

**Facultad de Tecnologías Interactivas**

**Desambiguación de Requisitos Funcionales mediante técnicas de Deep Learning**

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

**Autores:**

Daniel Ahmed Maya Valdés

Raiko Muñoz Hernández

**Tutores:**

Daniellys Cervante Lao

Yunieski Coca Bergolla

**Co-tutor:**

Samira de las Mercedes Enríquez González

**La Habana, febrero de 2023**

**“Año 65 de la Revolución”**

**DATOS DE CONTACTO**

**Nombre y apellidos del tutor**: Daniellys Cervante Lao

**Institución**: Universidad de las Ciencias Informáticas

**Título**: Master en Ciencias Informáticas

**Correo electrónico**: [dlao@uci.cu](mailto:dlao@uci.cu)

**Nombre y apellidos del tutor**: Yunieski Coca Bergolla

**Institución**: Universidad de las Ciencias Informáticas

**Título**: Doctor en Ciencias Informáticas

**Correo electrónico**: [ycoca@uci.cu](mailto:ycoca@uci.cu)

**Nombre y apellidos del tutor**: Samira de las Mercedes Enríquez González

**Institución**: Universidad de las Ciencias Informáticas

**Título**: Ingeniera en Ciencias Informáticas

**Correo electrónico**: [samiradlmeg@uci.cu](mailto:samiradlmeg@uci.cu)

**RESUMEN**

<Breve explicación de la investigación como un todo en un único párrafo entre 150 y 250 palabras, que refleje el por qué de la investigación, su objetivo y objeto de estudio, principales métodos o estrategia metodológica en su ejecución, principales hallazgos y resultados obtenidos y principales conclusiones. Debe lograrse al leer el resumen, obtener una panorámica general de la investigación tanto en su carácter teórico como metodológico y práctico>

PALABRAS CLAVE

Ambigüedad, Ingeniería de Requisitos, Procesamiento del Lenguaje Natural, Deep Learning.

***ABSTRACT***

<*Se escribe en idioma inglés la traducción del texto en el resumen. Se escribe en letra con formato “Italic” para diferenciarlo visualmente de su antecesor en idioma español*>

*KEYWORDS*

*<Se escriben en idioma inglés la traducción de las palabras clave en español. Igual se hace en letra con formato “Italic” para diferenciarlas visualmente de sus antecesoras en idioma español>*

**TABLA DE CONTENIDOS**

[INTRODUCCIÓN 1](#_Toc344763459)

[DESARROLLo 2](#_Toc278929498)

[I. Nombre del Epígrafe I 2](#_Toc390978404)

[II. Nombre del Epígrafe II 2](#_Toc134217981)

[III. Nombre del Epígrafe III 2](#_Toc1776030320)

[IV. Nombre del Epígrafe IV 2](#_Toc401846775)

[V. Nombre del Epígrafe V 3](#_Toc1052851871)

[VI. Nombre del Epígrafe VI 3](#_Toc302660974)

[VII. Nombre del Epígrafe VII 3](#_Toc502822672)

[VIII. Nombre del Epígrafe VIII 3](#_Toc2057043283)

[CONCLUSIONES 5](#_Toc1112242016)

[RECOMENDACIONES 6](#_Toc1317566085)

[REFERENCIAS BIBLIOGRÁFICAS 7](#_Toc1449060189)

[ANEXOS 10](#_Toc1480724508)

**ÍNDICE DE TABLAS**

<Listado de las tablas en orden de aparición en la memoria>

**ÍNDICE DE FIGURAS**

<Listado de las figuras en orden de aparición en la memoria>

**OPINIÓN DEL(OS) TUTOR(ES)**

<Contenido de la opinión de los tutores>

<Contiene la firma>

# INTRODUCCIÓN

La rápida evolución de las Tecnologías de la Información y las Comunicaciones (TIC) en beneficio de la actividad humana, adquiere en gran medida, el desarrollo de una cultura informática en la sociedad para la construcción de un nuevo entorno donde se facilite el quehacer diario del hombre. Cuba no está exenta de este acelerado ritmo si de tecnologías e informática se trata, por lo que también se encuentra en vías de desarrollo. La principal muestra de lo anterior se aprecia en la Universidad de las Ciencias Informáticas (UCI) la cual ha contribuido con la implementación de aplicaciones informáticas para lograr una mayor dinámica en el desarrollo de sus procesos sociales.

Durante el proceso de Desarrollo de Software, con motivo de entender y registrar las necesidades del cliente para traducirlo al lenguaje informático se hace necesario aplicar las buenas prácticas de la Ingeniería de Requisitos para obtener las funcionalidades necesarias de la solución a desarrollar. Es por ello que, durante las entrevistas entre el analista de software y el cliente, el primero toma notas sobre las peticiones que el segundo hace sobre solución deseada. La entrevista se realiza en lenguaje natural, lo que provoca que al analista en ocasiones no comprenda los requisitos funcionales que el cliente le comunica, la ambigüedad es otra de las causas por las que pudieran existir confusiones y/o malentendidos entre ambas partes lo que trae como consecuencia que el producto final no salga con la calidad requerida. La ambigüedad puede clasificarse en varios tipos, siendo estos: léxica, sintáctica, semántica y pragmática.

Virtualmente, cualquier lengua humana puede ser tratada por los ordenadores, lógicamente, limitaciones de interés económico o práctico hace que solo las lenguas más habladas o utilizadas en el mundo digital tengan aplicaciones en uso. Para lograr un tratamiento efectivo de estas lenguas, se hace necesario desarrollar el Procesamiento del Lenguaje Natural (PLN) que se refiere a la rama de la informática, y más específicamente a la rama de la [inteligencia artificial (IA](https://www.ibm.com/topics/artificial-intelligence)), que se ocupa de dar a las computadoras la capacidad de comprender textos y palabras habladas de la misma manera que los seres humanos (IBM, s. f.). Es muy común encontrar en cualquier idioma palabras con múltiples significados; por ejemplo, “lengua” puede significar la capacidad del habla de los humanos u órgano muscular muy movible que se encuentra fijado en la boca de los vertebrados.

El significado particular de una palabra viene determinado por el contexto que la rodea y en muchas ocasiones por la situación en que se emplea. Si por ejemplo, se tiene la oración: “Venía corriendo con la lengua afuera”, en este caso, se sabe exactamente el significado asociado a esta palabra. El procedimiento para decidir los significados de las palabras a partir del contexto que las rodea se conoce como “desambiguación” o “*Word Sense Disambiguation*” (WSD por sus siglas en inglés). En PLN las investigaciones en WSD han existido desde la aparición de esta área de investigación. En este campo el problema de la ambigüedad puede tratarse desde distintas perspectivas siendo las Redes Neuronales Transformers (RNT) las más utilizadas en el campo del PLN como solución a la ambigüedad del habla.

A partir de lo expuesto anteriormente y a través de las investigaciones realizadas fue posible determinar que Cuba hoy cuenta con una única herramienta que permite el entrenamiento, la prueba y validación de técnicas para eliminar la ambigüedad de requisitos de software, por lo que se evidencia una disminución de este problema. SIDARES inicialmente fue desarrollada para detectar ambigüedades léxica y sintáctica por lo que aún no tiene implementado entre sus funcionalidades la desambiguación de requisitos con ambigüedad semántica.

Como consecuencia de la situación planteada anteriormente se define como problema científico

**Situación problemática**

**Problema**

¿Cómo resolver la ambigüedad semántica en requisitos funcionales de software utilizando técnicas de Deep Learning?

**Objeto de estudio**

PLN para desambiguación de Requisitos Funcionales de Software

**Campo de acción**

PLN para desambiguación de Requisitos Funcionales de Software utilizando técnicas de Deep Learning

**Objetivo**

Desarrollar una herramienta para desambiguación de Requisitos Funcionales de Software utilizando técnicas de Deep Learning

**Tareas de investigación**

1. Elaborar el marco teórico de la investigación para establecer las principales tendencias de las herramientas para desambiguación de Requisitos Funcionales de Software utilizando técnicas de Deep Learning.

2. Realizar el análisis y la modelación de los requerimientos de la solución a desarrollar.

3. Asimilación de la metodología, plataforma, tecnologías, librerías, herramientas y pautas definidas por el equipo de investigación para el desarrollo de sus soluciones.

4. Definir la arquitectura a utilizar para dar solución a la problemática planteada.

5. Desarrollar una herramienta para desambiguación de Requisitos Funcionales de Software utilizando técnicas de Deep Learning.

6. Validar la solución propuesta mediante la utilización de métodos científicos de investigación.

**Métodos científicos**

**Teóricos**

**Analítico – Sintético**: posibilitará analizar por partes el objeto para facilitar su estudio mediante la determinación de sus componentes principales (PLN para desambiguación de Requisitos Funcionales de Software).

**Histórico – Lógico**: permitirá estudiar la evolución de los conceptos significativos relacionados con la investigación, así como trabajos similares con puntos coincidentes.

**Empírico**

Análisis documental: se utilizará en la revisión de la bibliografía consultada para extraer la información necesaria y obtener los referentes teóricos y conceptuales de la investigación siendo ACM, Springer, Google Schoolar e IEEE las principales bases de datos científicas utilizadas como fuente de investigación.

**Encuesta**: se empleará para conocer el nivel de satisfacción sobre la solución brindada.

**Preguntas científicas**: Al cabo del proceso de investigación a realizar se persigue dar respuesta a las siguientes preguntas de investigación:

¿En qué situación se encuentran las herramientas desarrolladas para el Procesamiento del Lenguaje Natural en Ingeniería de Requisitos?

¿Cuáles son los tipos de ambigüedad semántica que existen en los requisitos de software?

¿Qué técnicas han sido desarrolladas para detectar y corregir la ambigüedad semántica?

¿?

¿?

# Capítulo I – Fundamentación teórica

**Introducción al capítulo**

En este capítulo se realiza un estudio del estado del arte y se exponen conceptos y problemas relacionados con la ambigüedad de requisitos de software. Se hace referencia a los conceptos, técnicas, tareas, actividades y algoritmos del Procesamiento de Lenguaje Natural (PLN). Se describen además la metodología, herramientas y tecnologías utilizadas en el desarrollo de la solución. Todo lo abordado permitió conocer los puntos que pueden ser incluidos en la propuesta de solución que será abordada en el Capítulo 2. Finalmente se describen las características principales de la herramienta SIDARES y como contribuye a la solución de la problemática planteada.

## Ingeniería de Requisitos (IR)

La Ingeniería de Requisitos es tratada como una disciplina que tiene por propósito "desarrollar una especificación de requerimientos completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en dónde se describen las funciones que realizará el sistema".(Toro & Peláez, 2016). Por su parte Alan Davis1 define el concepto de requisito como “función o característica necesarias de un sistema que puede percibirse externamente”. El proceso de Ingeniería de Requisitos es una parte importante de la ingeniería de software y es crucial para el desarrollo de sistemas de software en el mundo real.(Rafiq et al., 2017). La práctica de levantamiento de requisitos depende en gran medida de algunos factores, incluyendo el tipo de Sistema, el tipo de software desarrollado y el grado de adaptabilidad(Rafiq et al., 2017). En el proceso de desarrollo de software de la IR consta de cuatro escenarios siendo estos el levantamiento o documentación, análisis, validación y especificación(Ashfaq et al., 2021). Durante la etapa de documentación todos los requisitos son redactados en lenguaje natural y por su naturaleza estos contienen ambigüedad debido a las múltiples interpretaciones que el analista de software puede tener sobre las peticiones del cliente, detectar dichas ambigüedades puede resultar difícil por las partes involucradas en el levantamiento de requisitos(Gervasi et al., 2019) la mayor parte

## Ambigüedad

Se entiende por ambigüedad a la posibilidad de que una palabra o frase tenga múltiples significados o interpretaciones. Una expresión en lenguaje natural es ambigua cuando puede ser interpretada en diferentes formas. En las entrevistas, las ambigüedades son asociadas con situaciones de malentendido, cuando la expresión de un cliente no es entendida o incorrectamente interpretada por el analista de requisitos.(Ferrari et al., 2016a)

Se dividen en cuatro clases

Tabla 1: Clases de la Ambigüedad. Fuente (Ferrari et al., 2016a)

|  |  |
| --- | --- |
| **Ambigüedad** | |
| **Descripción** | |
| **No claridad** | Incluye situaciones en las cuales los analistas de requisitos no pueden dar una interpretación o comprensión aceptable a una pieza de información expresada. |
| **Múltiples entendimientos** | Incluye situaciones en las cuales los analistas de requisitos son capaces de dar múltiples interpretaciones aceptables a la expresión del cliente, una correcta y otras incorrectas. |
| **Desambiguación incorrecta** | Incluye situaciones en las cuales los analistas de requisitos asignan una sola interpretación a la expresión del cliente, pero ésta es diferente del sentido que el cliente pretendía. |
| **Desambiguación correcta** | Incluye situaciones en las cuales los analistas de requisitos pueden asignar más de una interpretación al fragmento del discurso del cliente, pero la única interpretación que resulta aceptable para el analista es la que coincide con la pretendida por el cliente. |

Tabla 2: Clasificación de la Ambigüedad en la IR en Lenguaje Natural según varios autores

|  |  |  |  |
| --- | --- | --- | --- |
| (Ferrari et al., 2017)  (Porto Dapena, 2018)  (Gervasi et al., 2019)  (Ashfaq & Bajwa, 2021) | (Ferrari et al., 2016a) | (Gupta et al., 2019) | (Ashfaq et al., 2021) |
| Léxica  Sintáctica  Semántica  Pragmática | Léxica  Sintáctica  Semántica  Pragmática  Vaguedad  Generalidad | Léxica  Sintáctica  Semántica  Pragmática  Vaguedad  Generalidad  Genuina  Polisemia  Nociva  Anafórica  Incompletitud  Referencial  De alcance | Léxica  Polisemia  Discurso Referencial  Dominio  Sintáctica  Semántica  Pragmática  Vaguedad  Generalidad |

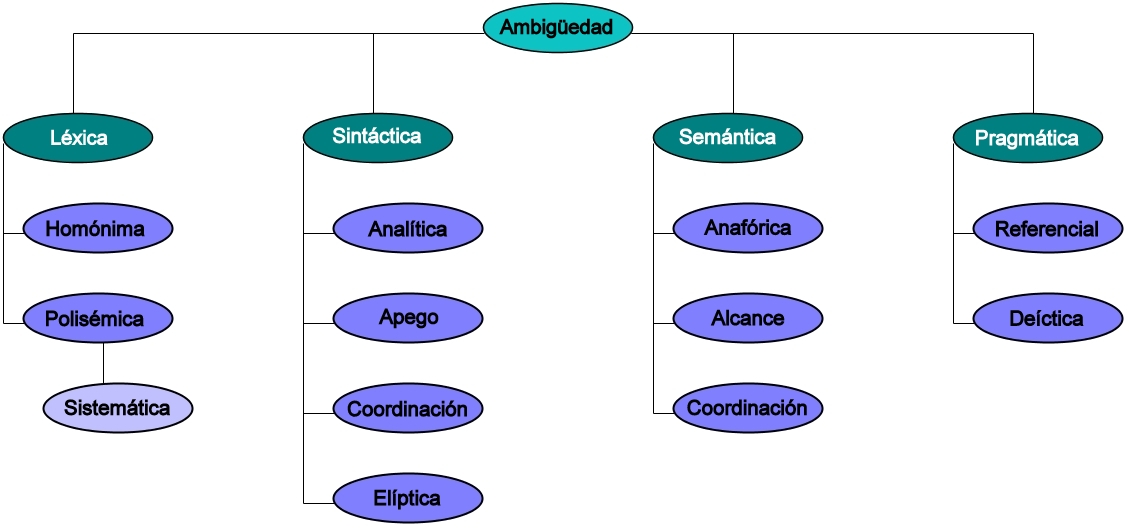


Figura 1 – Tipos de ambigüedad adaptado de (Ashfaq & Bajwa, 2021)

Como se menciona en (Ramírez Reyes & Enríquez González, 2022) donde se abordan las ambigüedades Léxica y Sintáctica, en este trabajo nos centraremos en el tratamiento de la ambigüedad Semántica. A continuación, se detallará sobre la misma y sus subtipos.

### **Ambigüedad Semántica**

La ambigüedad semántica es un tipo de ambigüedad que trata con el sentido, las posibles interpretaciones de texto. Esta Ocurre cuando una frase leva/posee múltiples formas de interpretarse dentro del contexto proporcionado, aun cuando no exista ambigüedad léxica, sintáctica o estructural (Gupta et al., 2019), (Ferrari et al., 2017), (Bhatia et al., 2016), (Bano, 2015). Es simple demostrar que una palabra tiene múltiples significados viendo el diccionario. No obstante, puede ser difícil probar que no tiene más de un significado, puesto que el lenguaje está en constante evolución y el sentido de las palabras también dependen del contexto en que se usen. (Hiltunen, 2020). Para una mejor comprensión tomemos el ejemplo de, “Todas las luces tienen un interruptor”; ¿un interruptor para muchas luces, o un interruptor para una luz? (Ferrari et al., 2016b). También puede observarse en palabras como banco (de peces, para sentarse, de datos), donde esta puede interpretarse como:

-Grupo masivo de peces

-Entidad donde se almacena dinero, se realizan peticiones de créditos.

-Almacenes digitales utilizados para guardar información.

La ambigüedad semántica como se menciona en la figura 1 se subdivide en 3 tipos,

#### **Ambigüedad Anafórica**

Una anáfora es un recurso de referencia textual en la que esta retoma la mención de una entidad que ha aparecido con antelación en el discurso (Bruscato & Baptista, 2021), ej. “Existe un prototipo y este será examinado para volver a usarse”, (“este” es una anáfora la cual se refiere a un prototipo). De forma similar una catáfora se refiere a una expresión que ocurrirá más adelante en el texto, ej. Si son válidos, los parámetros serán almacenados en el subsistema de procesamiento de datos, (“son” es una catáfora la cual se refiere a los parámetros). (Yang et al., 2011).

La ambigüedad anafórica ocurre cuando algún pronombre, ej. “él”, “este”, “ese”, “esto”, etc., se refiere a una parte previa del texto, siendo llamada antecedente la parte del texto referida (Ferrari et al., 2018). En la IR la anáfora normalmente se limita a la anáfora pronominal, ej. Cuando la anáfora es un pronombre. En la IR la detección de la ambigüedad anafórica es por tanto la tarea de identificar la aparición ambigua de pronombres. (Ezzini et al., 2022).

#### **Ambigüedad de Coordinación**

Esta ocurre cuando se usan conjunciones de coordinación (ej. y/o), estas llevan a múltiples interpretaciones de una oración, para esta ambigüedad se consideran dos tipos, el primero incluye oraciones donde más de una conjunción de coordinación son usadas en la misma oración. ej. “Hay un cambio de tarifa entre 1 y 100 kws y a partir de 300 kws debe haber un cambio de tarifa”. El segundo tipo incluye oraciones in las que una conjunción coordinadora es utilizada con un modificador, ej. “Enfoques y plataformas estructurados”, (“Estructurado” puede referirse solo al enfoque o a la plataforma también) (Ferrari et al., 2018)

#### **Ambigüedad de Alcance**

Una ambigüedad de alcance es una ambigüedad estructural que se deriva de interpretaciones alternativas del alcance de un operador. Ejemplos: «Javier hará dieta y ejercicio sólo si su médico lo aprueba» –alcance del operador sentencial: la aprobación del médico es condición necesaria para que haga dieta y también para que haga ejercicio (alcance largo de «sólo si») versus la aprobación es necesaria para el ejercicio pero no para la dieta (alcance largo de «y»); «Alberto tiene una teoría para todo» –alcance del cuantificador: una gran teoría omniexplicativa (alcance largo de «una teoría», que abarca «todo») versus cada cosa es explicada por alguna teoría («todo» tiene alcance largo). El alcance de un operador es la subfórmula entera más corta en la que aparece el operador. Así, en «(A B) ⊃ C», el alcance de «» es «(A B)». En los lenguajes naturales, el alcance de un operador es lo que le está C-subordinado (Y está C-subordinado a X en un diagrama de árbol cuando el nodo domina inmediatamente a X domina también a Y). Una ocurrencia de un operador tiene alcance largo con respecto al de otro operador siempre que el alcance del primero incluye propiamente al alcance del segundo. Ejemplos: en «¬ (A B)», «¬» tiene alcance largo con respecto a «»; en «(∃x) (∀y)Fxy», el cuantificador existencial tiene alcance largo con respecto al cuantificador universal (Audi, s. f.)

## Tipos de aprendizaje en Inteligencia Artificial

El *Machine Learning/*Aprendizaje Automático (ML/AA) es la técnica que los sistemas utilizan para mejorar aprendiendo de la experiencia adquirida de los métodos computacionales. (Zhou, 2021) En los sistemas computacionales, dicha experiencia se encuentra en forma de datos, siendo la principal tarea del ML el desarrollar algoritmos de aprendizaje para la construcción de modelos a partir de los datos. (Zhou, 2021) Esto significa que los programas de computadoras mejoran con experiencia con respecto a algunos tipos de tareas y medidas de actuación. Esto es logrado al aplicar algoritmos de que iterativamente aprenden de problemas específicos sobre unos datos de entrenamiento, permitiéndoles a la computadora encontrar señales ocultas y patrones complejos sin haber sido programados para esta función. (Janiesch et al., 2021)

Para algunos autores existen cuatro tipos de categorías para el ML:

**Supervisado**: Se utilizan datos etiquetados de entrada y de salida y comúnmente se aplican en tareas de clasificación de datos, predecir sentimientos en textos como los tweets y clasificar textos.

**No supervisado**: Analiza los conjuntos de datos no etiquetados sin la necesidad de la inferencia de los humanos, por ejemplo, el proceso de manejo de los datos, se usa en la extracción generativa de características, la identificación de tendencias significativas y estructuras, agrupando en resultados y propósitos de exploración. Comúnmente se aplican en tareas de Clusterización, estimación de destinos, aprendizaje de características, reducción de la dimensionalidad, detección de anomalías.

**Semi-supervisado**: es definido como un hibrido de lo mencionado anteriormente sobre los métodos supervisados y no supervisados, cómo este opera en la operación con datos etiquetados y no etiquetados. EL objetivo final de este método es el de proveer un mejor resultado para la predicción que el producido usando los datos etiquetados solamente para el modelo.

**Reforzado**: es un tipo de algoritmo de ML que permite a los softwares agentes y maquinas el evaluar automáticamente el mejor comportamiento en un contexto en particular o ambiente para mejorar su eficacia. (Sarker, 2021)

Según (Liu & Lang, 2019) existen dos tipos de Machine Learning:

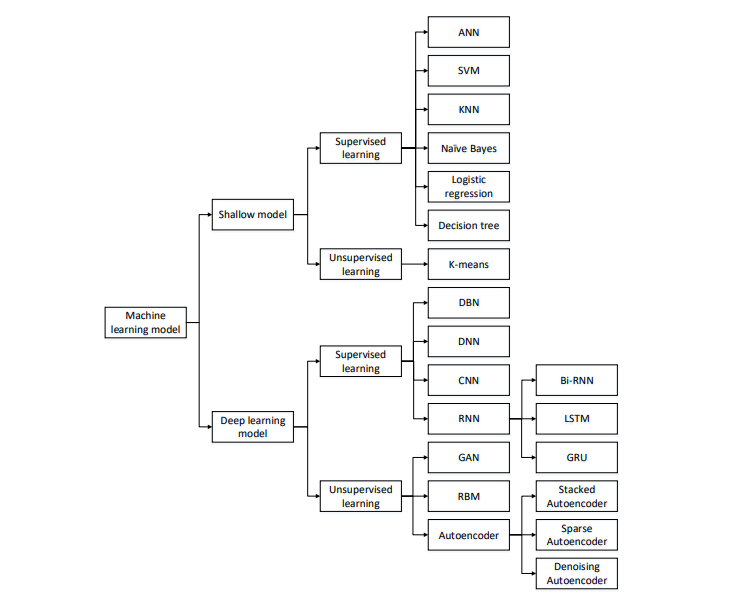


Figura 1 – Tipos de Machine Learning

**Deep Learnig**

Goldberg1 a branch of machine learning. a family of learning techniques that were historically inspired by the way computation works in the brain. DNNs stack up several hidden layers, with each layer acting as the input to the next layer. DL allows a computer to build complex concepts out of simpler concepts

DL emerged as especially promising for solving complex problems (e.g. machine translation or computer vision) and was primarily developed over the last ten years. Moreover, the combination of NLP and DL has been successfully embedded in user-centered artefacts, e.g., to enhance individual learning of students with adaptive writing support systems or adaptive tutoring through a smart personal assistant. A brief search reveals that there are over 40 different applications of NLP using a neural architecture

DL techniques

Multilayer Perceptron (MLP) is a feed-forward neural network with multiple (one or more) hidden layers between the input layer and output layer.

Autoencoder (AE) is an unsupervised model attempting to reconstruct its input data in the output layer.

Convolutional Neural Network (CNN) is a special kind of feed-forward neural network with convolution layers and pooling operations.

Recurrent Neural Network (RNN) use loops and memories in to remember former computations.

Deep Reinforcement Learning (DRL). Reinforcement learning operates on a trial-and-error paradigm. The whole framework mainly consists of the following components: agents, environments, states, actions, and rewards

**PLN**

Natural Language Processing (NLP) is a branch of artificial intelligence brimful of intricate, sophisticated, and challenging tasks related to the language, such as machine translation, question answering, summarization, and so on. NLP involves the design and implementation of models, systems, and algorithms to solve practical problems in understanding human languages.

NLP deals with applicative topics such as automatic extraction of relevant information (e.g., named entities and relations between them) from texts, translation of text between languages, summarization of documents, automatic answering of questions, classification and clustering of documents.

large neural networks have demonstrated to be superior to traditional ML algorithms. These models can often be trained with a single end-to-end architecture and they do not require traditional taskspecific feature engineering, making their adoption convenient. deep neural networks are able to handle a huge amount of training data.

**Tareas y aplicaciones**

Sequence classification. These NLP problems are full-fledged classification tasks. a function f c : X ! Yc able to assign a class to each sequence. Some relevant examples are (i) sentiment analysis, whose purpose is to classify a short text according to its polarity, (ii) document categorization, that finds the topic of a document (e.g., sport, finance...), and (iii) answer sentence selection, where the goal is to select the best sentence from a given paragraph/text to answer an input question.

Pairwise sequence classification. Pairwise sequence classification consists in comparing and classifying two different sequences according to their similarity, their semantics, and their meanings. One of the most popular applications is the Quora Question Pairs challenge,2 whose aim is to find duplicated questions from Quora.

Word labeling. In word labeling applications, a label is attached to each token wi 2 s. Examples of word labeling tasks are (i) Named Entity Recognition (NER), where relevant entities names, locations) are identified from the input sequence, (ii) classical question answering, where a probability distribution issued by an input paragraph is used to select a span containing the answer, or (iii) Part-of-Speech (PoS) tagging, that is the process of marking up a word in a text as corresponding to a particular part of speech (verb, noun, adjective, ...).

Sequence2sequence. In seq2seq problems, the input sequence is used to generate an output sequence this classification is not exhaustive but covers the most popular and relevant tasks. An example of a few NLP tasks applied to the input sentence.

Avances recientes

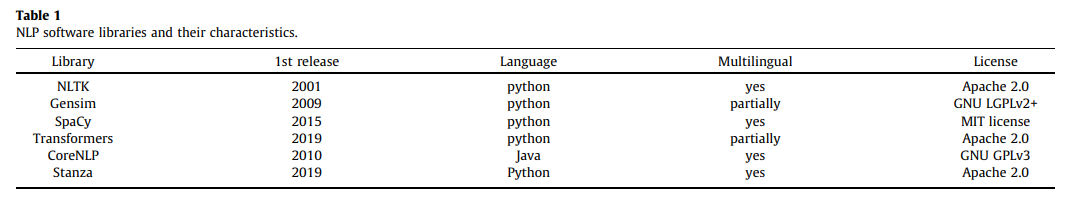
With the advent of word vectors, new methods to develop meaningful document and sentence level representations have been proposed. These methods can be categorized into two classes, i.e. unsupervised document embedding techniques, typically inspired by Word2vec, and supervised approaches. Unsupervised word/sentence vectors aim at extracting general representations that can be placed in various tasks. These methods can be trained on large scale unlabeled corpora through a language model objective function, which is a probability distribution over sequences of words

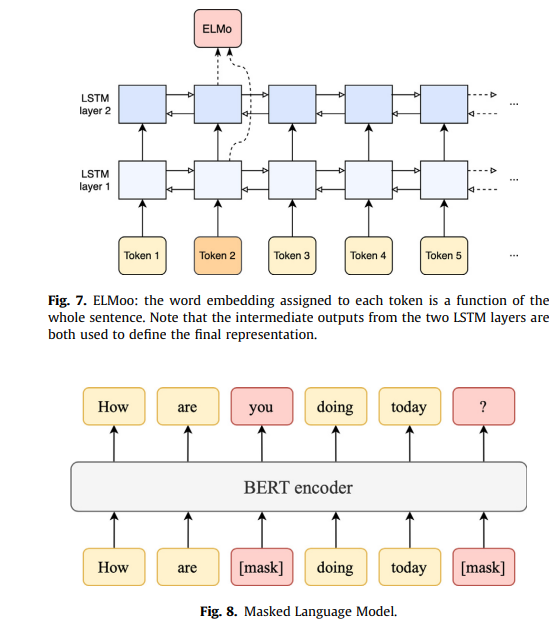
Modelos Transformers pre-entrenados

the last real boost in NLP after the advent of word vectors and unsupervised pre-training is the Transformer mode. The Transformer is the first architecture entirely based on attention to draw global dependencies between input and output, replacing the recurrent layers most commonly used in encoderdecoder architectures. The model showed a new state of the art in translation quality, while it can be trained significantly faster than architectures based on recurrent or convolutional layers. The evolution of language models pre-trained on large unlabeled corpora and the surprisingly empirical effectiveness of Transformer architectures are the two main pillars of modern NLP. One of the most popular pre-trained Transformer models is BERT [39] (Bidirectional Encoder Representations from Transformers). BERT is designed to pre-train deep bidirectional representations from unlabeled texts by jointly conditioning on both left and right contexts in all layers. The pre-training was driven by two language model objectives, i.e. Masked Language Model (MLM) and Next Sentence Prediction (NSP). In MLM, showed in Fig. 8, the network masks a small number of words of the input sequence and it tries to predict them in output, whereas in NSP the network tries to understand the relations between sentences by means of a binary loss. Specifically, the model has to select if two sentences are consecutive or not. After a pre-training phase, the model can be easily used in downstream tasks by fine-tuning the network on the target domain. BERT can be used in several different tasks, such as sequence classification, word- labeling, sequence2sequence, and so on. These methods rely on two main strengths, (i) the architecture strongly based on self-attention mechanisms that allow to read and to keep track of the whole input sequence, and (ii) the pre-training that allows the network to read and to (at least apparently) understand a text, its semantic and the meaning. Inspired by BERT, several pre-trained Transformers have been subsequently proposed, as is the case of RoBERTa [40], ALBERT [41], and DistilBERT [42]. These extensions of BERT were based on the same Transformer architecture with few small differences, without introducing additional features. For instance, RoBERTa criticized the NSP loss arguing that NSP is a critical task also for humans, and it does not improve the performance of the network. Other relevant methods based on the same concepts are GPT [43,44] (Generative Pre-Training), Transformer-XL [45], and its extension XLNet

**Tools**

NLTK [57]3 (Natural Language Toolkit) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. Gensim [58]4 is a Python library for topic modelling, document indexing and similarity retrieval with large corpora. Target audience is NLP and Information Retrieval (IR) communities. The library contains efficient implementations of popular algorithms, such as Latent Semantic Analysis (LSA/LSI/SVD), Latent Dirichlet Allocation (LDA), Random Projections (RP), Hierarchical Dirichlet Process (HDP). The library also includes several Word2vec pretrained models. SpaCy [59]5 is an industrial-strength library for performing NLP tasks in Python. The library is specifically designed to build complex industrial systems, and it interoperates seamlessly with TensorFlow, PyTorch, scikit-learn, Gensim and the rest of Python’s AI ecosystem. Spacy includes several functionalities, such as tokenization, NER, sentence segmentation, PoS tagging, and dependency parsing. The library also contains various pre-trained word vectors. Transformers [60]6 provides general-purpose architectures (BERT, GPT-2, RoBERTa, XLNet...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch. The library is curated by Huggingface, an NLP-focused startup with a large opensource community. The library exposes APIs to use many wellknown pre-trained transformer architectures described in the previous sections. CoreNLP [61]7 is a NLP library encompassing the main NLP modules and tasks, i.e. tokenization, PoS tagging, parsing, NER, for the Java programming language. The library provides support for 6 different languages. Stanza [62]8 is a collection of accurate and efficient tools for many human languages. Stanza includes a Python interface to the CoreNLP Java package. The toolkit is designed to be parallel among more than 70 languages, using the Universal Dependencies formalism. Finally, the package provides specialized tools and models for Biomedical and clinical tasks.





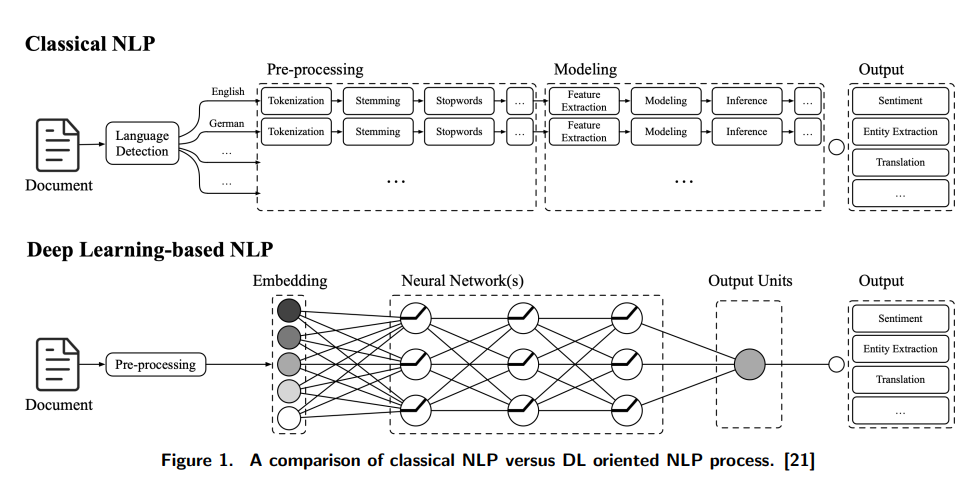
(Lauriola et al., 2022)

Over the last decade, the increase in available textual data and use of DNNs has propelled forward the field of NLP. What once used to be just a bag-of-words approach has become a major discipline in several academic fields such as computer science, psychology or linguistics.

Otter2 et al. broadly define NLP as: “. . . the engineering of computational models and processes to solve practical problems in understanding human languages

Using a classical approach, the data is first pre-processed which usually includes tokenization, pruning, stopword removal, filtering or stemming. In a DL workflow, the document is first pre-processed. The resulting artefact is then processed by a word embedding algorithm (e.g.Word2Vec) and passed over to one or more NN(s). A core element of NLP in DL are word embeddings. Goldberg describes word embeddings as follows: “A major component in neural networks for language is the use of an embedding layer, a mapping of discrete symbols to continuous vectors in a relatively low dimensional space” [16, p. 3]. This way of representing context as embeddings, allows language models to generalise unseen data very well [15]. When embedding words, they are transformed from isolated, distinct symbols into mathematical objects that can be operated on [16]. Word embeddings are generated by using already pre-trained neural nets/algorithms, where its input is a text corpus. An early and popular embedding algorithm is known as Word2Vec and was developed by Tomas Mikolov from Google [24, 16]. Today, there exist many variants of the Word2Vec algorithm such as fastText (https://fasttext.cc) or GloVe [25].

Otter et al. [20] underline the importance of gained understanding of real-world problems, and that a pure engineering perspective is meaningless in and of itself. Given the research interest in NLP, there are several different NLP applications. For instance, Ruder compiled a non-exhaustive list of over 40 NLP applications [10]. From this list as well as the deep learning NLP workflow, it can be derived, that there is a need for a holistic framework in order to categorise those applications and how they fit together with the configuration and workflow of DNNs.



(Landolt et al., 2021)

deep learning is a set of methods that permits computers to learn from data without human supervision and intervention.

Today, deep learning is prevalent in our everyday life in the form of Google’s search, Apple’s Siri, and Amazon’s and Netflix’s recommendation engines to name but a few examples. When we interact with our email systems, online chatbots, and voice or image recognition systems deployed at businesses ranging from healthcare to financial services, we see robust applications of deep learning in action.

Python has become the go-to language for many analytics applications, due to its simplicity, cleanliness of syntax, multiple data science libraries, and extensibility (specifically with C++). This simplicity and extensibility have led to most top deep learning frameworks to be built on Python or adopt Python interfaces that wrap high-performance C++ and GPU-optimized extensions.

Machine learning, a subcategory of AI, is composed of three areas: supervised learning, unsupervised learning, and reinforcement learning. Deep learning is a collection of learning algorithms that has been applied to each of these three áreas.

(Kamath et al., 2019)

**Redes Neuronales Artificales**

as the name already reveals, are computational networks that are able to solve complex, nonlinear mathematical problems, The field of ANN has been inspired by the ambition to model biological neural systems. NN are modelled as collections of layers of neurons that are connected in an acyclic graph

## Procesamiento del Lenguaje Natural para la IR

Las herramientas diseñadas para el propósito de levantamiento de requisitos se identifican como herramientas basadas en el lenguaje natural y están dirigidas al procesamiento de especificaciones textuales de requisitos y documentos (Simón et al., 2018).

## Tareas, herramientas y técnicas de PLN para IR

Extracción de texto plano, segmentación, etiquetado POS, análisis sintáctico superficial y de dependencias, reconocimiento de entidades, desambiguación y análisis semántico;

Analizadores sintácticos o parser, algoritmos de desambiguación, reconocedores de entidades;

Uso de patrones o reglas, de conocimiento externo y de algoritmos de aprendizaje automático;

Recursos externos

Ontologías, bases de datos léxicas como WordNet, taxonomías, bolsas de palabras, Wikipedia. (Simón et al., 2018)

La PLN es el procesamiento del lenguaje natural humano de forma automática o semiautomática. La PLN es esencialmente multidisciplinar y está relacionada con la lingüística. En el desarrollo de software, el ciclo de vida se puede aplicar a todas las fases. Hay artefactos textuales en las fases de análisis y diseño, como el documento de requisitos y la especificación de diseño de software. Los atributos de calidad como el rendimiento, las características, la confiabilidad, la estética y la percepción deben evaluarse en el software de PLN. Estas declaraciones de requisitos deben analizarse y luego debe utilizarse el procesamiento del lenguaje natural para ese propósito pues se proporcionan muchas herramientas que ayudan en el análisis lingüístico y ayudan en la asistencia automatizada. (Dawood, 2017) EL PLN es un enfoque computarizado que se ocupa de la interacción entre las computadoras y el lenguaje humano. El objetivo principal de la PLN es lograr un procesamiento del lenguaje similar al humano que permita a las computadoras comprender y generar el lenguaje utilizado por los humanos. (Al-Harbi et al., 2017)

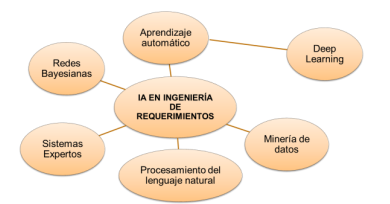
****

Figura 2 - Principales técnicas de Inteligencia Artificial aplicadas a la Ingeniería de Requisitos Tomada de (Seleey et al., 2022)

## Desambiguación del Sentido de la Palabra DSP

**Word Sense Disambiguation (WSD)** La tarea de determinar computacionalmente el sentido correcto de una palabra en contexto es un tema central de investigación en lingüística computacional y procesamiento del lenguaje natural. La razón de su importancia radica en la ambigüedad del lenguaje humano, que es tan generalizado que un gran número de palabras pueden interpretarse de múltiples maneras dependiendo del contexto en el que ocurren. (Navigli, 2012)

## Herramientas SIDARES

SIDARES constituye una herramienta de PLN desarrollada en la Universidad de las Ciencias Informáticas (UCI) con el objetivo de detectar la ambigüedad léxica y sintáctica presente en los requisitos de software, la cual utiliza el lenguaje de programación Python y la herramienta de procesamiento del lenguaje natural NLTK debido a que es de código libre. La herramienta se encarga de procesar y analizar exhaustivamente los requisitos de software y a su vez detectar los defectos en ellos. (Espinosa Mateo, 2023) Por la relevancia de esta herramienta para la validación de requisitos en el desarrollo de software en la UCI, es necesario el seguir mejorando dicha herramienta y para lograr este propósito se ha decidido el desarrollo de un módulo que se encargue de la desambiguación de

# Referencias Bibliográficas

Al-Harbi, O., Jusoh, S., & Norwawi, N. M. (2017). *Lexical Disambiguation in Natural Language Questions (NLQs)* (arXiv:1709.09250). arXiv. http://arxiv.org/abs/1709.09250

Ashfaq, F., & Bajwa, I. S. (2021). Natural language ambiguity resolution by intelligent semantic annotation of software requirements. *Automated Software Engineering*, *28*(2), 13. https://doi.org/10.1007/s10515-021-00291-0

Ashfaq, F., Bajwa, I. S., Kazmi, R., Khan, A., & Ilyas, M. (2021). An Intelligent Analytics Approach to Minimize Complexity in Ambiguous Software Requirements. *Scientific Programming*, *2021*, e6616564. https://doi.org/10.1155/2021/6616564

Audi, R. (s. f.). *Diccionario Akal de filosofia* (Second Edition). Cambridge University Press, Ediciones Akal, S. A., 2004. Recuperado 18 de febrero de 2024, de https://biblioteca.multiversidadreal.com/index.php?page=12&id=1789

Bano, M. (2015). Addressing the challenges of requirements ambiguity: A review of empirical literature. *2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE)*, 21-24. https://doi.org/10.1109/EmpiRE.2015.7431303

Bhatia, J., Breaux, T. D., Reidenberg, J. R., & Norton, T. B. (2016). A theory of vagueness and privacy risk perception. *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 26-35. https://ieeexplore.ieee.org/abstract/document/7765508/

Bruscato, A. M., & Baptista, J. (2021). THE RESOLUTION OF AMBIGUOUS ANAPHORA IN PORTUGUESE, ENGLISH AND SPANISH (A PILOT STUDY). *Alfa: Revista de Linguística (São José Do Rio Preto)*, *65*, e13626.

Dawood, O. S. (2017). From requirements engineering to uml using natural language processing–survey study. *European Journal of Industrial Engineering*, *2*(1), pp-44.

Espinosa Mateo, G. (2023). *Detección de ambigüedad léxica en requisitos de software en español mediante técnicas de Aprendizaje Automático*.

Ezzini, S., Abualhaija, S., Arora, C., & Sabetzadeh, M. (2022). Automated handling of anaphoric ambiguity in requirements: A multi-solution study. *Proceedings of the 44th International Conference on Software Engineering*, 187-199. https://doi.org/10.1145/3510003.3510157

Ferrari, A., Donati, B., & Gnesi, S. (2017). Detecting Domain-Specific Ambiguities: An NLP Approach Based on Wikipedia Crawling and Word Embeddings. *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 393-399. https://doi.org/10.1109/REW.2017.20

Ferrari, A., Gori, G., Rosadini, B., Trotta, I., Bacherini, S., Fantechi, A., & Gnesi, S. (2018). Detecting requirements defects with NLP patterns: An industrial experience in the railway domain. *Empirical Software Engineering*, *23*(6), 3684-3733. https://doi.org/10.1007/s10664-018-9596-7

Ferrari, A., Spoletini, P., & Gnesi, S. (2016a). Ambiguity and tacit knowledge in requirements elicitation interviews. *Requirements Engineering*, *21*(3), 333-355. https://doi.org/10.1007/s00766-016-0249-3

Ferrari, A., Spoletini, P., & Gnesi, S. (2016b). Ambiguity cues in requirements elicitation interviews. *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 56-65. https://ieeexplore.ieee.org/abstract/document/7765511/

Gervasi, V., Ferrari, A., Zowghi, D., & Spoletini, P. (2019). Ambiguity in Requirements Engineering: Towards a Unifying Framework. En M. H. ter Beek, A. Fantechi, & L. Semini (Eds.), *From Software Engineering to Formal Methods and Tools, and Back: Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday* (pp. 191-210). Springer International Publishing. https://doi.org/10.1007/978-3-030-30985-5\_12

Gupta, A. K., Deraman, A., & Siddiqui, S. T. (2019). A survey of software requirements specification ambiguity. *ARPN Journal of Engineering and Applied Sciences*, *14*(17), 3046-3061.

Hiltunen, L. (2020, septiembre 17). *Reducing structural ambiguity in natural language software requirements specifications* [Diplomityö]. Laturi.Oulu.Fi. https://oulurepo.oulu.fi/handle/10024/16840

IBM. (s. f.). *¿Qué es el procesamiento del lenguaje natural (PLN)? | IBM*. Recuperado 27 de noviembre de 2023, de https://www.ibm.com/mx-es/topics/natural-language-processing

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685-695. https://doi.org/10.1007/s12525-021-00475-2

Kamath, U., Liu, J., & Whitaker, J. (2019). *Deep Learning for NLP and Speech Recognition*. Springer International Publishing. https://doi.org/10.1007/978-3-030-14596-5

Landolt, S., Wambsganss, T., & Söllner, M. (2021). A taxonomy for deep learning in natural language processing. *HICSS*, 1-10. https://www.alexandria.unisg.ch/bitstreams/1c0aad38-2ed6-4963-97d5-ee3bb9e4cde5/download

Lauriola, I., Lavelli, A., & Aiolli, F. (2022). An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools. *Neurocomputing*, *470*, 443-456. https://doi.org/10.1016/j.neucom.2021.05.103

Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, *9*(20), 4396.

Navigli, R. (2012). A Quick Tour of Word Sense Disambiguation, Induction and Related Approaches. En M. Bieliková, G. Friedrich, G. Gottlob, S. Katzenbeisser, & G. Turán (Eds.), *SOFSEM 2012: Theory and Practice of Computer Science* (pp. 115-129). Springer. https://doi.org/10.1007/978-3-642-27660-6\_10

Porto Dapena, J. Á. (2018). *Sobre ambigüedad y vaguedad en los diccionarios*. https://doi.org/10.25145/j.refiull.2018.36.017

Rafiq, U., Bajwa, S. S., Wang, X., & Lunesu, I. (2017). Requirements Elicitation Techniques Applied in Software Startups. *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 141-144. https://doi.org/10.1109/SEAA.2017.73

Ramírez Reyes, J., & Enríquez González, S. de las M. (2022). *SIDARES: herramienta de procesamiento del lenguaje natural para la detección de ambigüedad léxica y sintáctica en requisitos de software*.

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, *2*(3), 160. https://doi.org/10.1007/s42979-021-00592-x

Seleey, D. A., Triana, C. E. P., & López, D. C. C. (2022). INGENIERÍA DE REQUERMIENTOS E INTELIGENCIA ARTIFICIAL: UNA REVISIÓN DE LA LITERATURA. *REVISTA COLOMBIANA DE TECNOLOGIAS DE AVANZADA (RCTA)*, *1*(39), Article 39. https://doi.org/10.24054/rcta.v1i39.1395

Simón, A., García Bermúdez, R., Del, M., Cruz, R., & Párraga Valle, J. E. (2018). *A Review of Automatic Requirements Extraction from Natural Language Text*. 47-58.

Toro, A., & Peláez, L. E. (2016). Ingeniería de Requisitos: De la especificación de requisitos de software al aseguramiento de la calidad. Cómo lo hacen las Mipymes desarrolladoras de software de la ciudad de Pereira. *Entre Ciencia e Ingeniería*, *10*(20), 117-123.

Yang, H., de Roeck, A., Gervasi, V., Willis, A., & Nuseibeh, B. (2011). Analysing anaphoric ambiguity in natural language requirements. *Requirements Engineering*, *16*(3), 163-189. https://doi.org/10.1007/s00766-011-0119-y

Zhou, Z.-H. (2021). *Machine Learning*. Springer Nature.