Diseño de un Velero
Análisis hidrodinámico

Carlos Yves Tizón Martínez
luis.mata@uam.es

Una tesis presentada para el Grado de Maestría en
Tu grado fav

Supervised by:

Supervisor 1
juan.delara@uam.es

Supervisor 2
esther.guerra@uam.es

Universidad {Autónoma, Complutense, Politécnica} de Madrid,
Madrid, Spain
23 de septiembre de 2023

I, Luis Mata Aguilar confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Inglés

Resumen

Español

Agradecimientos

This work has been funded by the Spanish Ministry of Science (RTI2018-095255-B-I00), and by the R&D programme of Madrid (P2018/TCS-4314).

Thanks to all the people who helped out:

Índice general

Índice de figuras

Índice de cuadros

Abreviaciones

ABC	Aqui Boca Caña

Capítulo 1

Introducción

This section is meant to introduce the background and objective of the research conducted for this thesis. The purpose of the first section of this chapter is to provide a concise summary of the study issue. Following the background are the objectives of the research. The contributions of this thesis are covered in the next section. Finally, here is a quick outline of the chapters of this thesis.

1.1 Contexto

Model Driven Engineering (MDE) speeds up the process of making high-quality software and keeps people from having to write the same solutions over and over [?]. It is a useful technique for software development since models facilitate the communication between requirements and implementation phases of the software lifecycle. Simultaneously conveying a more complete picture, models may also be used to reach agreement on a certain point of view or idea.

Model Driven Engineering and Machine Learning (ML-MDE) refers to the application of ML (Machine Learning) techniques to MDE issues. As stated previously, the model (or meta-model)[1] is the primary artifact in MDE, and can be represented as a graph. In order to apply ML techniques, it must be transformed into a lower level representation, such

[1]When we denote *(meta-)model* we are not referring to just meta-models but possibly to any model that can be used to describe a system. On the contrary, with *meta-model* we want to denote only meta-models. This is because we intend to expand this work to support virtually any kind of model, not just those that are used as proof of concept. The term is used broadly to maintain consistency with the rest of the literature.

as text.

1.2 Objetivos

We demonstrate the use of RoBERTa[?], a transformer-based architecture to support the construction of models by means of a recommender system able to predict a masked element of choice in a textual representation of the (meta-)model.

1.3 Contribuciones

This dissertation has produced the publication of a short work titled “Towards a Deep Learning Architecture for Software Models” [?], in which the beginning steps of this investigation are explored.

Capítulo 2

Marco Teórico

This section is intended to be a starting point for the reader, so that she can understand the fundamental ideas and the underlying concepts of the work presented in this thesis. It is separated in three sections, each one with a different purpose. First, we describe broadly MDE and its role in the development of software. Second, we describe the use of NLP and language models (LM) to extract usable text embeddings out of software models. Third, we describe the transformer architecture, a deep learning model state of the art in language modelling, able to extract usable embeddings from text corpora.

2.1 Model Driven Engineering

2.1.1 Language Models

Models that assign probabilities to sequences of words are called language models or LMs. Given a sequence of tokens, they are usually trained to compute the probability of the sequence given its history, i.e. the next word in a sequence of words [?]. They attempt to model understanding of a domain language in a corpus of text, and sometimes they can act as a knowledge base. LMs enable applications such as text summarization, text classification, and information retrieval among many others.

Language modeling is usually framed as unsupervised probability distribution over each token within a sequence, conditioned on the sequence of tokens observed so far. We denote the random variable representing the next token as x_t and the sequence of the tokens observed as x_c, i.e. language models compute $p(x_t|x_c)$.

For instance, in the unigram model, each word (token) w_i is treated independently of each other and the probability of a sentence $w_1, w_2, ..., w_n$, is computed as the product of the probabilities of each word (see Equation 2.1).

$$p(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} p(w_i) \tag{2.1}$$

Similarly the bigram model treats a sentence as pairs (w_1, w_2), (w_2, w_3), ..., (w_{n-1}, w_n). In such a way that the probability of a token is approximated as the probability of the token itself conditioned to its immediate predecessor. As in the unigram model, the probability of a sentence $w_1, w_2, ..., w_n$ is computed as the product of the probabilities of each token (see Equation 2.2).

$$p(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} p(w_i|w_{i-1}) \tag{2.2}$$

Generally we refer to the n-gram model, for a sequence of n words, where the probability of the sequence $p(w_1, w_2, ..., w_n)$ is approximated as the following.

$$\begin{aligned} p(w_1, w_2, ..., w_n) &= p(w_1)p(w_2|w_1)p(w_3|w_{1:2})...p(w_n|w_{1:n-1}) \\ &= \prod_{i=1}^{n} p(w_i|w_{1:i-1}) \end{aligned} \tag{2.3}$$

For example for the sentence “A bank of fishes approaches”:

$$\begin{aligned} p(\text{“A bank of fishes approaches.”}) = p(\text{“A”})\times \\ p(\text{“bank”}|\text{“A”}) \times p(\text{“of”}|\text{“A bank”})\times \\ p(\text{“fishes”}|\text{“A bank of”})\times \\ p(\text{“approaches”}|\text{“A bank of fishes”}) \end{aligned} \tag{2.4}$$

If we were to produce the simplest embedding (sparse vector) for a text application, that would be the one-hot vector. To represent every word as an $\mathbb{R}^{|V|\times 1}$ vector with all 0s and one 1 at the index of that word in the sorted language vocabulary. In this notation, $|V|$ is the size of our vocabulary. Word vectors in this type of encoding would appear as the following:

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \cdots w^{zero} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

$$O = E \times T \times Q$$

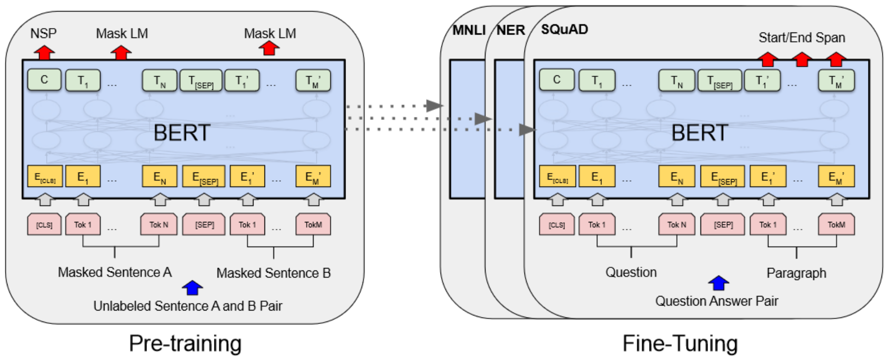

Figura 2.1: BERT fine tuning. The [CLS] token is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating question-s/answers), from [?]

Gastos compartidos						
Concepto	Luis	Total	Concepto	Mar	Total	
Algo que Luis pagó	327,65 €	Algo que Mar pagó	82,71 €			
lidl	3,88 €	finetwork	25,00 €			
verduras	8,55 €	iberdrola	27,91 €			
verduras	3,55 €	endesa	15,90 €			
uber	33,80 €	mercado san anton	13,90 €			
uber	27,70 €					
lidl	15,45 €					
frutas	17,85 €					
primavera	13,50 €					
asia	33,02 €					
cerves	30,00 €					
lild	10,33 €					
corta	5,90 €					
panko	32,75 €					
lidl	91,37 €					

Listing 2.1: Third query for the meta-model. We ask for associations.

```
#include <iostream>
using namespace std;

int main() {
  int n;

  cout << "Enter an integer: ";
  cin >> n;

  if ( n % 2 == 0)
    cout << n << " is even.";
  else
    cout << n << " is odd.";

  return 0;
}
```

Capítulo 3

Trabajo Relacionado

Capítulo 4

Propuesta

Capítulo 5

Experimentos

Capítulo 6

Conclusióon y Trabajo Futuro

We conclude the thesis by describing what we have achieved so far as well as prospective future research areas and advancements we examined throughout the execution of this thesis.

6.1 Conclusión

6.2 Trabajo Futuro

Apéndice A

Apéndice 1: Título

The following tables contain the keys and elements of the JSONs produced from the metamodels in our parsing process. The meta-model is represented as a three leveled dictionary with the following elements.

Key	**Value**	**Type**
’name’	Name of the metamodel	String
’annotations’	A list of annotations, this is usually empty, but it is intended to be filled with OCL restrictions and other relevant information.	List
’classes’	Class elements of the metamodel.	List of dictionaries
’enums’	A list of enumerates	List of dictionaries.

Cuadro A.1: The JSON object is a three level dictionary that contains the following keys (first level)

Apéndice B

Título apéndice 2

The technology used in this work has its foundations in the artificial neuron model. Right after the neuron model, the perceptron is explained along with its learning algorithm.