8[vo] CONGRESO NACIONAL

INGENIERÍA INFORMÁTICA / SISTEMAS DE INFORMACIÓN

VIRTUAL 2020
CONAIISI

50 UTN ANIVERSARIO FACULTAD REGIONAL SAN FRANCISCO

www.sanfrancisco.utn.edu.ar

05 | NOV.
06

RIISIC Red de Ingenierías en Informática / Sistemas de Información

confedi Consejo Federal de Decanos de Ingeniería República Argentina

150 ING la Argentina celebra su ingeniería 1870-2020

Octavo Congreso Nacional de

Ingeniería Informática/Sistemas de información

CONAIISI

5 y 6 de noviembre de 2020

Universidad Tecnológica Nacional,

Facultad Regional San Francisco

Memoria de Trabajos

Comité Organizador CONAIISI 2020

Comisión Ejecutiva RIISIC 2020

- **Coordinador:** Mag. Juan Carlos Calloni UTN FR San Francisco
- **Coordinador Suplente:** Lic. Augusto Nasrallah UTN FR Tucumán
- **Secretarios Titulares**
- Ing. Roberto Sotomayor - Universidad Fasta
- Ing. Gastón Martin - FICH Universidad Nacional del Litoral
- **Secretarios Suplentes**
- Calc. Liliana Estela Rathmann - Universidad Atlántida Argentina Mar de Ajo
- Ing. Marcela Viviana Fernández - UTN FR Mendoza
- Ing. Patricia Zachman - Universidad Nacional del Chaco Austral

Comisión Permanente RIISIC

- Dr. Ing. Marcelo Marciszack (UTN FR Córdoba)
- Ing. Luis Perna (UTN FR Delta)
- Ing. Andrés Bursztyn (UTN FR BA)
- Dr. Daniel Riesco (UNSL)
- Ing. Beatríz Gallo (UCASAL)
- Ing. Roberto Muñoz (UTN Regional Córdoba)
- Ing. Osvaldo Marcovecchio (Universidad Atlántida Argentina)
- Dr. Carlos Neil (UAI)

Autoridades Universitarias

<u>Universidad Tecnológica Nacional</u>

Rector: Ing. Héctor Eduardo Aiassa

Vicerector: Ing. Haroldo Tomás Avetta

<u>Facultad Regional San Francisco – UTN</u>

Decano: Ing. Esp. Alberto Toloza

Vice Decano y Director Dpto. Ingeniería en Sistemas de Información: Ing. Esp. Gabriel Cerutti

Comité Organizador Local

Coordinador Comité Local: Esp. Ing. Gabriel Cerutti

Coordinador Comité Académico: Mag. Juan Carlos Calloni

Coordinadora Comité Operativa: Ing Claudia Verino

Coordinador OCS (Open Conference Systems): Dr. Mario Groppo

Comité Operativo y Técnico

Emanuel Molina
Mauro Enrico
Sergio Felissia
Martín Ponce
Carlos Carrara
Andrés Giraudo
Tatiana Karlich
Ezequiel Fusero
Laura Vaudagna
Adrián Calorio
Trossero Mauricio

Asistentes técnicos en presentaciones orales

Apendino Carolina
Saldarini Javier
Carrizo Claudio
Chiapero Iván
Emanuel Molina
Franco Pomiro
Hernán Aquilano
Mónica Depetris
Florencia Francescato
Noelia Binotto
Verino Claudia
Antonela Calloni
Candela Ojeda

Comité Académico
CHAIRS

Aplicaciones Informáticas y de Sistemas de Información
Ing. Nelson Roberto Sotomayor - Universidad FASTA
Dra. Milagros Gutierrez - UTN F.R. Santa Fe
Dr. Elias Todorovich - Universidad Nacional del Centro de la Provincia de Buenos Aires

Aspectos legales y profesionales
Dra. Ana Garis - Universidad Nacional de San Luis
Ing. Gaston Martin - Universidad Nacional del Litoral
Ing. Andrés Bursztyn - UTN F.R. Buenos Aires

Bases de Datos
Dr. Aldo Vechietti - UTN F.R. Santa Fe
Magter. Vilallonga Gabriel - Universidad Nacional de San Luis
Magter. Pablo García - Universidad Nacional de La Pampa

Educación en Ingeniería
Dra. Herminia Beatriz Parra de Gallo - Universidad Católica de Salta
Dr. Carlos Gerardo Neil - Universidad Abierta Interamericana
Lic. Sandra Cirimelo - Universidad FASTA, Universidad Nacional de Mar del Plata

Ingeniería de Sistemas, Ingeniería de Software y Gestión de proyectos
Dr. Pablo Fillottrani - Universidad Nacional del Sur
Dr. Marcelo Martín Marciszack - UTN F.R. Córdoba
Ing. Felipe Evans - Universidad Nacional de Mar del Plata

Redes - Sistemas Operativos - Fundamentos Informáticos - Seguridad
Dr. Mario A. Groppo - UTN F.R. Córdoba
Dr. Germán Montejano - Universidad Nacional de San Luis
Dr. Cicerchia, Cesar Daniel - Universidad de la Defensa Nacional

Trabajos de cátedra
Ing. Patricia Raquel Cristaldo - UTN F.R. Concepción del Uruguay
Ing. Paula Andrea Zanetti - Universidad Marina Mercante
Esp. Roberto Miguel Muñoz - UTN F.R. Córdoba

Trabajos de Final de Cátedra
Ing. Patricia Paola Zachman - Universidad Nacional de Chaco Austral
Ing. Norberto Gaspar Cena - UTN F.R. Villa María
Mg. Marisa Perez - UTN F.R. San Francisco

Trabajos de Investigación de Estudiantes extra-cátedra
Calc. Liliana Rathmann - UAA
Lic. Augusto Nasrallah - UTN F.R. Tucumán
Ing. Guillermina Rosana Nievas - Universidad Católica de Salta

EVALUADORES

Aballay Patricia
Abratte Pablo
Acosta Nelson
Acuña Borobach César
Agüero Martín
Aguirre Jesus
Albanez Herrera Daniel
Alvez Carlos
Amar Eduardo
Ambort Daniel
Amor Matías Nicolás
Andriano Natalia
Angeleri Paula María
Antonini Sergo Andres
Apolloni Javier
Apolloni Ruben
Arce José Francisco
Arenas María Silvina
Armando Silvana
Arroyo Arzubi Alejandro Luis
Aspis Analía
Astudillo Hernán
Aubin Verónica Inés
Baigorria Lorena
Ballarin Virginia Laura
Ballejos Luciana
Barrera María Alejandra
Báscolo Alejandro
Battaglia Nicolas
Bavera Francisco
Becker Pablo
Belloni Edgardo Aníbal
Bengochea Ignacio
Bentin Marcela Paola
Berdun Luis
Bergamini María Lorena
Bernardis Hernán
Berón Mario
Bertossi Valeria Iliana
Bigatti Cristian German
Blanco Néstor
Blas María Julia
Bogado Verónica
Boggio María Alejandra
Bollati Verónica Andrea
Bracalenti Tomas
Braun Germán
Breslin Roberto
Bressano Mario Osvaldo
Bruno Oscar R.
Bruno Oscar R.
Budan Paola Daniela
Bursztyn Andrés
Cáceres Nélida Raquel
Caliusco María Laura
Calloni Juan Carlos
Canali Luis
Diván Mario José
Echazú Alejandro Luis
Encinas Diego
Eribe Roberto Angel
Escudero Federico Nicolás
Estayno Marcelo
Eterovic Jorge Esteban
Evans Felipe
Farfan José Humberto
Faure Omar Roberto
Favro Daniel Ignacio
Fennema Marta Cristina
Fernández David Alejandro
Fernández Jacqueline
Fernandez Marcela Viviana
Ferreyra Juan Pablo
Fillottrani Pablo
Flores Soledad Fabiana
Fraga Alvaro Luis
Frias Pablo Sebastian
Funes Ana
Gaetan Gabriela
Galoppo José Luis
Garay Laura Adriana
Garbarini Ramiro Walter
Garcia Berta
García Mario Alejandro
García Pablo
Gastañaga Iris
Genin Fernando
Gentiletti Gerardo Gabriel
Gerzel Stella Maris
Giandini Roxana Silvia
Gibellini Fabian Alejandro
Giordano Lerena Roberto
Golobisky María Fernanda
Gómez Sergio Alejandro
Gonnet Silvio
Gonzalez Amarillo Angela Maria
Groppo Mario A.
Gualpa Mariano Martín
Guatelli Renata Silvia
Guerrieri Ruben Alberto
Gutiérrez María Milagros
Guzmán Analía
Hadad Salomon Rosana
Hasbani Carlos Natalio
Herrera Daniel Rodolfo
Herrera Norma Edith
Igarza Santiago
Istvan Romina Mariel
Jewsbury Alejandra
Joselevich María
Juarez Gustavo Eduardo
Kabusch Andrés
Kabusch Andrés Ricardo
Kloster Miriam Cristina
Nievas Guillermina Rosana
Nigro Alberto María
Novara Pablo
Odetti María Alejandra
Oliva Juan Facundo
Olsina Luis
Ortiz Maria Cecilia
Paccor Gastón Pablo
Pacheco Federico
Padua Wilson
Páez Sergio Ramon
Panizzi Marisa Daniela
Papa María Fernanda
Parra De Gallo Herminia Beatriz
Pascal Andres Jorge
Paz Menvielle María Alejandra
Pedroni Juan Pablo
Peralta Mario Gabriel
Perez Marisa Norma
Perez Norma Beatriz
Perez Sofia Beatriz
Perna Luis Horacio
Pessolani Pablo
Pinto Noelia
Poliche Maria Valeria
Pollo-Cattaneo María Florencia
Poncio Silvia Victoria
Pons Claudia
Portillo Rosana Marcela
Procopio Demian
Prodolliet Jorge Adrián
Pytel Pablo
Quinteros Sergio
Quispe José Rolando
Rapallini Ulises M. A.
Rathmann Liliana Estela
Redolfi Javier
Repetti Nicolas
Retamar María Soledad
Ribotta Gabriela Del Fátima
Rico Carlos Alberto
Rico Mariela
Ridao Marcela Natividad
Riesco Daniel
Riva Fabiana María
Rojas Cristina
Roldán Luciana
Romandetta Aníbal Julio
Romero Daniel
Romero Gilda R.
Romero Lucila
Romero Soledad
Roqué Luis
Rosenzvaig Federico
Rottigni Gisela
Rottoli Giovanni Daian
Rubio Diego

Canavesio Maria Mercedes
Carabelli Luis Maria
Carballo Carlos
Carbonari Daniela
Cardenas Marina Elizabeth
Cardoso Alejandra Carolina
Carignano María Celeste
Carrizo Claudio
Casalini Clara
Casanova Carlos
Casanova Pietrobani Carlos Anto
Casanovas Eduardo
Casanovas Ines
Casatti Martín
Cassani Matías Alberto
Castellaro Marta
Castillo Angel Fabian
Castillo Julio
Castillo Peña Fabián
Castro Alicia
Castro Claudia Susana
Castro Lechtaler Antonio Ricardo
Castro Marcelo
Caymes Scutari Paola
Cena Norberto Gaspar
Cenci Karina
Césari Matilde Inés
Chiodi Gustavo Alberto
Cicerchia Cesar Daniel
Cipriano Marcelo
Cirimelo Sandra
Cocconi Diego Alejandro
Conde Sergio Daniel
Constable Leticia Edith
Constanzo Bruno
Corso Cynthia Lorena
Cortez Alberto
Crespo Abel
Cristaldo Patricia Raquel
Crotti Patricia Susana
Cuenca Pletsch Liliana Raquel
Cukierman Uriel
Curti Hugo Javier
Damiano Luis Esteban
Dania Claudia Estela
Daniele Marcela E.
Dasso Arístides
De Battista Anabella Cecilia
De Federico Sara Ester
De Gregorio Matias Sebastian
De San Pedro María Eugenia
Degiampietro Maria Gabriela
Del Fresno Mariana
Delgado Andrea Fabiana
Destefanis María Laura
Di Gianantonio María Alejandra
Di Iorio Ana Haydée Haydée
Kowalski Victor Andrés
La Red Martínez David Luis
Lacquaniti Edgardo Luis
Lado Silvana Inés
Lamperti Sabrina Bibiana
Lasso Marta
Leale Guillermo
Ledesma Ernesto Esteban
Lema María Gabriela
Leone Horacio
Lesser Nestor Eugenio
Liberatori Héctor Pedro
Liberatori Marcelo Sadí
Liporace Julio César
Lopez De Luise, María Daniela
Lopez Nocera Marcelo Ubaldo
Lucero Nancy Del Carmen
Lucero Ucasal Nancy
Ludueña Veronica
Lujan Vega Luis
Luz Clara Bibiana Beatriz
Mac William Maria Irene
Macaione Rosa Eufracia
Maldonado Calixto Alejandro
Mana Franco
Marciszack Marcelo Martín
Marsiglia Silvana Carolina
Martín María De Los Angeles
Martinez Duque Dennis
Martinez Liliana Inés
Martinez Maria Roxana
Martinez Roxana
Martínez Sergio Luis
Massa Stella Maris
Medel Ricardo
Medina Oscar Carlos
Meles Judith
Meloni Brenda Elizabeth
Mendez Garabetti Miguel
Meschino Gustavo Javier
Mina Hector Omar
Mira Natalia Carolina
Miranda Enrique Alfredo
Miranda Natalia Carolina
Molina Silvia
Monferrato Marcelo
Montejano Germán
Montezanti Diego Miguel
Moralejo Raúl Omar
Morcela Antonio
Moreno Juan Carlos
Moyano Claudia Patricia
Mugetti Mare José Oscar
Muñoz Roberto Miguel
Mussetta Sebastián Norberto
Nadal Jorgelina Cecilia
Nasrallah Augusto José
Nazar Patricia
Rubiolo Mariano
Ruhl Analia Lorena
Ruiz De Mendarozqueta Alvaro
Ruiz Diaz Soledad
Rustan Silvina
Saclier Lucas Javier
Saldarini Javier
Sánchez Alejandro
Sanchez Cecilia
Sánchez Ernesto
Sanchez Reynoso Maria Laura
Sánchez Rivero Víctor David
Sandobal Verón Valeria Celeste
Sardi Duilio Agustin
Sarli Juan Leonardo
Savi Cecilia Andrea
Schab Esteban Alejandro
Servetto Arturo Carlos
Simieli Mariana Paola
Simieli Paola
Sitnyk Martín Enrique
Solis Isidro
Soria Mercedes
Sosa Zitto Rossana Elizabeth
Soto Juan Manuel
Sotomayor Nelson Roberto
Stefanoni María Eugenia
Stegmayer Georgina
Storti Mario
Straccia Luciano Nicolas
Strub Ana María
Suarez Maria De Las Mercedes
Such Victor Eduardo
Talamé María Lorena
Testa Oscar Alfredo
Todorovich Elias
Tolaba Ana Carolina
Tolaba Miguel Nestor
Tomaselli Gabriela
Torres Chapeta Alexis Olvany
Torres Gladys Mabel Del Valle
Toselli Paula Andrea
Tosselli Laura Ramona
Trettel Marta Cecilia
Varela Laura Veronica
Vazquez Alejandro
Vazquez Juan Carlos
Vecchietti Aldo
Vegetti Marcela
Verino Claudia Mariela
Vitri Hernán
Vitturini María Mercedes
Yennerich Germán
Yuan Rebeca Judith
Zachman Patricia Paola
Zanetti Paula Andrea
Zohil Julio
Zuñiga Mariela

Octavo Congreso Nacional de Ingeniería Informática/Sistemas de información

CONAIISI

5 y 6 de noviembre de 2020

Universidad Tecnológica Nacional, Facultad Regional San Francisco

Memoria de Trabajos

Claudia Verino, Juan Carlos Calloni, Gabriel Cerutti, Alfonsina E. Andreatta
(Compiladores)

San Francisco, Córdoba - Argentina, Marzo de 2021

Octavo Congreso Nacional de Ingeniería Informática/Sistemas de Información: CONAIISI. 5 y 6 de noviembre de 2020 / Claudia Verino, Juan Carlos Calloni, Gabriel Cerutti, Alfonsina E. Andreatta; compilado por Claudia Verino , Juan Carlos Calloni, Gabriel Cerutti, Alfonsina E. Andreatta. - 1a ed. -
Ciudad Autónoma de Buenos Aires: Universidad Tecnológica Nacional. Facultad Regional San Francisco, 2021.
Libro digital, PDF

Archivo Digital: descarga
ISBN 978-950-42-0202-8

1. Sistemas de Información. 2. Ingeniería Informática. I. Verino, Caludia, Calloni, Juan Carlos, Cerutti, Gabriel, Andreatta, Alfonsina E. comp.
CDD 004.07

Octavo Congreso Nacional de Ingeniería Informática/Sistemas de información
CONAIISI
5 y 6 de noviembre de 2020
Universidad Tecnológica Nacional, Facultad Regional San Francisco

Memorias de trabajo

Diseño de Tapa: María Laura Vaudagna

ISBN 978-950-42-0202-8
9 789504 202028

Auspiciantes:

Identification of user stories in software issues records applying pre-trained natural language processing models.

Francisco Javier Peña, María Luciana Roldán, María Marcela Vegetti
Instituto de Desarrollo y Diseño
INGAR (CONICET / UTN)
{fpveitia, lroldan, mvegetti}@santafe-conicet.gov.ar

Abstract

In the last decades, agile development methods have been increasingly adopted by the software industry. User stories are one of the primary development artifacts for agile project teams. Issue Management Systems are widely used by software development teams to generate user stories, and organize them in meaningful fragments: epics, themes, and sprints. In addition, these tools enable generating any kind of issues, like bugs, change requests, tasks, etc. The responsibility for correctly categorizing an issue is in the hands of the team members, so it is a task prone to errors and frequently omitted due to lack of time or bad practices. Thus, a current problem is that many issues in projects remain uncategorized or mislabeled. Several studies have shown that it is common to find the uncategorized user stories of a software project in large volumes of issues records maintained by Issue Management Systems. In this work, we present two Neural Network models for text classification that were implemented for the identification of user stories in issue records.

1. Introduction

Issue tracking, often called bug tracking, is the process of keeping track of the open development issues in a software development project [1]. An Issue Management System (IMS), also called issue tracker system, is a computer application designed to help ensure software quality and support to programmers and other stakeholders in the issue tracking process. The term “issue” is attributed to the unit of work to make an improvement in a computer system. In addition, it describes most of the types of tasks that are needed to be tracked when developing a computer system [1].

Today most of software development organizations have adopted agile development methodologies like SCRUM, Kanban and XP. Most of these agile methodologies recommend capturing requirements through user stories [2], which are frequently managed in an IMS, such as Jira. These systems allow development teams to organize a collection of user stories in meaningful fragments: epics, themes and sprints.

By using an IMS it is possible to store and manage other issues besides user stories, such as bugs, change requests, enhancements, new requirements, tasks, etc. Although these systems offer to the user the possibility of explicitly categorizing the type of an issue, the decision to select the category of a new issue is up to the person who creates the issue, and this information is often omitted or incorrectly specified, making it difficult their later identification. This means that a great deal of user stories ends up buried in repositories with large volumes of data and various types of "issues".

The correct identification of user stories is of interest to software engineering for several reasons. For organizations that have multiple related projects, it is important to count with an integrated requirements base. Requirements engineering activities are no longer associated with an individual system development process and thus an individual project [3]. In contrast, it is viewed as an independent activity executed across multiple projects and product developments. Therefore, an approach to identifying in a repository the issues that constitute "user stories" is useful to retrieve them, regardless they were categorized or not as "user stories."

For members of software project teams that are IMS users, having a tool for the automatic categorization of incidents in user stories would save time and error occurrences, which improves the quality of the project documentation.

On the other hand, as the user stories are often poorly written in practice and exhibit inherent quality defects, a research trend is the application of computational linguistic techniques to these user stories to solve classic challenges in requirements engineering, such as the formulation of high quality requirements or the creation of better models of system functionalities [2]. However, findings from these studies are dependent on the quality of labels assigned to issue reports. Therefore, as a starting point to apply these approaches, it is necessary to have correctly identified user stories, which must be recovered from large volumes of data.

For all the above, it is necessary a proposal to effectively identify "user stories" immersed in large issue logs repositories. In order to reach such objective, in this work, we propose two different models of Neural Networks for text classification using the Python library, TensorFlow [4], and carried out a comparison between them. The first model is called ElMo [5] and the second one BERT [6], being both of them pre-trained models published by Google through the TensorFlow-Hub [7] platform.

This paper is organized as follows. Section 2 reports some related works. Section 3 presents some theoretical concepts of the methods used for a better understanding of this work. Then, in Section 4, some details about the datasets generated and used are discussed and the implemented models are presented. In Section 5, the main results obtained by testing the different models are presented and a comparison is offered that takes into account various aspects such as accuracy of the models, syntactic analysis and semantic analysis capability. Finally, in Section 6, conclusions are drawn.

2. Related Works

One of the features provided by most IMS is the possibility to define a set of labels/tags to classify the issues and, at least in theory, facilitate their management. Several authors have explored the use of labels to categorize issues in IMS. In [8] the authors analyzed a population of more than three million of GitHub projects and give some insights on how labels are used in them. Their results reveal that, even if the label mechanism is scarcely used, using labels favors the resolution of issues. They also conclude that not all projects use labels in the same way (e.g., for some labels are only a way to prioritize the project while others use them to signal their temporal evolution as they move along in the development workflow).

In a study conducted on closed issue reports of three open source software systems from Jira, it has been observed that the label given to the issue reports about bug or improvement is not correct [9]. The authors manually classified more than 7000 closed issue reports from five popular open source software systems to analyze the accuracy of already labeled reports. Their findings state that 33.8% of closed issue reports are misclassified. Moreover, manual analysis of issues is time consuming, tough and may be error prone.

Thung et al. manually classified a dataset and applied machine learning algorithms for bug classification [10]. In [11] an automated approach is proposed to label an issue either as bug or other request based on fuzzy set theory. The labeling of bug reports is done in three phases. First, text from the bug reports is preprocessed. Second, the Fuzzy technique is applied and third the labeling is done using scores obtained after fuzzification. In [12], the authors selected seven projects in GitHub and built classification models based on issue information, text description and comments, in order to improve the maintenance tasks for development teams. Text information was preprocessed with text data mining techniques and information retrieval. Then, they evaluated the performance of classifiers with several metrics. They conclude that very suitable classifiers may be obtained to label the issues or suggest the most suitable candidate labels.

The aforementioned contributions employed datasets obtained from repositories of IMS used just for software development, and not for project management. For that reason, the focus of these works has been on the correct classification or labeling of defects or bugs. However, our work employs datasets obtained from IMS repositories used to manage both project and software development; that is, they include issues related to requirements' definition, such as the "user stories".

In the last years, a research trend has emerged regarding the application of computational linguistic techniques to user stories to solve classic challenges in requirements engineering, such as the formulation of high quality requirements or the creation of better models of system functionalities [2]. A related line of research is the extraction of conceptual models from natural language (NL) requirements, which can help to identify dependencies, redundancies, and conflicts between requirements from lengthy textual specifications. To extract meaningful models from NL requirements, researchers have been proposing heuristic rules for the identification of entities and relationships whenever the text matches certain patterns of the given language (usually English). For example, in [13] is proposed an automated approach based on natural language processing that extracts conceptual models from user story requirements. In another work, [14] proposed an approach to generate i* models from user stories. In addition in [15] are made contributions towards mapping user stories and use case models. Also in [16], user stories are used to extract quality attributes for early architecture decision making. As these proposals require user stories as input, mislabeled user stories greatly impact the results of such studies. So, to envisage better results from these studies on user stories, it is required to correctly label issues either as ‘user stories or ‘non-user stories.

3. Background

This section introduces the concept of user stores and types of neural networks architectures in which the models developed in this work are based. First, it describes the Bidirectional Long Short-Term Memory Recurrent Neural Networks and then Natural Language

Processing with Neural Networks. Then, it also the two strategies for applying pre-training in linguistic models are introduced.

3.1. User stories

Outside the world of software, a user story could be referred as a customer's testimonial or narrative, however it has a whole different meaning for software professionals. In terms of software development, a user story is a short description of something or a piece of software it is supposed to do, told from the perspective of the person who desires the new feature. Although going back to its beginnings, user stories were proposed as unstructured text but with some size restrictions [17], nowadays it is followed a compact template for write them. This template captures who is it, what it expect from the system feature, and optionally why it is important [18]. Although many different templates exists, 70% of practitioners use the template "**As a** (type of **user**) , **I want** (**goal**), [**so that** (some **reason**)]" [2].

Example 1:
As a visitor,
I want to purchase an event ticket

Example 2:
As an event organizer,
I want to search for new events by favorited organizers,
So that I know of events first

Example 3:
As an event organizer,
I want to receive an email when a contact form is submitted,
So that I can respond to it

3.2. Bidirectional Long Short-Term Memory Recurrent Neural Networks

Recurrent Neural Network (RNN) architectures have recently become a typical and famous neural networks model because of its capabilities to process sequential inputs and learn its dependencies [19]. An RNN is a type of neural network where the connections between neurons form a directed graph making a temporal sequence trough time steps feeding each hidden state to the next time step as shown in **Fig. 1**. This allows the network to have a dynamic temporal behavior, unlike common networks, RNNs can use an internal state (memory state) to process sequences of inputs. However, they have problems with long-term dependencies due to gradient vanishing [19].

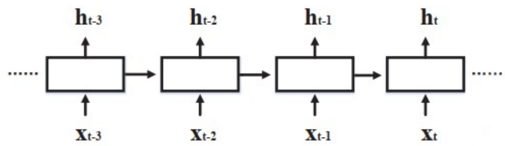

Fig. 1 Recurrent Neural Network, from [19]

Long short-term memory (LSTM) is a recurrent neural network architecture type that avoids the problem of gradient vanishing. LSTM is augmented by recurrent gates called "forgetting" gates, preventing the backward propagation error from vanishing or exploding. In this type of networks errors can go backward through a virtual unlimited number of layers unfolded in space. As is shown in **Fig. 2**, the internal memory cell C_t is controlled by a set of gate networks, including a forget gate network *f*, an input gate network i and an output gate network o. The forget gate network controls how much information of internal cell C_t should be passed into the next time step. The input gate network is used to scale the input block u to the internal cell. This means that LSTM can learn tasks that require memory of events that happened thousands of times in previous training steps, thus being able to handle long-term dependencies.

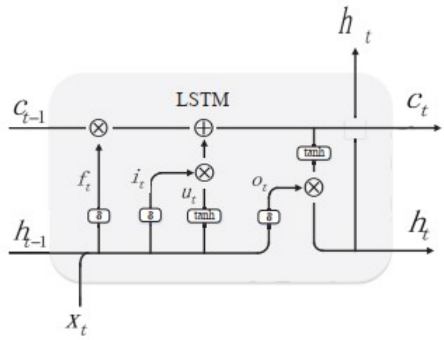

Fig. 2 Schematic of the LSTM, from [19]

Bi-directional Recurrent Neuronal Networks (BRNN) have a specific structure. The state neurons of a regular RNN are split in a part that is responsible for the positive time direction (forward states) and a part for the negative time direction (backward states), as shown in **Fig. 3**. These outputs of two types of states are not necessarily connected to inputs in the opposite states [11]. Using both time directions in same network, input information in the past (t-1 **in Fig. 3**) and the future (t+1 in **Fig. 3** of the currently evaluated time frame (t) can be used to minimize the objective function without the need for delays, unlike common RNN that require these "delays" to include future information.

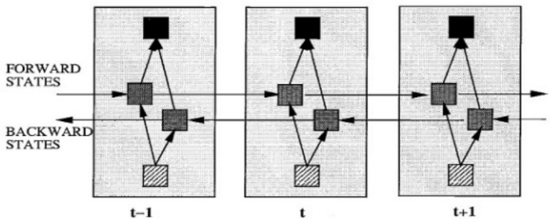

Fig. 3 . General structure of the bidirectional recurrent neural network (BRNN) shown unfolded in time for three-time steps, from [11]

3.3. Natural Language Processing with Neural Networks

Natural Language Processing (NPL) is a subfield of linguistic, computer, information engineering and artificial intelligence sciences dedicated to the interaction between computer equipment and human natural language, particularly how computer pro-grams process and analyze large amounts of information. The problems often ad-dressed with these techniques are speech recognition, understanding natural language such as sentiment analysis, text generation, automatic text summarization, and automatic entity recognition [12]. Although there exist several natural language processing techniques, in the last years, there has been a great boom in the use of Deep Learning models [12] because of their ability to capture the syntactic and semantic information of words in large unlabeled bodies of text. Word vectors (also called word embeddings) are a standard component found in most current NLP system architectures [12]. Word embeddings are vectors of real numbers that represent terms correlating relative similarities with semantic similarities [20], generally learned by neural networks.

3.4. Transfer learning

It is common that, different NLP tasks entail a great effort in terms of time and computing power consumption, so as an alternative to create a model from scratch or too general, the transfer learning technology has emerged [21]. Transfer Learning(TL) is a machine learning method with the perspective of providing a better and faster solution with less effort for collecting the needed training information and re-use it in another similar model [21]. In [21] it is defined as: “Given a Ds domain and a source Ts learning task, and a Dt domain and target Tt learning task, the TL aims to enhance learning of the target predictive function f(x) in Dt using the knowledge in Ds and Ts, where Ds ≠ Dt, or Ts ≠Tt”. Word embeddings are a good example of transfer learning since they are generally learned by neural networks in a domain for a learning task and these learned word embeddings can be applied in a different domain for other learning tasks, hence, those vectors of real numbers are transferred from a model to another model.

3.5. Embeddings from Language Models

Word representations, such as Word Embeddings, are a key component in many neural language models [22]. ELMo (Embeddings from Language Models) incorporates a form of deep word representation based on a feature-based approach, where each token is assigned a representation that is a function of the entire input sequence [22]. The vectors derived from a trained LSTM network with a pair of linguistic models are used in a long text corpus. These representations are a function of all the layers of a Bidirectional Linguistic Model (biLM) [22]. ELMo looks at the entire sentence before assigning each word in its embedding. It uses a bi-directional LSTM trained on a specific task, to be able to create contextual word embedding. The ELMo LSTM, after being trained on a massive dataset, can then be used as a component in other NLP models that are for language modelling. In [5], an implementation of a module with this architecture and an application trained in 1 billion words is presented. This module returns as output a set of fixed embeddings for each LSTM layer, the learned aggregation composed by 3 layers, and a mean-pooled vector representation of the input.

3.6. Attention models and Transformer

Attention mechanisms have become an integral part of sequential modeling in various tasks, allowing the modeling of dependencies regardless of the distance between input and output sequences, these are generally used with some type of RNN [23]. These models use the so-called attention functions, which are nothing more than a function that can be described as the mapping of a query and a set of identifier-value pairs to an output, where the query, the identifiers and the values are all vectors. The output is calculated as the weighted sum of the values where the weight of each value is calculated by a query compatibility function with the corresponding identifier[23]. In [23] the explanation of various types of attention functions, such as "Scaled Dot-Product Attention", "Multi-Head Attention" and "Self- Attention" can be seen.

As part of these models emerges the "Transformer" [23], a model completely based on the Self-Attention and Multi-Head Attention that for first time does not use alienated RNNs or convolutions, it follows an encoder-decoder architecture completely connected between its layers, the encoder maps an input sequence of symbol representations to a continuous representation, then the decoder generates an output of the symbols for each element at a time.

Its complete architecture and explanation can be seen in [23] and in [24] a notebook implementation can be obtained.

3.7. Bidirectional Encoder Representations from Transformers

There are two strategies for applying pre-training in linguistic models, the characteristics-based approach and the parameter adjustment approach [25]. Feature-based models such as ELMo [22] use architectures that include pre-trained representations as additional features. On the other hand, models that use parameter resetting introduce parameters to specific tasks trying to simplify and adjust all the pre-trained parameters. However, current techniques based on the parameter matching approach use unidirectional linguistic models [25].

BERT(Bidirectional Encoder Representations from Transformers) [25] alleviates this problem by making use of a masked linguistic model. The linguistic model masks some of the input tokens and aims to predict the original id of the vocabulary by linking the contexts from the right and left, hence it is bidirectional.

In [6], it can be found an implementation and examples of use of a module that fits this architecture trained in Wikipedia and BookCorpus. Assuming that the entries are pre-processed as required by this module implementation, it returns as output representations of each token in the input sequence and an entire grouped representation of the entry.

4. Proposed models for identifying user stories

This section describes the two recurrent neural network models developed to identify user stories in issue management systems records. Additionally, the details about the generation of the dataset employed to train and test both models are given.

4.1. Dataset

To train the models, data were taken from public sources that contain issues from real software development projects [26][27]. These sources contain positive examples of user stories (sentences in the format described previously) and negative examples (erroneous user stories or sentences with a similar syntaxis to user stories but with a different purpose).

To obtain a larger data set suitable for testing the models, an algorithm was implemented [28] for generating additional negative examples by splitting and mixing positive examples into random parts using the Tokenizer of TensorFlow. This implementation is available in [28]. In order to differentiate the examples to which each classification class belonged, a manual classification work was performed, which may have introduced to the model some human error index since there was no record of the previously classified data. The resulting dataset includes a total of 7997 positive and negative examples, of which 2618 are positive as those shown in Table 1, and the rest are negative as those shown in Table 2. Therefore, a binary classification problem is presented, where the issues classified as user stories belong to the positive (**1**) class and the rest to the negative (**0**) class. The whole obtained dataset can be found in [29].

Table 1. Sample of positive examples in the dataset

No.	Issue	Class
1	As a Carequality implementer, I want CONNECT to leverage the Carequality framework so I can exchange with other Carequality participants	1
2	As a CONNECT administrator, I want the ability to logout of the admin GUI application	1
3	As a CONNECT administrator, I want CONNECT to push audits and events via web services	1
4	As a CONNECT Adopter I need CONNECT to be database independent and support different databases such as Oracle.	1
5	As a CONNECT Adapter, I want to be able to respond to requests and receive responses to requests asynchronously in addition to synchronously	1
6	As a CONNECT adopter, I want a sourceless distribution option to enable to me to configure how connect is packaged and deployed	1

Table 2. Sample of negative examples in the dataset

No.	Issue	Class
1	Add enable/disable exchange refresh function to Exchange Manager GUI	0
2	Add details should anchor tag you back to the expanded section that you added from.	0
3	Add JUnit tests for mail classes for Mail package	0
4	i want to take a dataset offline so that i can perform a long running maintenance or migration procedure	0
5	as a url to social networks so that i can	0
6	as necessary including title date s language s and other facets	0

4.2. ELMo model

Using ELMo it allows to take advantage of its pre-trained embeddings for transfer and tune the previous learnt knowledge (see sections 2.4 and 2.5) to this model along with the aforenamed advantages of BRNN-LSTM architectures. The second implemented neural network for identification of user stories is a model based in ELMo embeddings. In it, a customized Keras layer was used for Tensor Flow. The implementation of the mentioned layer was taken from [30] and later integrated and adjusted to our model were issues are passed to the model as X_n inputs. Besides, a dropout layer was added to the model for preventing overfitting and a sigmoid activation function f. **Fig. 6** illustrates a general view of the whole sequential model using ELMo module.

For this implementation Tensor Flow 1.14 was used due to support and compatibility problems of the module with TensorFlow2.0 and the Tensor Flow-Hub library[7].

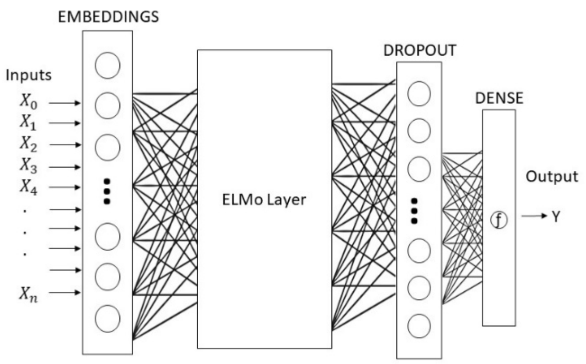

Fig. 1. The proposed model using ELMo module

4.3. BERT model

As an alternative to the ELMo model, it could be used BERT, a module that uses bidirectional encoders for transformers. The use of transformers could improve the semantic analysis of the issues, since these models are capable of learn were to focus the "attention" in sentences. Also, since BERT is bidirectional it can analyze the whole sentence regarding its length learning long term dependencies. For this model it was adapted another custom class as a Keras layer for integrate into this model taken from [31]. After the inputs are preprocessed obtaining the ids for the tokens and their respective masks, these are fed to the BERT layer, then a dropout layer it is used for preventing overfitting also and finally a sigmoid function as it is shown in **Fig. 7**.

For the implementation of this model, Tensor Flow 1.14 was used as well since there are some compatibility issues with TensorFlow2.0 and the Tensor Flow-Hub library [7].

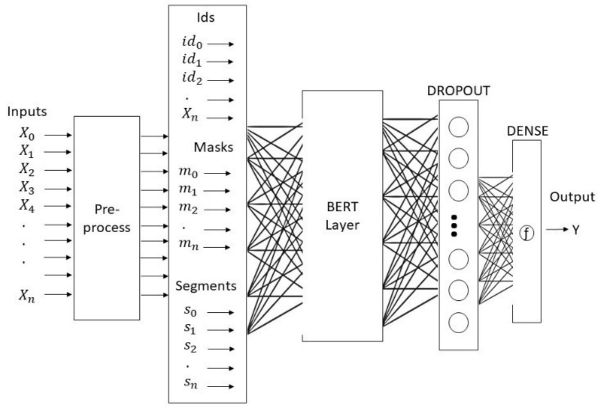

Fig. 2. The proposed model using BERT module

5. Results

All the models were trained with a random sample of the dataset. Then, in every training step iteration (**epoch**) are analyzed the values of **accuracy** and **loss** against values of accuracy (**val_ accuracy**) and loss (**val_loss**) during validation to check how well it is generalizing the model. Finally, the models are tested with new examples that were not analyzed during training or validation to obtain more realistic accuracies.

5.1. Results obtained using the ELMo module

The ELMo model is used as follows. The dataset it is randomly divided in 70% for training and 30% for testing, where the 25% of the training set is used for validation (see the full implementation in [32]). After 34 epochs of 23 seconds each one, an accuracy of 0.9607 in validation it is obtained. An analysis of the performance of this model shows a better performance than the previous one without a relevant overfitting (Fig. 3 and Fig. 4).

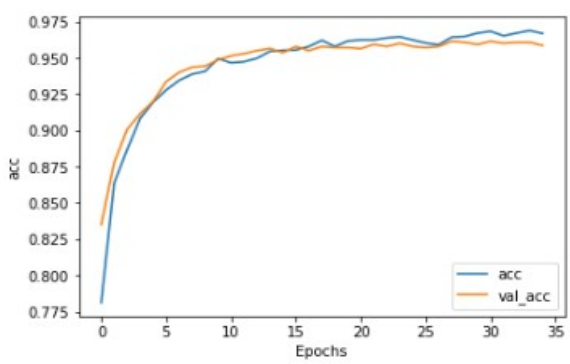

Fig. 3. Accuracy analysis for the model using ELMo

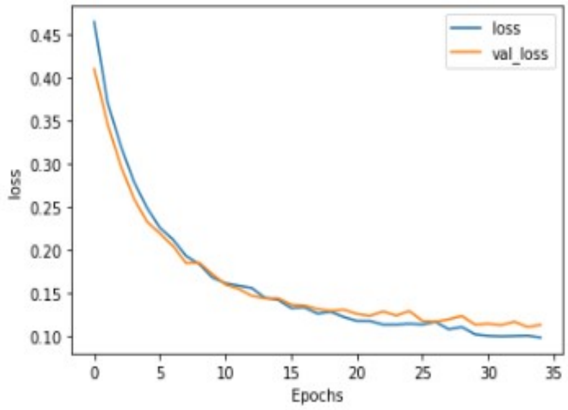

Fig. 4. Loss analysis for the model using ELMo

Testing the user stories examples in **Table 3**, which are out of the training and validation dataset, the results listed in **Table 3**, in the column titled "Probability of being a User Story using ELmo" were obtained. From these results, several observations can be made:

- The model has improved quite significantly identifying short user stories like the first example.
- Besides orthographic errors or unknown words the model continues generalizing correctly, as it is seen in the User Story 7 example.
- Regarding the User Story 5 example, despite there have been used similar words before for user stories, this sentence is not a positive example and therefore the model returns a low probability, which it is correct.

Table 3. Testing new examples in the ELMo and BERT models

No.	*Issue*	*Probability of being a User Story using ELmo*	*Probability of being a User Story using BERT*
1	As a developer, I want to implement tests	0.9155	0.9935
2	As a tester, I want to implement tests so i can assure the softwares quality	0.6258	0.9910
3	as an administrator i want a gui admin for configuration options	0.9973	0.8623
4	A tester want to implement tests so he can assure the software quality	0.0038	0.0012
5	**I want a developer as much as good tester so I have a good team**	**0.1686**	**0.0151**
7	As a IA tester, I want to wrtie with ortografics errors to test efficiency	0.8843	0.9195
8	An administrator will audit event via the system administration module	0.0044	0.0021
9	As a developer the default build should take less than 5 minuts	0.0008	0.0166

From the exposed results regarding the ELMo model, it can be concluded that a model with an accuracy of 0.96 accuracy is obtained. Furthermore, despite having an accuracy similar to the previous model, it can be said that it gives better results in terms of semantic and syntactic evaluation of the context of the cases. However, due to its higher complexity, it takes a little bit longer to train.

5.2. Results obtained using the BERT module

The BERT model is used as follows. The dataset it is randomly divided in 70% for training and 30% for testing, where the 25% of the training set is used for validation (see the full implementation in [33]) after preprocessing the entries. Then, after 7 epochs of 4 minutes each one, an accuracy of 0.9676 in validation it is obtained. An analysis of the performance of this model shows a better performance than the previous one without a relevant overfitting (Fig. 5 and Fig. 6).

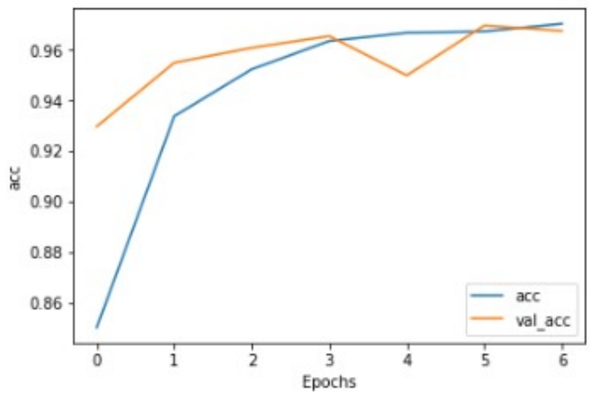

Fig. 5. Accuracy analysis for the model using BERT

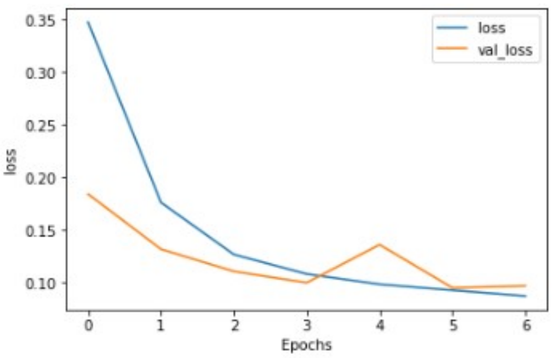

Fig. 6. Loss analysis for the model using BERT

Testing the user stories examples in **Table 3**, which are out of the training and validation dataset, the results listed in **Table 3**, in the column titled "Probability of being a User Story using BERT" were obtained. From these results, several observations can be made:

- The model has a slightly improve identifying short user stories like the first example.
- Besides orthographic errors or unknown words the model continues generalizing correctly, as it is seen in the User Story 7 example.
- Regarding the User Story 5 example, the model returns a lower probability making sure that this is not a positive example.

5.3. Comparison using ELMo vs BERT

After implementing the two different models and having evaluated their results, a comparison can be made, regarding the test accuracy (acc), the complexity, the training time (tr-effort), the syntactic analysis (parsing), and semantic analysis (semantic). **Table 4** shows the results of the comparison.

Table 4. Comparing ELMo and BERT models

Model	acc	com-plexity	tr-effort	parsing	semantic
ELMo	0.96	high	high	high	middle
BERT	0.97	high	high	high	high

As it can be observed in Table 4, the two tested models obtained almost the same accuracy, the BERT model has a slightly superior accuracy. Although BERT has a higher complexity if there are compared the times and number of training epochs of each model is noticed an improvement in the semantic interpretation, while the parsing analysis of issues is similar for both models as it is shown in Table 4.

6. Conclusions

In this work, two different neural network models were implemented for the identification of user stories in large volumes of data. From the results obtained using both models, we analyzed which is better for classification of Issues records. We found that the BERT model is the one that best fits the problem posed, managing to classify the Issues in user stories with an efficiency of approximately 97%. Besides, the BERT model is able to analyze the text both syntactically and semantically. This work could be the entry point to apply any other automatic process that applies other NLP techniques that aim to analyze user stories within Issues logging systems. A future work is to improve the employed dataset by increasing the number of cases and finding a better balance between positive and negative classes, and then trying to train the models again in order to improve the results.

References

[1] C. Henderson, *Building Scalable Web Sites*, no. May. O'Reilly Media, 2006.

[2] G. Lucassen, *Understanding User Stories*. 2017.

[3] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[4] TensorFlow, "TensorFlow." https://www.tensorflow.org/.

[5] Tensorflow Hub - Google, "Elmo-Tensorflow Hub," 2018. https://tfhub.dev/google/elmo/3.

[6] Tensorflow Hub - Google, "Bert_uncased - TensorFlow Hub," 2019. https://tfhub.dev/google/bert_uncased_L-12_H-768_A-12/1.

[7] T. Hub, "TensorFlow Hub." https://www.tensorflow.org/hub.

[8] J. Cabot, J. L. C. Izquierdo, V. Cosentino, and B. Rolandi, "Exploring the use of labels to categorize issues in Open-Source Software projects," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015 - Proceedings*, Apr. 2015, pp. 550–554, doi: 10.1109/SANER.2015.7081875.

[9] K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: How misclassification impacts bug prediction," in *Proceedings - International Conference on Software Engineering*, 2013, pp. 392–401, doi: 10.1109/ICSE.2013.6606585.

[10] F. Thung, D. Lo, and L. Jiang, "Automatic defect categorization," in *Proceedings - Working Conference on Reverse Engineering, WCRE*, 2012, pp. 205–214, doi: 10.1109/WCRE.2012.30.

[11] I. Chawla and S. K. Singh, "An automated approach for bug categorization using fuzzy logic," in *ACM International Conference Proceeding Series*, Feb. 2015, vol. 18-20-Febr, pp. 90–99, doi: 10.1145/2723742.2723751.

[12] J. M. Alonso-Abad, C. López-Nozal, J. M. Maudes-Raedo, and R. Marticorena-Sánchez, "Label prediction on issue tracking systems using text

mining," *Prog. Artif. Intell.*, vol. 8, no. 3, pp. 325–342, Sep. 2019, doi: 10.1007/s13748-019-00182-2.

[13] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Extracting conceptual models from user stories with Visual Narrator," *Requir. Eng.*, vol. 22, no. 3, pp. 339–358, 2017, doi: 10.1007/s00766-017-0270-1.

[14] R. Mesquita, A. Jaqueira, C. Agra, M. Lucena, and F. Alencar, "US2StarTool: Generating i∗ models from user stories," *CEUR Workshop Proc.*, vol. 1402, no. istar, pp. 102–108, 2015.

[15] P. S. Wautelet Y, Heng S, Hintea D, Kolp M, "Bridging User Story Sets with the Use Case Model," *Link S, Trujillo JC Proc. ER Work.*, pp. 127–138, 2016, doi: 10.1007/978-3-319-47717-6 11.

[16] F. Gilson, M. Galster, and F. Georis, "Extracting Quality Attributes from User Stories for Early Architecture Decision Making," *Proc. - 2019 IEEE Int. Conf. Softw. Archit. - Companion, ICSA-C 2019*, pp. 129–136, 2019, doi: 10.1109/ICSA-C.2019.00031.

[17] M. Cohn, *User Stories Applied: For Agile Software Development*, 1st ed., vol. 284. Boston: Pearson Education, Inc, 2004.

[18] Y. Wautelet, S. Heng, M. Kolp, and I. Mirbel, "Unifying and extending user story models," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8484 LNCS, pp. 211–225, 2014, doi: 10.1007/978-3-319-07881-6_15.

[19] W. Xia, W. Zhu, B. Liao, M. Chen, L. Cai, and L. Huang, "Novel architecture for long short-term memory used in question classification," *Neurocomputing*, vol. 299, pp. 20–31, 2018, doi: 10.1016/j.neucom.2018.03.020.

[20] M. Sahlgren, "A brief history of word embeddings (and some clarifications) | LinkedIn," 2015. https://www.linkedin.com/pulse/brief-history-word-embeddings-some-clarifications-magnus-sahlgren/ (accessed Oct. 29, 2019).

[21] Y. P. Lin and T. P. Jung, "Improving EEG-based emotion classification using conditional transfer learning," *Front. Hum. Neurosci.*, vol. 11, no. June, pp. 1–11, 2017, doi: 10.3389/fnhum.2017.00334.

[22] M. Peters *et al.*, "Deep Contextualized Word Representations," *Proc. NAACL. Assoc. Comput. Linguist. (ACL);*, pp. 2227–2237, 2018, doi: 10.18653/v1/n18-1202.

[23] A. Vaswani *et al.*, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.

[24] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush, "OpenNMT: Open-Source Toolkit for Neural Machine Translation," in *Proceedings of ACL 2017, System Demonstrations*, 2017, pp. 67–72, doi: 10.18653/v1/P17-4012.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," no. Mlm, 2018, [Online]. Available: http://arxiv.org/abs/1810.04805.

[26] CONNECT, "Navegador de incidencias - Issue Tracker CONNECT," 2019. https://connectopensource.atlassian.net/issues/?jql=order by created DESC&startIndex=50 (accessed Oct. 30, 2019).

[27] Mendeley, "Requirements data sets (user stories)," vol. 1, Jul. 2018, doi: 10.17632/7ZBK8ZSD8Y.1.

[28] F. J. Peña Veitía, "fjpena35226/augmentingdataset_userstories." https://github.com/fjpena35226/augmentingdataset_userstories (accessed Aug. 04, 2020).

[29] F. J. Peña Veitía, "Identifying User Stories in Issues records," vol. 1. Mendeley, Apr. 14, 2020, doi: 10.17632/BW9MD35C29.1.

[30] J. Zweig, "keras-elmo/Elmo Keras.ipynb at master · strongio/keras-elmo," 2018. https://github.com/strongio/keras-elmo/blob/master/Elmo Keras.ipynb (accessed Oct. 31, 2019).

[31] J. Zweig, "keras-bert/keras-bert.ipynb at master · strongio/keras-bert," 2019. https://github.com/strongio/keras-bert/blob/master/keras-bert.ipynb (accessed Oct. 31, 2019).

[32] F. J. Peña Veitía, "fjpena35226/rnn_ElMo_userstories_recognition," 2019. https://github.com/fjpena35226/rnn_ElMo_userstories_recognition.

[33] F. J. Peña Veitía, "fjpena35226/bert_userstories_recognition." https://github.com/fjpena35226/bert_userstories_recognition (accessed Mar. 24, 2020).