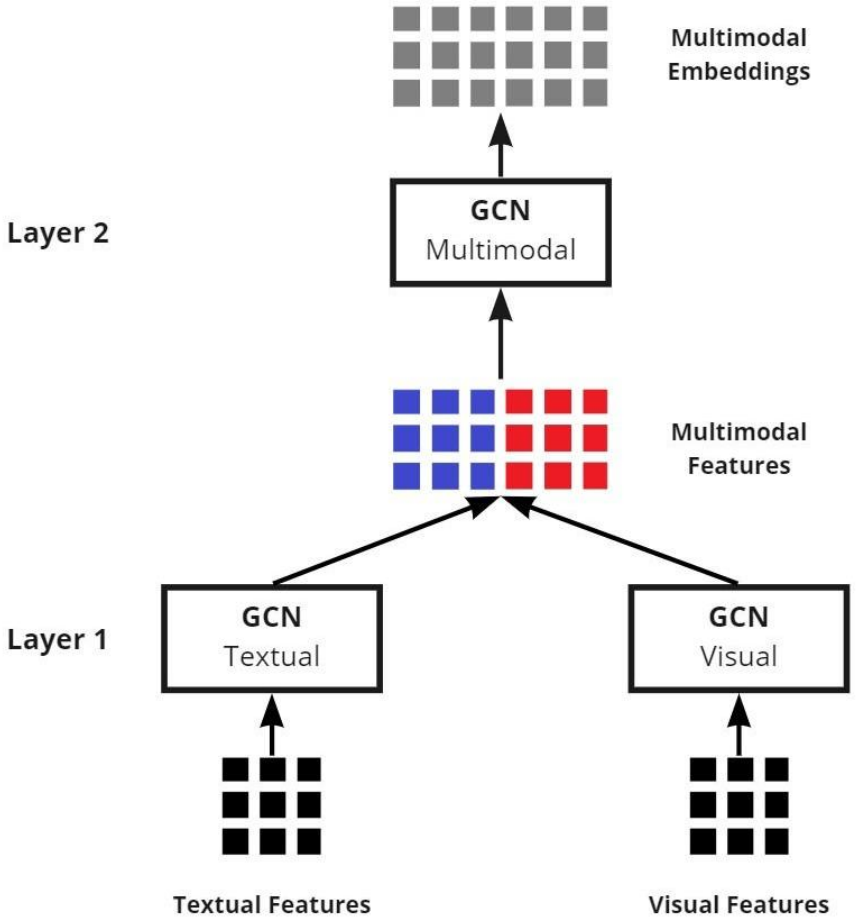


Figure 14. Proposed model for learning semantic representations with the help of Graph Convolutional Networks

Nevertheless, before using both textual and visual data as inputs of each GCN, these word representations must be converted on graphs (nodes and edges). This was possible with the use of PyTorch Geometric [34], which is a geometric deep learning extension library for PyTorch and it consists of various methods for deep learning on graphs and other irregular structures from a variety of published papers.

It is crucial to define what means each hyperparameter in the model, to decide which is the most convenient during the training. These hyperparameters are:

- **Loss function**: The error for the current state of the model must be estimated repeatedly and this function given an input and a target, calculates the loss, i.e., the difference between output and target variable [35]. Each loss function will give different errors for the same prediction, and therefore have an important effect on the performance of the model.
- **Optimizer**: Optimization Algorithms are used to update weights and biases, i.e., the internal parameters of a model to reduce the error [36]. They can be divided into constant learning rate algorithms and adaptive learning rate algorithms.
- **Learning rate**: This hyperparameter controls how much to change the model in response to the estimated error each time the model weights are updated [37]. A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.
- **Dropout**: It is a method for addressing the problem of overfitting. The main idea is to randomly drop units (along with their connections) from the neural network during training and this prevents units from co-adapting too much [38].

The GCNs use *message passing* scheme with cross-entropy as a loss function (this criterion combines LogSoftmax and NLLLoss in one single class, and it is useful when training a classification problem with C classes [39]), Adam as optimizer (Adam has a good convergence speed and quality [24], in addition to being used in most GNN [34]), and a dropout default value of 0.5. Besides, the best results were achieved with a learning rate of 0.01, with other values the quality of the word representations decreased.

The graph structure (edges) of each graph was determined by obtaining the similarity matrix of the data, and with the help of the similarity histogram, it was decided which nodes (word representations) are connected. This will be explained in detail in the next chapter, but how can you expect, the hyperparameter settings will be different for each data type.

The model aims to enhance each unimodal representation individually in Layer 1 and later in Layer 2 map both into a common embedding space by inducing semantic representations that integrate both modalities [12]. However, not only the multimodal embeddings were evaluated, but also the outputs of each GCN and the multimodal features (the result of concatenating both textual and visual embeddings). Besides, another experiment was to swap the adjacency matrix between each modality, i.e., the GCN for the textual data used the adjacency matrix obtained for the visual data, and the GCN for visual data used the adjacency matrix obtained for the textual data.

4.5 Fine-Tune the System

The model in Figure 14 has three GCNs that need to be tuned individually to get better results. The hyperparameters that can be modified are:

- **Number of edges**: As argued in the previous step, the edges of each graph were determined by obtaining the similarity matrix of the data and deciding which nodes are connected, with the help of the similarity histogram. It is crucial to determine the number of edges to ensure the best word embeddings during the training.
- **Embedding shape**: This hyperparameter modifies the size (number of features) of the convolutional layer in the GCN.
- **Epochs**: The epochs define the number of times that the algorithm will work through the entire dataset, i.e., each sample in the dataset has had an opportunity to update the internal model parameters.

To find a great combination of hyperparameters, these were changed to different values and the word embeddings generated by each GCN were evaluated with the benchmark of word similarity used in [4]. This was an iterative process with step 6 (testing a validation) and was applied for both representations (first and second set).

4.6 Testing and Validation

Finally, in this step were evaluated the word representations of each GCN of the model. As argued in chapter 2, a widely adopted way to test vector-based models is to measure how well the model predictions of word similarity correlate with human semantic and visual similarity ratings [12]. This possible with benchmarks that provide gold labels (i.e., human judgment scores) for word pairs, and measuring how well model predictions (cosine similarities) correlate with (mean) human similarity ratings using Spearman's correlation. The higher the correlation is, the more semantic is captured in the embeddings [10].

The benchmark used was the one created by [4], which consists exclusively of concrete nouns and which is useful for the development and evaluation of grounded semantic space models. Besides, it consists of 7576-word pairs and the similarity ratings were obtained using Amazon Mechanical Turk (AMT). To understand the benchmark of word similarity, some word pairs are presented in Table 3.

Table 3. Some word pairs of the benchmark of word similarity used in [4]

word	word	semantic	visual
pants	robe	3.25	1.25
boots	bridge	0	0
crab	frog	1.5	0.25
lantern	racquet	0.25	0
banana	cauliflower	2	0.25
taxi	van	3	2.25
clock	ruler	0.6	0

For example, the cosine similarity between the vector of "pants" and "robe" is computed and so on for all word pairs, and lastly, it is determined how this similarity correlate with human similarity rating (*semantic* and *visual*, that will be called **SemSim** and **VisSim**, respectively). This task serves as a biased indicator of the quality of the embeddings. The results obtained were compared to the model of the state-of-the-art [4]. The next chapter will be indicated the tests performed in each analysis to obtain the results and the number of tests applied in each case.

Chapter 5 **Results and Analysis**

This chapter will show how was the process to fine-tune the system and the performance of the proposed model for both representations (attribute-based and skip-gram-based) when being evaluated in the benchmark of word similarity to know the quality of the embeddings. Also, all tests performed in each analysis to obtain the results and the number of tests applied in each case will be described. An analysis will be performed for each experiment and the chapter concludes with an economic analysis of the project.

Firstly, the model was tuned for the attribute-based representations (both textual and visual), and the results obtained were validated against the results of the state of the art [4]. Later, the model was tuned again for the skip-gram-based representations and as with the first set, the results were validated against the results of the state of the art. Lastly, all results were summarized and compared against them.

5.1 Experiments with First Set (attribute-based)

As argued in the previous chapter, before using both textual and visual data as inputs of each GCN, these word representations must be converted on graphs (nodes and edges). Once you have the graph, results can be obtained by modifying GCN hyperparameters. It will be shown the results for the textual data, visual data, and finally multimodal data (concatenating textual and visual embeddings) according to the model in Figure 14.

5.1.1 Textual data

To create the graph, it was computed the similarity histogram and similarity matrix (also called affinity matrix) of the textual data through cosine similarity, and the results obtained appear in Figure 15. The main goal is to use this method to determine the adjacency matrix, defining the *optimal cosine similarity*. For example, if the optimal cosine similarity is 0.2, all word representations (nodes) that have a similarity between them greater than 0.2 are going to be connected.

A similarity matrix is an essential statistical technique used to organize the mutual similarities between a set of vectors. These similarity measures can be interpreted as the probability that two words (vectors) are related, then, if two words have close features, then their cosine

similarity score will be much closer to 1 (lighter points) than two words with a lot of difference between them (darker points). The similarity histogram organizes a group of vectors into user-specified ranges, that in this case, the y-axis indicates the percentage of occurrences and the x-axis the cosine similarity.

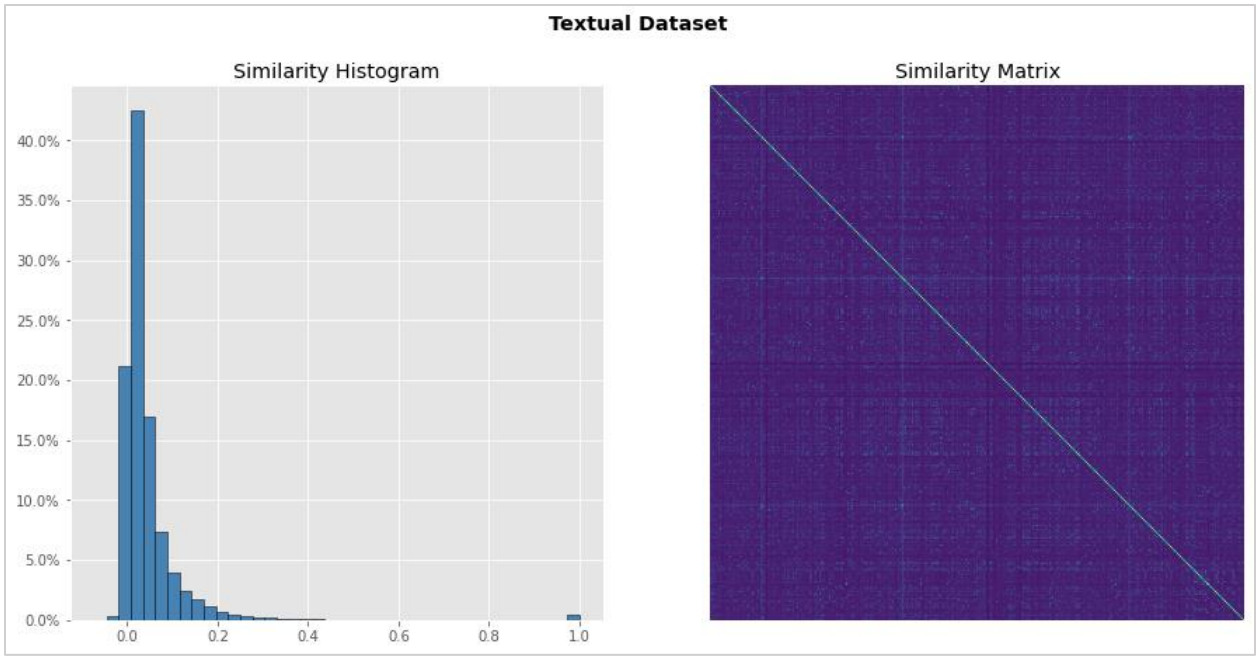


Figure 15. Similarity histogram and Similarity matrix for textual data of the First Set (attribute-based) to determine the adjacency matrix

With this information, we can define the optimal cosine similarity to create a graph in which only the words (vectors) with the higher similarity between them will be connected. By assigning a numerical value to the abstract concept of similarity, the similarity matrix allows ML algorithms to mimic human logic by making educated guesses about what information is related and how similar they are. Besides, enables systems to work with data from unlabeled or corrupted data sets with human-like intuition.

The hyperparameters of the GCN for textual data were modified to different values. Each combination gives as a result new word embeddings that were evaluated in the benchmark of word similarity (SemSim and VisSim). Besides, the loss function was cross-entropy, the optimizer was Adam with a learning rate of 0.01 and a dropout value of 0.5. The results are presented in Table 4.

Table 4. Spearman's correlation of model predictions for textual data of the First Set (attribute-based). The best results are highlighted in bold for the columns of SemSim and VisSim

Hyperparameter	Embedding shape	Optimal Cosine similarity	Epochs	SemSim	VisSim
Embedding shape	**10**	0.1	150	0.74	0.54
	50	0.1	150	**0.75**	0.56
	100	0.1	150	**0.75**	0.56
	150	0.1	150	**0.75**	0.56
	250	0.1	150	**0.75**	**0.57**
Number of edges	100	**0.05**	150	0.70	0.50
	100	**0.075**	150	**0.75**	0.54
	100	**0.15**	150	0.73	**0.57**
	100	**0.2**	150	0.61	0.54
Epochs	100	0.1	**15**	0.73	0.53
	100	0.1	**50**	**0.75**	0.55
	100	0.1	**250**	**0.75**	**0.57**
	100	0.1	**500**	0.74	**0.57**

The number of tests realized for this data was a total of 13. The embedding shape of GCN was changed to five different values, the number of edges (determined by the optimal cosine similarity) was changed to four different values and the number of epochs was changed to four different values. The best results were obtained for 1) embedding shape of 250, optimal cosine similarity of 0.1 and 150 epochs, and 2) embedding shape of 100, optimal cosine similarity of 0.1, and 250 epochs. The first hyperparameter settings were chosen and the loss and train accuracy of the GCN is illustrated in Figure 16.

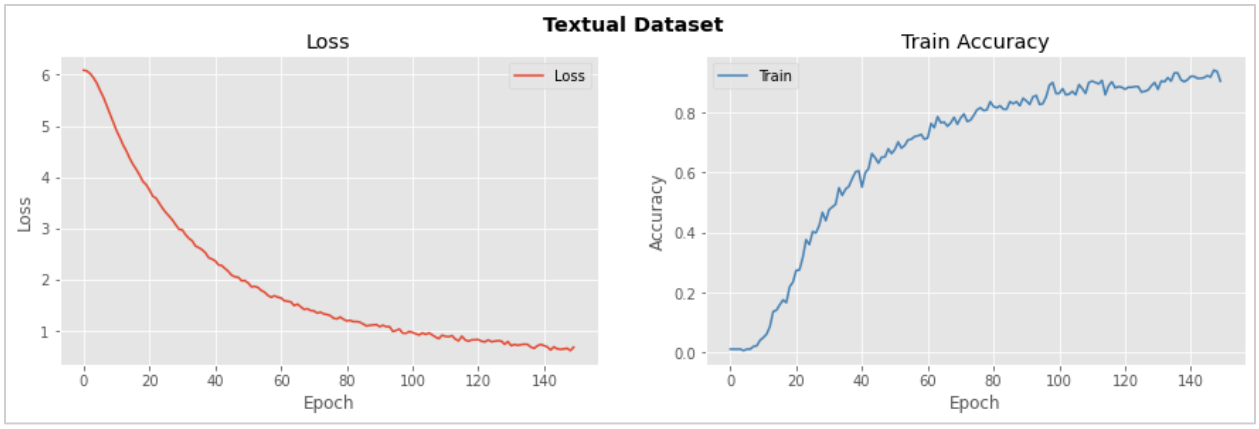


Figure 16. Loss and train accuracy of the GCN for textual data of the First Set (attribute-based)

Although the loss has a high correlation with the quality of the embeddings, both are not perfectly aligned. The classification is the Pretext Task, and measuring the quality of the embeddings is the Downstream Task [40]. This is known as self-supervised learning, which is a particular case of unsupervised learning.

5.1.2 Visual data

As for textual data, the graph for this data was created computing the similarity histogram and similarity matrix of the visual data through cosine similarity and the results obtained appear in Figure 17. If two words have close features, then their cosine similarity score will be much closer to 1 (lighter points) than two words with a lot of difference between them (darker points). With this method, it is possible to determine the adjacency matrix to create a graph in which only the words (vectors) with the higher similarity between them will be connected.

The hyperparameters of the GCN for visual data were modified to different values. Each combination gives as a result new word embeddings that were evaluated in the benchmark of word similarity. Besides, the loss function was cross-entropy, the optimizer was Adam with a learning rate of 0.01 and a dropout value of 0.5. The results are presented in Table 5.

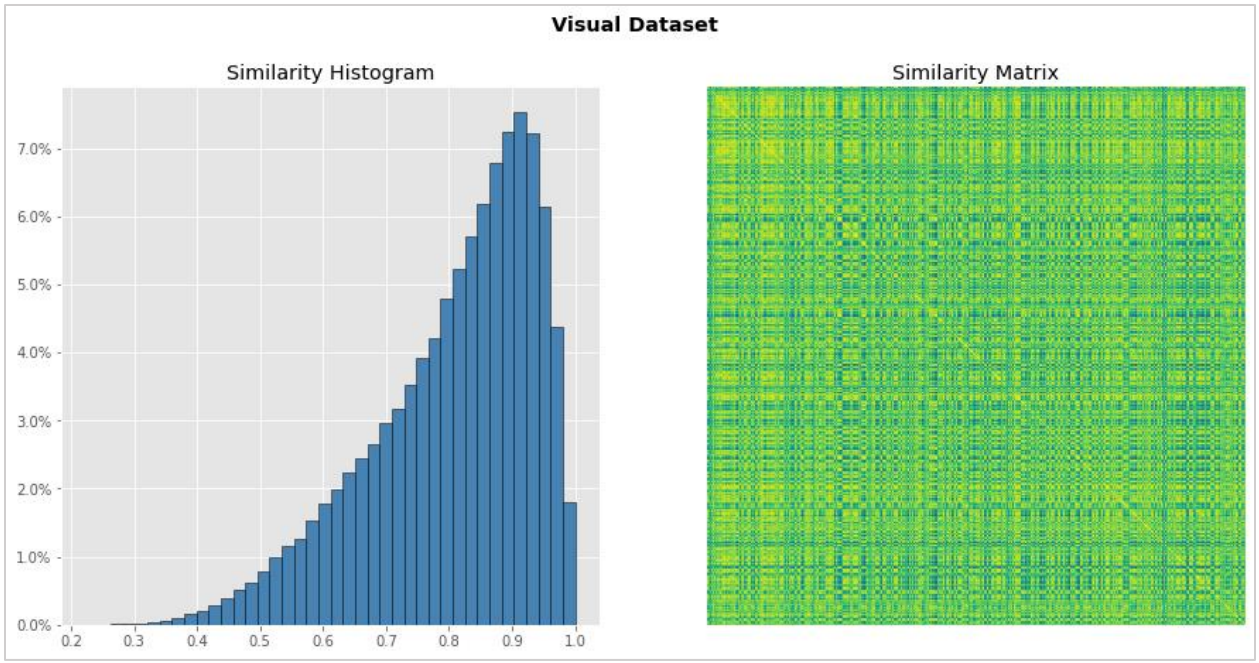


Figure 17. Similarity histogram and Similarity matrix for visual data of the First Set (attribute-based) to determine the adjacency matrix

Table 5. Spearman's correlation of model predictions for visual data of the First Set (attribute-based). The best results are highlighted in bold for the columns of SemSim and VisSim

Hyperparameter	Embedding shape	Optimal Cosine similarity	Epochs	SemSim	VisSim
Embedding shape	**10**	0.95	150	0.56	0.48
	50	0.95	150	0.64	0.53
	100	0.95	150	0.63	0.53
	150	0.95	150	0.64	0.54
	250	0.95	150	0.64	0.55
Number of edges	250	**0.9**	150	0.54	0.44
	250	**0.925**	150	0.60	0.50
	250	**0.975**	150	0.64	0.57
Epochs	250	0.95	**15**	0.58	0.49
	250	0.95	**50**	0.62	0.52
	250	0.95	**250**	**0.65**	0.56
	250	0.95	**500**	**0.65**	0.57
	250	0.95	**2500**	**0.65**	**0.59**

The number of tests realized for this data was a total of 13. The embedding shape of GCN was changed to five different values, the number of edges (determined by the optimal cosine similarity) was changed to three different values and the number of epochs was changed to five different values. The best results were obtained for: embedding shape of 250, optimal cosine similarity of 0.95, and 2500 epochs.

5.1.3 Swapping the adjacency matrix

As argued in chapter 4, another experiment was to swap the adjacency matrix between each modality, i.e., the GCN for the textual data used the adjacency matrix obtained for the visual data, and the GCN for visual data used the adjacency matrix obtained for the textual data (this will be called *swap edges*). This was proposed in [12] with the HM-DGE model. The results are presented in Table 6. The hyperparameter settings are the same both textual and visual.

Table 6. Spearman's correlation of model predictions for textual and visual data of the First Set, and their respective graphs swapping the adjacency matrix

Model	SemSim	VisSim
GCN Textual	0.75	0.57
GCN Textual - Swap edges	0.63	0.60
GCN Visual	0.65	0.59
GCN Visual - Swap edges	0.71	0.52

5.1.4 Multimodal data

Once the GCNs for the textual and visual data has been defined, the word representations generated of both are concatenated, according to Figure 14. The adjacency matrix for this graph with new data (multimodal features) was created by computing the similarity histogram and similarity matrix through cosine similarity, in which only the vectors with the higher similarity between them will be connected. The results obtained appear in Figure 18.

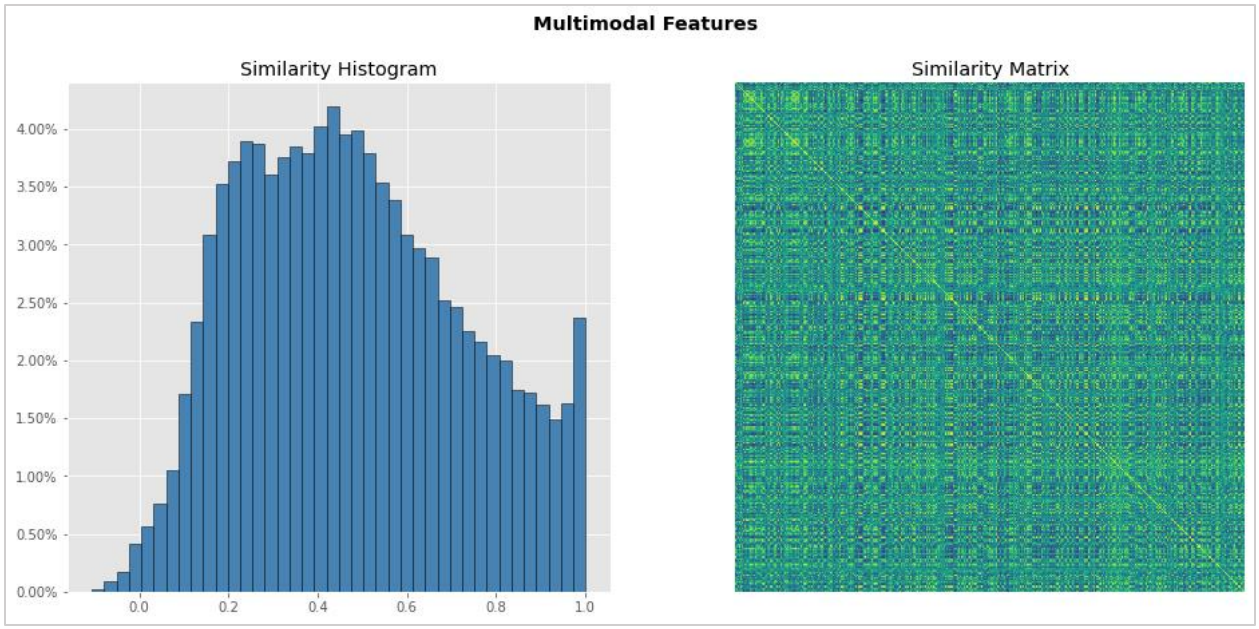


Figure 18. Similarity histogram and Similarity matrix for multimodal data of the First Set (attribute-based) to determine the adjacency matrix

The best combination of hyperparameters of the GCN for multimodal features was determined through grid search, giving as result: embedding shape of 150, optimal cosine similarity of 0.9 and 500 epochs. Besides, the loss function was cross-entropy, the optimizer was Adam with a learning rate of 0.01 and a dropout value of 0.5. The Spearman's correlations for the model predictions are: 0.72 (SemSim) and 0.59 (VisSim). Using as input of this GCN the word embeddings (both textual and visual) that swapping their adjacency matrix, the Spearman's correlations are 0.74 (SemSim) and 0.58 (VisSim).

The results in terms of Spearman's correlation for each GCN in the proposed model were summarized in Table 7 and compared against the results (SAE model) of the state of the art [4]. *Semantic Similarity* refers to SemSim and *Visual Similarity* refers to VisSim. Each column corresponds to a unimodal (textual (T) or visual (V)) or multimodal (T+V) representation. For example, in *Semantic Similarity*: **T** refers to SemSim value for textual data, **V** to SemSim value for visual data, and **T+V** to SemSim value for multimodal data.

Table 7. Comparative results in terms of Spearman's correlations between model predictions of each GCN of the model (for the First Set) and the state-of-the-art

Models	Semantic Similarity			Visual Similarity		
	T	V	T+V	T	V	T+V
SAE (attribute-based)	0.67	0.61	0.72	0.55	**0.60**	**0.65**
GCN (attribute-based)	**0.75**	0.65	0.72	0.57	0.59	0.59
			0.73*			0.62*
GCN (attribute-based) - Swap edges	0.63	**0.71**	**0.74**	**0.60**	0.52	0.58
			0.73*			0.57*

In rows where only one value appears with an asterisk (*), it refers to Spearman's correlation of the multimodal features, i.e., the concatenation of the word embeddings for both textual and visual, but without to be trained for the GCN multimodal.

The proposed model trained on attribute-based input performs best on semantic similarity (against the SAE model) than on visual similarity. The SAE model is better with multimodal representations, but this is not the case with the GCN model, where the unimodal representations get higher Spearman's correlations. Besides, the multimodal features ($\rho = 0.73$ and $\rho = 0.62$) obtain better results than the multimodal embeddings ($\rho = 0.72$ and $\rho = 0.59$), for the case without swap edges. This suggests that the GCN (layer 2) of the proposed model Figure 14 does not contribute in enhance the quality of the word embeddings. Further, as one would expect, textual data dominates visual data when modeling semantic similarity, while visual data dominates textual data when modeling visual similarity ratings.

The highest semantic similarity ($\rho = 0.75$) was reached with the output of the GCN trained on textual data, whereas the highest visual similarity ($\rho = 0.65$) was reached for the bimodal SAE. For the GCN the highest visual similarity ($\rho = 0.62$) was reached with the multimodal features (without being trained for the GCN of layer 2).

5.2 Experiments with Second Set (skip-gram based)

As for First Set, before using both textual and visual data as inputs of each GCN, these word representations must be converted on graphs (nodes and edges). Once you have the graph,

results can be obtained by modifying GCN hyperparameters. It will be shown the results for the textual data, visual data, and finally multimodal data according to the model in Figure 14.

5.2.1 Textual data

To create the graph, it was computed the similarity histogram and similarity matrix of the textual data through cosine similarity, and the results obtained appear in Figure 19. Defining the optimal cosine similarity, the adjacency matrix can be determined. For example, if the optimal cosine similarity is 0.1, all word representations (nodes) that have a similarity between them greater than 0.1 are going to be connected.

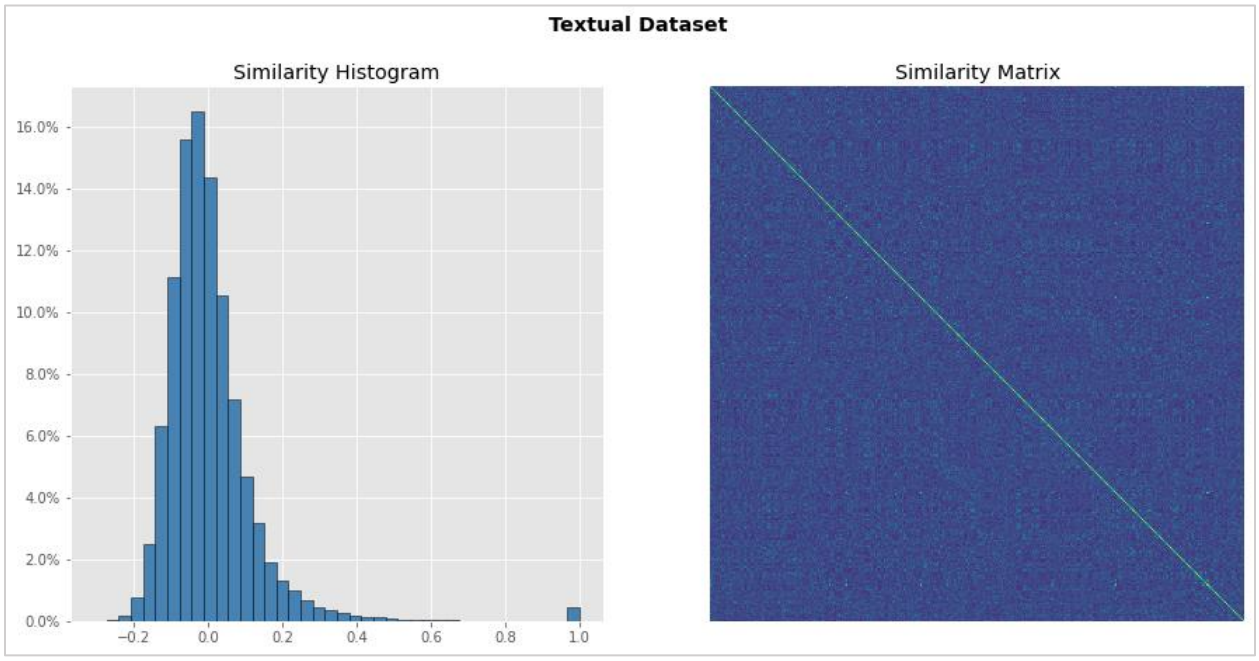


Figure 19. Similarity histogram and Similarity matrix for textual data of the Second Set (skip-gram-based) to determine the adjacency matrix

The hyperparameters of the GCN for textual data were modified to different values. Each combination gives as a result new word embeddings that were evaluated in the benchmark of word similarity (SemSim and VisSim) used in the state-of-the-art [4]. The results obtained are presented in Table 8.

The number of tests realized for this data was a total of 17. The embedding shape of GCN was changed to six different values, the number of edges (determined by the optimal cosine

similarity) was changed to six different values and the number of epochs was changed to five different values. The best result was obtained for: embedding shape of 300, optimal cosine similarity of 0.125 and 150 epochs. Besides, the loss function was cross-entropy, the optimizer was Adam with a learning rate of 0.01 and a dropout value of 0.5.

Table 8. Spearman's correlation of model predictions for textual data of the Second Set (skip-gram-based). The best results are highlighted in bold for the columns of SemSim and VisSim

Hyperparameter	Embedding shape	Optimal Cosine similarity	Epochs	SemSim	VisSim
Embedding shape	**30**	0.125	150	0.73	0.55
	70	0.125	150	0.73	**0.56**
	100	0.125	150	0.73	0.55
	150	0.125	150	0.73	0.55
	250	0.125	150	0.73	0.55
	300	0.125	150	**0.74**	**0.56**
Number of edges	50	**0.075**	150	0.71	0.52
	50	**0.1**	150	0.73	0.53
	50	**0.125**	150	**0.74**	0.55
	50	**0.15**	150	0.73	**0.56**
	50	**0.175**	150	0.73	**0.56**
	50	**0.2**	150	0.71	0.55
Epochs	300	0.125	**10**	0.72	0.53
	300	0.125	**50**	0.73	0.55
	300	0.125	**100**	0.73	0.55
	300	0.125	**250**	0.73	0.55
	300	0.125	**500**	0.71	**0.56**

5.2.2 Visual data

As for textual data, the graph was created computing the similarity histogram and similarity matrix of the visual data through cosine similarity and the results obtained appear in Figure 20. With this method, it is possible to determine the adjacency matrix.

The hyperparameters of the GCN for visual data were modified to different values. Each combination gives as result new word embeddings that were evaluated in the benchmark of word similarity, and the results are presented in Table 9.

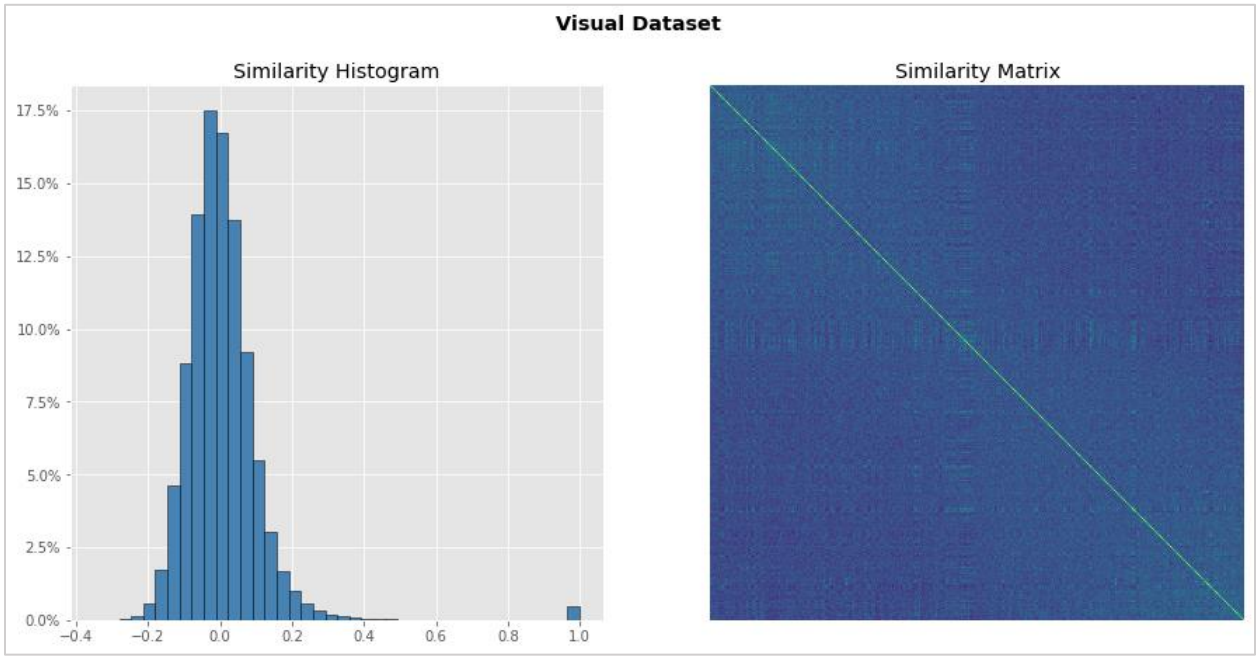


Figure 20. Similarity histogram and Similarity matrix for visual data of the Second Set (skip-gram-based) to determine the adjacency matrix

The number of tests realized for this data was a total of 18. The embedding shape of GCN was changed to five different values, the number of edges was changed to six different values and the number of epochs was changed to seven different values. The best results were obtained for: embedding shape of 100, optimal cosine similarity of 0.125, and 1000 epochs. Besides, the loss function was cross-entropy, the optimizer was Adam with a learning rate of 0.01 and a dropout value of 0.5.

Table 9. Spearman's correlation of model predictions for visual data of the Second Set (skip-gram-based). The best results are highlighted in bold for the columns of SemSim and VisSim

Hyperparameter	Embedding shape	Optimal Cosine similarity	Epochs	SemSim	VisSim
Embedding shape	**30**	0.125	150	0.46	0.35
	70	0.125	150	0.46	0.35
	150	0.125	150	0.45	0.34
	250	0.125	150	0.45	0.34
	300	0.125	150	0.44	0.34
Number of edges	100	**0.075**	150	0.40	0.30
	100	**0.1**	150	0.43	0.32
	100	**0.125**	150	0.47	0.35
	100	**0.15**	150	0.45	0.35
	100	**0.175**	150	0.46	0.36
	100	**0.2**	150	0.43	0.34
Epochs	100	0.125	**10**	0.42	0.31
	100	0.125	**50**	0.45	0.33
	100	0.125	**100**	0.46	0.34
	100	0.125	**250**	0.47	0.35
	100	0.125	**500**	0.46	0.35
	100	0.125	**1000**	**0.48**	**0.37**
	100	0.125	**2500**	0.47	**0.37**

5.2.3 Swapping the adjacency matrix

As for the First Set, another experiment was to swap the adjacency matrix between each modality. The results are presented in Table 10. The hyperparameter settings are the same both textual and visual.

Table 10. Spearman's correlation of model predictions for textual and visual data of the Second Set, and their respective graphs swapping the adjacency matrix

Model	SemSim	VisSim
GCN Textual	0.74	0.56
GCN Textual - Swap edges	0.62	0.45
GCN Visual	0.48	0.37
GCN Visual - Swap edges	0.73	0.56

5.2.4 Multimodal data

Once the GCNs for the textual and visual data has been defined, the model predictions are concatenated, according to Figure 14. The adjacency matrix for this graph with new data (multimodal features) was created by computing the similarity histogram and similarity matrix of the visual data through cosine similarity, and the results obtained appear in Figure 21.

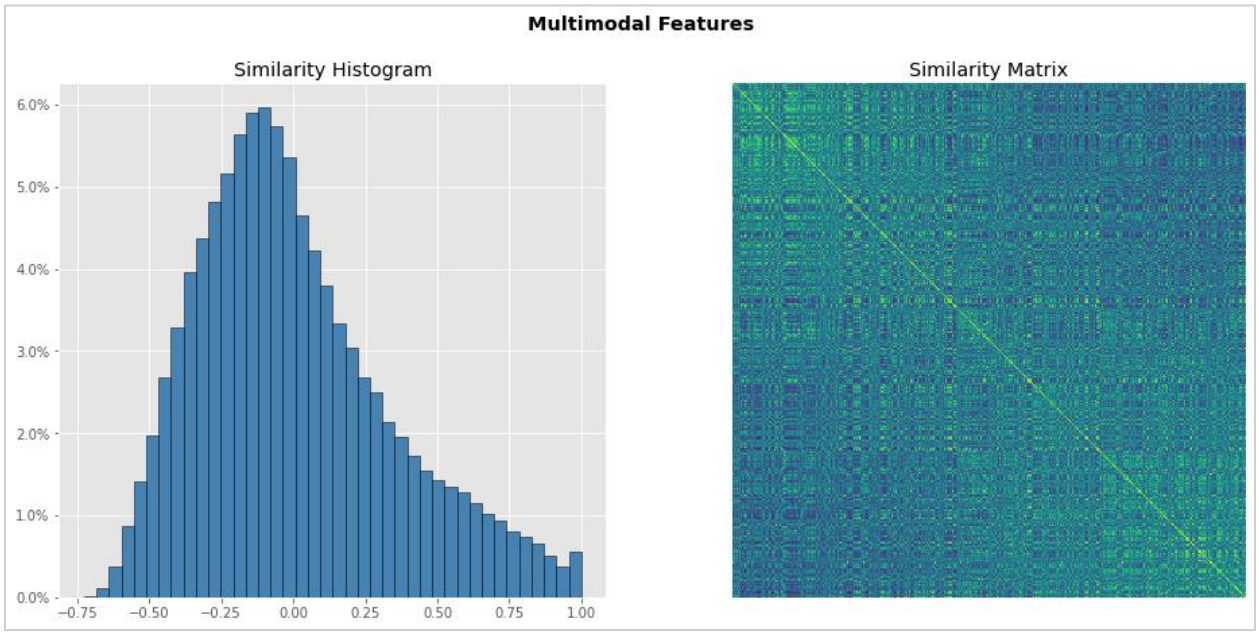


Figure 21. Similarity histogram and Similarity matrix for multimodal data of the Second Set (skip-gram-based) to determine the adjacency matrix

Through grid search, the best combination of hyperparameters of the GCN for multimodal features was determined, giving as result: embedding shape of 400, optimal cosine similarity of

0.5 and 500 epochs. Besides, the loss function was cross-entropy, the optimizer was Adam with a learning rate of 0.01 and a dropout value of 0.5. The Spearman's correlations for the model predictions are: 0.63 (SemSim) and 0.44 (VisSim). Using as input of this GCN the word embeddings that swapping their adjacency matrix, Spearman's correlations are 0.69 (SemSim) and 0.51 (VisSim).

The results in terms of Spearman's correlation for each GCN in the proposed model were summarized in Table 11 and compared against the results (SAE model) of the state of the art [4]. *Semantic Similarity* refers to SemSim and *Visual Similarity* refers to VisSim. Each column corresponds to a unimodal (textual (T) or visual (V)) or multimodal (T+V) representation. For example, in *Visual Similarity*: **T** refers to VisSim value for textual data, **V** to VisSim value for visual data, and **T+V** to VisSim value for multimodal data.

Table 11. Comparative results in terms of Spearman's correlations between model predictions of each GCN of the model (for the Second Set) and the state-of-the-art. The best results are highlighted in bold

Models	Semantic Similarity			Visual Similarity		
	T	V	T+V	T	V	T+V
SAE (attribute-based)	0.67	0.61	0.72	0.55	**0.60**	**0.65**
GCN (skip-gram-based)	**0.74**	0.48	0.63	**0.56**	0.37	0.44
			0.73*			0.55*
GCN (skip-gram-based) - Swap edges	0.62	**0.73**	0.69	0.45	0.56	0.51
			0.73*			0.55*

In rows where only one value appears with an asterisk (*), it refers to Spearman's correlation of the multimodal features, i.e., the concatenation of the word embeddings for both textual and visual, but without to be trained for GCN multimodal.

Same as with the First Set, the proposed model trained on skip-gram-based input performs best on semantic similarity (against the SAE model) than on visual similarity. The SAE model is better with multimodal representations, but the GCN model gets higher Spearman's correlations, with the unimodal representations. Also, the multimodal features ($\rho = 0.73$ and $\rho = 0.55$) obtain better results than the multimodal embeddings ($\rho = 0.63$ and $\rho = 0.44$), for the case without swap edges. This suggests that the GCN (layer 2) of the proposed model in Figure 14 does not contribute in enhance the quality of the word embeddings. Further, as one would expect, textual data

dominates visual data when modeling semantic similarity, while visual data dominates textual data when modeling visual similarity ratings.

The highest semantic similarity (ρ = 0.74) was reached with the output of the GCN trained on textual data, whereas the highest visual similarity (ρ = 0.65) was reached for the bimodal SAE. For the GCN the highest visual similarity (ρ = 0.56) was reached with the output of the GCN trained on textual data and with the output of the GCN trained on visual data (with the adjacency matrix of the textual data).

5.3 Summarizing all experiments

The results in terms of Spearman's correlation for each GCN in the proposed model (both for the First and Second Set) were summarized in Table 12 and compared against the result obtained for the model of the state-of-the-art [4].

Table 12. Comparative results in terms of Spearman's correlations between model predictions of each GCN of the model (for the First and Second Set) and the state-of-the-art. The best results are highlighted in bold

Models	Semantic Similarity			Visual Similarity		
	T	V	T+V	T	V	T+V
SAE (attribute-based)	0.67	0.61	0.72	0.55	**0.60**	**0.65**
GCN (attribute-based)	**0.75**	0.65	0.72	0.57	0.59	0.59
			0.73*			0.62*
GCN (attribute-based) - Swap edges	0.63	0.71	**0.74**	**0.60**	0.52	0.58
			0.73*			0.57*
GCN (skip-gram-based)	0.74	0.48	0.63	0.56	0.37	0.44
			0.73*			0.55*
GCN (skip-gram-based) - Swap edges	0.62	**0.73**	0.69	0.45	0.56	0.51
			0.73*			0.55*

In rows where only one value appears with an asterisk (*), it refers to Spearman's correlation of the multimodal features, i.e., the concatenation of the word embeddings for both textual and visual, but without to be trained for GCN multimodal.

After analyzing the First and Second Set with the proposed model, it is determined that performs best on semantic similarity than on visual similarity, in relation to the SAE model. The SAE model is better with multimodal representations, but the GCN model gets higher Spearman's correlations, with the unimodal representations. The highest semantic similarity ($\rho = 0.75$) was reached with the output of the GCN trained on textual data (attribute-based), whereas the highest visual similarity ($\rho = 0.65$) was reached for the bimodal SAE. For the GCN the highest visual similarity ($\rho = 0.60$) was reached with the output of the GCN trained on textual data (attribute-based) with the adjacency matrix of the visual data.

It is important to notice that the visual data obtained with the skip-gram model get lower results in comparison with the attribute-based visual data. However, when comparing the results obtained with the Second Set, one must be careful, since the most logical thing is to only compare the results obtained with the trained model with the same input representations (attribute-based). The Second Set was generated with the data used in [10] and it was interesting to test the word representations obtained from the skip-gram model.

5.4 Model Validation

To validate the proposed model illustrated in Figure 14, in Table 13 and Table 14 appear the results in terms of Spearman's correlations between the model predictions (new word embeddings) of each GCN of layer 1 and the initial representations (attribute-based and skip-gram-based raw data). It can be seen that both GCN Textual and GCN Visual improve the performance when compared with initial representations (Textual data and Visual data of each set). This validates the capabilities of the individual GCNs to enhance the vectors and to learn meaningful word representations.

Table 13. Comparative results in terms of Spearman's correlations between model predictions of each GCN of layer 1 and the First Set (attribute-based)

Concept	SemSim	VisSim
Textual data (attribute-based)	0.64	0.51
GCN Textual	0.75	0.57
GCN Textual - Swap edges	0.63	0.60
Visual data (attribute-based)	0.62	0.57
GCN Visual	0.65	0.59
GCN Visual - Swap edges	0.71	0.52

Table 14. Comparative results in terms of Spearman's correlations between model predictions of each GCN of layer 1 and the Second Set (skip-gram-based)

Concept	SemSim	VisSim
Textual data (skip-gram-based)	0.69	0.56
GCN Textual	0.74	0.56
GCN Textual - Swap edges	0.62	0.45
Visual data (skip-gram-based)	0.46	0.39
GCN Visual	0.48	0.37
GCN Visual - Swap edges	0.73	0.56

The GCN for the attribute-based representations (Table 13) enhances the initial representations up to 17% in SemSim and 15% in VisSim. On the other hand, the GCN for skip-gram-based representations (Table 14) enhances the initial representations up to 7% in SemSim and keeping the same value in VisSim.

However, how was demonstrated in Table 12, the GCN Multimodal (T+V) does not improve the word representations obtained from the GCNs Textual and Visual. It could be considered that it is better just to concatenate the model predictions from layer 1 (multimodal features) instead that use the GCN of layer 2 to get new word embeddings (multimodal embeddings), but it is necessary to do more experiments with different datasets, hyperparameters, and benchmarks to conclude that.

5.5 Validation with Robotics Systems

Although the idea of this project was to apply the results obtained in IRI robotic systems, a significant number of challenges still need to be addressed before we can create a robot that can learn and understand language through natural interactions with human participants (despite an extensive and growing body of research). According to [3], we must tackle the following problems (excluding the topics that have been intensively studied, e.g., robot audition and vision, nonverbal communication in social robotics, and so forth):

- Logic probabilistic programming and learning distributed semantics.
- Unsupervised syntactic parsing with grounding phrases and predicates.
- Category and concept formation.
- Metaphorical expressions.
- Affordance and action learning.
- Pragmatics and social language.
- Dataset, simulator, and competition.

The option of testing in a robotic simulator was considered, but a conventional robotics simulator tends to focus on physical simulation to provide a reliable evaluation of the behavior and physical control of robots [3], and since the main goal is precision, it takes a long time to get a simulation result. Besides, the language acquisition process has a strong relationship not only between simple actions such as "forward" or "turn right", but also more complex actions using a realistic body and limbs.

These results will play an important role in the evaluation of learning and recognition performance. As demonstrated in [5], robots using language must learn how words are grounded on the noisy perceptual world in which a robot operates, and natural language systems can benefit from the rich contextual information provided by sensor data about the world. Understanding and acquiring languages in real-world environments is an important task for the robotics systems of the future because our social environment is filled with rich and dynamic linguistic interactions.

5.6 Economic Analysis

The economic analysis goes beyond something financial and involves the efficient use of resources. In this part will be mentioned the different costs of the project (personnel, equipment, and energetical) and what this project is expected to contribute to the scientific production of IRI.

5.6.1 Personnel cost

This project was carried out with the help of a researcher from the IRI. The details of work by the personnel involved in this project can be summarized as follows:

- Four months (16 weeks) of personal work with a daily dedication between 5 and 7 hours per day, 5 days per week.
- Assistance by the researcher with an average attendance of 2 hours per week each.

The personnel cost of the project can be computed assuming an hourly rate of 20 €/h for the student (as a technical engineer) and 50 €/h for the researchers (approximate pay rate for external hiring of a CSIC researcher. The breakdown of the cost is presented in Table 15.

.

Table 15. Personnel cost breakdown of the project

	Dedication (h)	**Hourly rate** (€/h)	**Cost** (€)
Technical Engineer (student)	520	20	10400
Researcher (advisor)	32	50	1600
		Total	**12000**

5.6.2 Equipment cost

The computer used costs 600 euros and the GPU used 1000 euros. Besides, none of these products were obtained exclusively for this project and it is expected to be operated during all their lifetime. For that reason, the equipment cost will be computed by dividing the total worth of each hardware by its lifespan and multiplying it by the time that it has been used for this project. The breakdown of the equipment cost is presented in Table 16.

Table 16. Equipment cost breakdown of the project

Equipment	**Total cost** (€)	**Expected lifespan** (months)	**Time of usage** (months)	**Cost** (€)
Main Computer	600	60	4	40
GPU (Tesla k40c)	1000	60	4	66.67
			Total	**106.67**

5.6.3 Energetical cost

The energetical cost of this project comes from the electricity usage of the different types of equipment used and of the workplace. To compute the energetical cost, a rate of 0.11 €/kWh will be used (extracted by actual rates from Endesa [41]). Consumption is assumed based on the average annual consumption of 52.5 kWh/m^2 (approximately, the office has a surface of 60 m^2). The energetical cost of the project is presented in Table 17.

Table 17. Electricity usage and energetical cost breakdown of the project

Source	**Power usage** (W)	**Usage time** (h)	**Energy** (kWh)	**Cost** (€)
Main Computer	220	520	114.4	12.58
GPU (Tesla k40c)	245	96	23.52	2.59
Workplace	360	520	187.2	20.59
		Total	**137.92**	**35.76**

5.6.4 Total cost

The total cost of the project (summarizing all previous costs) is presented in Table 18.

Table 18. The total cost of the project

Concept	**Cost** (€)
Personnel cost	12000
Equipment cost	106.67
Energetical cost	35.76
Total	**12142.43**

5.6.5 Scientific production

A summary of publications made by IRI in the last 10 years [42] is presented in Table 19, and in Figure 22 this information is presented in a chart.

Table 19. Summary of publications made by IRI in the last 10 years

Publications	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Books	4	1	0	0	1	0	2	3	4	1
Journal Articles	29	28	32	39	41	44	54	54	64	54
Book Chapters	1	0	2	2	7	2	12	2	8	2
Conference Articles	43	51	54	67	47	53	70	81	71	42
Technical Reports	4	0	3	5	6	5	5	1	4	4
Total	**81**	**80**	**91**	**113**	**102**	**104**	**143**	**141**	**151**	**103**

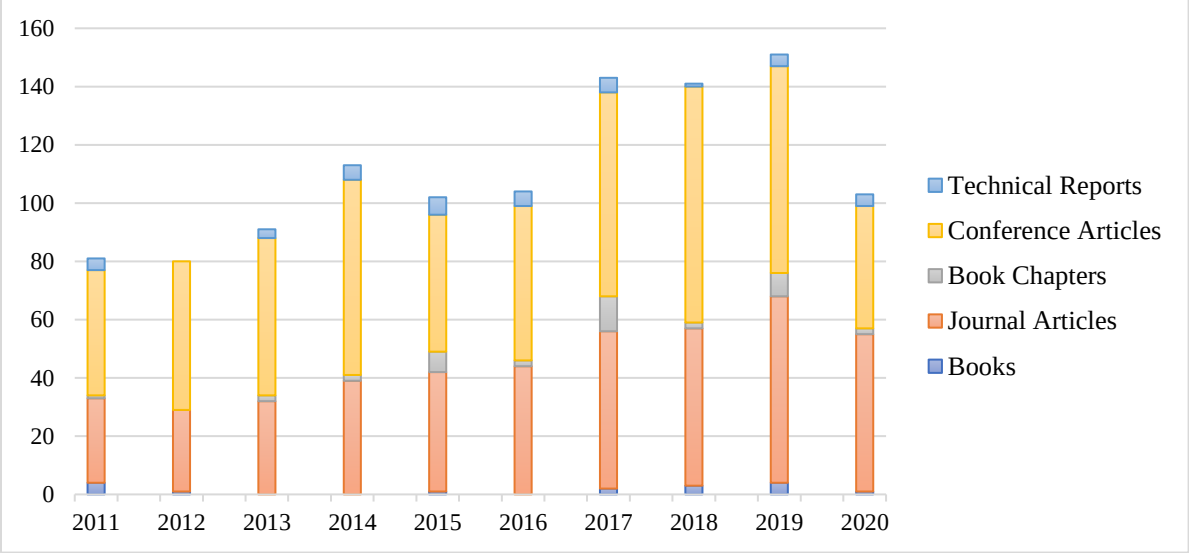


Figure 22. Summary of publications made by IRI (chart)

IRI's scientific production is published in the top journals in the fields of interest. In addition, the researchers present their work in the top conferences, promoting the exchange of ideas with research institutions and related companies. This project (and those that will be generated from it) for now is an unpublished work, but it is expected to contribute to the scientific production of the IRI, which in bibliometric terms has grown steadily for more than a decade and has continued during this period.

Chapter 6 **Conclusions and Recommendations**

This chapter includes the conclusions obtained from the project and the results regarding its objectives, to determine whether the project was successful or not. Lastly, a section of recommendations is included to prepare the way for the approach of future works.

6.1 Conclusions

Regarding the project objectives, the influential characteristics helped to frame the problem and to consider all necessary steps to find the best solution to the project. Besides, the process of acquiring and prepare the data (through different functions and transformations) to the model was automated. Even these algorithms can be used the next time we get a new fresh dataset or in future projects. Lastly, the programming logic was established to proposed a model that addresses the problem of learning semantic representations, and these representations were validated through a benchmark of word similarity and comparing the results against the state-of-the-art [4].

In detail, a model that uses Graph Convolutional Networks to learn semantic representations of objects by simultaneously combining textual and visual modalities was presented. The first layer encodes unimodal representations, and the second layer integrates these enhanced unimodal representations into one. The two representations of the dataset (attribute-based and skip-gram-based) were encoded as word embeddings and obtained from text and image data. To the best of our knowledge, the proposed model is novel in its use of GCNs to enhance the quality of word representations.

Comparative results on the task of simulation of word similarity show that the proposed model outperforms baseline in some cases (mainly in semantic similarity). The GCNs of layer 1 validates the capabilities of the unimodal representations to learn meaningful word representations and enhance the embeddings up to 17% in SemSim and 15% in VisSim for the attribute-based representations and up to 7% in SemSim and keeping the same value in VisSim for the skip-gram-based representations. However, the GCN of layer 2 (multimodal) does not improve the quality of the word representations.

6.2 Recommendations

In future works, it should be tested with different parameters and data when creating the word embeddings in the skip-gram model. Even use other techniques such as Global Vectors for Word Representation (GloVe) or the recent Transformers.

Besides, it will be important to test bigger datasets, since the dataset used during the project consists exclusively of concrete nouns and count with only 439 words. Therefore, different benchmarks of word similarity must also be applied, such as the widely used WordSim353 [43] or Rel-122 norms [44], since they contain many abstract words.

Also, everyday interest in GNNs increases, which generates more and more methods such as GCNII, SAGE, or GAT (to mention a few), so that, implementing these methods and comparing the results obtained would contribute to the investigation of the learning of semantic representations.

Chapter 7 **References**

[1] IRI, “About us.” https://www.iri.upc.edu/overview.

[2] H. Samani, *Cognitive Robotics*. Boca Raton: CRC Press, 2015.

[3] T. Tangiuchi *et al.*, “Survey on frontiers of language and robotics,” *Adv. Robot.*, vol. 33, pp. 700–730, 2019, doi: 10.1080/01691864.2019.1632223.

[4] C. Silberer, V. Ferrari, and M. Lapata, “Visually Grounded Meaning Representations,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, Nov. 2017.

[5] C. Matuszek, “Grounded Language Learning: Where Robotics and NLP Meet *,” *IJCAI-18*, 2018.

[6] R. J. Mooney, “Learning to Connect Language and Perception,” *Proc. Twenty-Third AAAI Conf. Artif. Intell.*, pp. 1598–1601, 2008.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” *NIPS*, Oct. 2013.

[8] Z. S. Harris, “Distributional Structure,” *Distrib. Struct. WORD*, vol. 10, no. 3, pp. 146–162, 1954, doi: 10.1080/00437956.1954.11659520.

[9] A. J. Anderson, D. Kiela, S. Clark, and M. Poesio, “Visually Grounded and Textual Semantic Models Differentially Decode Brain Activity Associated with Concrete and Abstract Nouns,” *TACL*, vol. 5, pp. 17–30, 2017.

[10] É. Zablocki, B. Piwowarski, L. Soulier, and P. Gallinari, “Learning Multi-Modal Word Representation Grounded in Visual Context,” *arXiv e-prints*, Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.03483.

[11] A. Lazaridou, N. T. Pham, and M. Baroni, “Combining Language and Vision with a Multimodal Skip-gram Model,” *arXiv Prepr. arXiv1501.02598v3*, Mar. 2015.

[12] M. Dimiccoli, “Learning visually grounded word meaning representations by hierarchical

multi-modal dynamic graph embedding," 2021.

[13] J. Brownlee, "What Is Natural Language Processing?," Sep. 2017. https://machinelearningmastery.com/natural-language-processing/ (accessed Jun. 06, 2021).

[14] NSS, "Understanding Word Embeddings: From Word2Vec to Count Vectors," Jun. 2017. https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/.

[15] H. Kung-Hsiang, "Word2Vec and FastText Word Embedding with Gensim," Feb. 2018. https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c.

[16] C. McCormick, "Word2Vec Tutorial - The Skip-Gram Model," 2016. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/.

[17] R. Anand, "An Illustrated Guide to Graph Neural Networks," Mar. 2020. https://medium.com/dair-ai/an-illustrated-guide-to-graph-neural-networks-d5564a551783.

[18] W. L. Hamilton, "Graph Representation Learning," *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 14, no. 3, pp. 1–159, 2020.

[19] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation Learning on Graphs: Methods and Applications," *IEEE Data Eng. Bull.*, Apr. 2017.

[20] T. Kipf, "Graph Convolutional Networks," Sep. 2016. https://tkipf.github.io/graph-convolutional-networks/ (accessed May 30, 2021).

[21] T. S. Jepsen, "How to do Deep Learning on Graphs with Graph Convolutional Networks," Sep. 2018. https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ICLR*, Feb. 2017.

[23] S. Khandani, "Engineering Design Process," *IISME*, vol. 6, Aug. 2005.

[24] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, 2nd ed. O'Reilly Media, Inc, 2019.

[25] R. Krishna *et al.*, "Visual Genome Connecting Language and Vision Using Crowdsourced Dense Image Annotations," *IJCV*, 2017, [Online]. Available: https://visualgenome.org.

[26] C. Fellbaum, "WordNet and Wordnets," in *Encyclopedia of Language and Linguistics*, A. Barber, Ed. Elsevier, 2005.

[27] L. Weng, F. Menczer, and Y.-Y. Ahn, "Virality Prediction and Community Structure in Social Networks," *Sci. Rep.*, vol. 3, 2013, doi: 10.1038/srep02522.

[28] M. Baroni, B. Murphy, E. Barbu, and M. Poesio, "Strudel: A Corpus-Based Semantic Model Based on Properties and Types," *Cogn. Sci.*, vol. 34, 2010, doi: 10.1111/j.1551-6709.2009.01068.x.

[29] M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta, "The WaCky Wide Web: A collection of very large linguistically processed web-crawled corpora," *Lang. Resour. Eval.*, vol. 43, pp. 209–226, 2009, doi: 10.1007/s10579-009-9081-4.

[30] K. McRae, G. Cree, M. Seidenberg, and C. Mcnorgan, "Semantic feature production norms for a large set of living and nonliving things," *Behav. Res. Methods*, vol. 37, pp. 547–559, 2005, doi: 10.3758/BF03192726.

[31] Google, "Colaboratory - Frequently Asked Questions." https://research.google.com/colaboratory/faq.html (accessed Jun. 02, 2021).

[32] R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010, pp. 45–50, [Online]. Available: http://is.muni.cz/publication/884893/en.

[33] G. Collell, T. Z. Zhang, and M.-F. Moens, "Imagined Visual Representations as Multimodal Embeddings," *AAAI*, vol. 31, 2017.

[34] M. Fey and J. E. Lenssen, “Fast Graph Representation Learning with PyTorch Geometric,” Mar. 2019, [Online]. Available: http://arxiv.org/abs/1903.02428.

[35] J. Brownlee, “How to Choose Loss Functions When Training Deep Learning Neural Networks,” Jan. 2019. https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/ (accessed Jun. 09, 2021).

[36] J. Brownlee, “How to Choose an Optimization Algorithm,” Dec. 2020. https://machinelearningmastery.com/tour-of-optimization-algorithms/ (accessed Jun. 09, 2021).

[37] J. Brownlee, “Understand the Impact of Learning Rate on Neural Network Performance,” Jan. 2019. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/ (accessed Jun. 09, 2021).

[38] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” 2014.

[39] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.01703.

[40] L. Jing and Y. Tian, “Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, Feb. 2019.

[41] Endesa, “Precio Kw/h Endesa.” https://www.endesa.com/es/blog/blog-de-endesa/luz/cuanto-cuesta-electricidad (accessed Jun. 07, 2021).

[42] IRI, “Activity Report,” 2020.

[43] L. Finkelstein *et al.*, “Placing Search in Context: The Concept Revisited,” *ACM Trans. Inf. Syst.*, vol. 20, no. 1, pp. 116–131, Jan. 2002.

[44] S. Szumlanski, F. Gomez, and V. K. Sims, “A New Set of Norms for Semantic Relatedness Measures,” *Proc. 51st Annu. Meet. Assoc. Comput. Linguist.*, pp. 890–895, Aug. 2013.

Chapter 8 **Annex**

To maintain order in this report, in the following link you will find all the programming codes used to carry out this project:

https://github.com/lainerjose/Final_Graduation_Project