Fine-Tuning Semantic Information for Optimized Classification of the Internet of Things Patterns Using Neural Word Embeddings

Vusi Sithole and Linda Marshall

***Abstract*—Word embeddings is a natural language processing modelling technique used to map semantically related words and phrases in proximity vectors. Such embeddings generally reflect semantic similarities between words taken from natural contexts in large corpora. Nonetheless, most natural contexts tend to also have numerous words which do not bear any particular close relationship with regard to their meaning. This results in a lot of noisy data, which also makes the training of word embedding models much more expensive. In this paper, we show that fine-tuning semantic information provide additional benefits for training optimized neural word embeddings. In particular, we use explicit semantic extractions of the Internet of Things patterns attributes as our input data into the model. We propose extracting specific sentences from a large number of the IoT-related documents. These sentences describe the attributes for different IoT patterns. To make our corpora semantically rich, we further extract synonymous words from a thesaurus for some individual words taken from the extracted sentences. This also makes the context of the data more natural. We then embed several IoT pattern names in vector spaces and surround each pattern name with core word units taken from its attributes. In this way, each IoT pattern is classified in close vector spaces with words that represent its core attributes. Furthermore, the IoT patterns belonging in the same family are also classified in close vector spaces based on their attributes. The word vectors obtained from such strict supervised training show improved results on intelligent classification tasks, suggesting that they can be useful in machine learning efforts for building applications used in the categorization of items into both distinct and indistinct classes.**

***Index Terms*—Internet of things, word embeddings, classification, neural networks, patterns.**

I. Introduction

In recent years, there has been a growing interest in text processing models, particularly, those using artificial neural networks. One of the most popular neural probabilistic language models is *word embeddings* which embed words in vector spaces based on contexts. As a result, words taken from similar contexts end up having numerical vectors which are in close proximity [1]. More often than not, words trained using word embeddings techniques such as *word2Vec* and *Glove* tend to have a semantic relationship [2]. In theory, word vectors with such semantic relationships can be leveraged to enhance many natural language processing tasks such as information retrieval, machine translation and developing question answering applications, etc [3]. However, at the moment, there is a limited number of studies published in the literature that document the usefulness of word embeddings to solve real-life problems. In this paper, the aim is to show the practicality of using *word embeddings* to solve a real-life problem – to classify the Internet of Things Patterns.

A known challenge in word embeddings is that some contexts in natural language are ineffectual [4]. In other words, some contexts play a very important role in meaning acquisition, while other contexts are less helpful in establishing semantic and syntactic relationships between neighboring words. In addition, even for those contexts in which semantic relationships between words is largely noticeable, some associations on a word-level segmentation still do not bear any semantic relationship with other words in the same context. For instance, consider this statement taken from Wikipedia:

"The Seventh-day Adventist Church is the largest of several Adventist groups which arose from the Millerite movement of the 1840s in upstate New York, a phase of the Second Great Awakening." (Wikipedia [5])

In the above expression, words such as *`Adventist'*, *`Church'*, *`Millerite'*, *`1840s'*, *`Seventh-day'* and *`Movement'* share certain commonalities with regard to the historical background of adventism. Nonetheless, some of the words in the expression cannot be semantically associated with these words, e.g. *`arose'*, *`phase'*, *`largest'*, *`second'*, etc. In this paper, we propose injecting fine-tuned semantic information into the word embedding model. That is, instead of using semantic information taken from general contexts, we extract certain information that have stronger semantic connections. In particular, we investigate using extracted IoT pattern attributes as our context information and then train the word embedding model using the *major parts of speech*[1] extracted further from those attributes.

II. Problem Statement

Training a large corpus of text based on generic language context is a very expensive and time consuming task. In computational linguistics and machine learning, this is known as *generic context embedding*. Generic context embedding is particularly expensive because it is based on *sentential contexts* which are composed of plain and unstandardized word units. In this paper, sentential contexts refers to the allocation of vector representations to each word

Manuscript received July 12, 2019; revised May 2, 2020.

The authors are with the Department of Computer Science, University of Pretoria, South Africa, Pretoria, South Africa (e-mail: u04409477@tuks.co.za, Lmarshall@cs.up.ac.za).

[1] Vector representations of words are only assigned to Nouns, Adjectives, Adverbs and Verbs.

in a sentence. Thus, the number of neighboring words taken into consideration around the target word becomes the window size. Due to the complexity of language, words in sentential contexts are sometimes remotely removed from each other both syntactically and semantically. For this reason, generic word embeddings are known to produce rather poor word vector representations. We also note that such generic word embedddings cause a problem of word sense disambiguation. To counteract these challenges associated with using large plain text corpora taken from generic contexs, we propose using specific extracted information which is of a limited size. While we are are fully aware that the main purpose of building a word embedding model is to use it on unrestricted and unsupervised data, particularly on a large corpus, we argue that using a small corpus consisting of fine-tuned information as a baseline for classifying objects can facilitate better and improved unsupervised embedding.

III. Related Work

Human vocabulary always comes in the form of free text. However, in order to make a machine learning model understand and process the natural language, we need to transform the free-text words into numeric values [6]. One of the simplest transformation techniques to do this is known as *one-hot encoding*[2].

Word embedding is a dense representation of words in the form of numeric vectors based on one-hot encoding. Word embeddings are a very popular and active research area in deep learning and Natural Language Processing. However, Word embeddings are commonly considered a to be a `shallow' technique of deep learning. Word embeddings, such as those created by the word2vec family of algorithms identified in Mikolov *et al.* are the current state-of-the-art for modeling lexical semantics in Computational Linguistics. Historically, *word embeddings* can be traced back to the works of John Firth, Zellig Harris, and Ludwig Wittgenstein in the early 1950's. In the earlier days, the quantification of semantic relations between objects was manually done using handcrafted features. An example of this is found in the work of Osgood carried out in 1964 on semantic differentials. In *Computational Linguistics*, Word embeddings are related to other distributional semantic models or distributional representations such as *Latent Semantic Analysis* and *Explicit Semantic Analysis, Principal Component Analysis, Topic Models*, *Compound Term Processing,* etc. Generally, word embeddings refer to numerical representations of texts. More formally, word embeddings are the numerical representations of words usually in a shape of a vector in distributed representation $\boldsymbol{R^d}$. In practice, word embeddings are unsupervisedly learned word representation vectors whose relative similarities correlate with semantic similarity.

The recent results by Mikolov *et al.* who used neural-network based word embeddings constructed in a non-supervised fashion have contributed significantly to natural language processing and the word embedding algorithm. In this experiments, neural networks feed on huge amounts of text and produce real-valued word vectors. Thus, the input of the algorithm is a large corpus of text, which are converted into random vector values, and then summarised in a $\mathbf{n} \times \mathbf{n}$ matrix $\boldsymbol{M}$ that encodes the relationships between $\boldsymbol{n}$ unique words. Generally, $\boldsymbol{M}$ ($\boldsymbol{i, j}$) records semantical relationships between the target word and the context word, such as the occurence in the same context window between word$_{\boldsymbol{i}}$ and word$_{\boldsymbol{j}}$. Subsequently, $\boldsymbol{M}$ is factorized and the coordinates (i.e. N-dimensional vector space) in the $\boldsymbol{d \ll n}$ most significant components define the vector representation of words.

There are some recent interesting works in word embeddings that are moving beyond representations for words. These are variations of the traditional word embeddings models discussed above. Some of these works extend word embeddings to cover phrases and sentences using different statistical models. Examples of these include Doc2Vec [7] and CompValEvac [8]. A somewhat similar model to what we propose in this paper is discussed in Chen and de Melo in which the authors extract information in the form of definitions, synonyms and lists. However, the model presented in this paper is significantly different since it is based on document analysis.

The shortcoming of word embeddings, however, is that words with opposite meanings (i.e. antonyms) also tend to appear in similar contexts, while these words, by definition, have opposite meanings. Therefore, word embeddings trained only with neighboring contexts tend to place words improperly in vector spaces. For instance, even though antonyms (e.g. true and false) are semantically opposite in meaning, their contexts are similar in most cases, which makes them share close vectors in vector spaces. This is evident in Mikolov *et al.* where it was found that the cosine similarity score of the (long, short) and (accept, reject) pairs is 0.71 and 0.73 respectively. Schwartz *et al.* [9] uses a symmetric based approach to resolve the antonyms problem in word embeddings. Symmetric patterns are a special type of patterns that contain exactly two wildcards that are instantiated by wildcard pairs such that each member of the pair can take the $\boldsymbol{X}$ or $\boldsymbol{Y}$ position. For instance, the symmetry of the pattern $\boldsymbol{X}$ or $\boldsymbol{Y}$ is illustrated by the semantically plausible expressions *"cats or dogs"* and *"dogs or cats"*. In this study, the findings revealed that these two patterns are particularly indicative of antonyms: "from $\boldsymbol{X}$ to $\boldsymbol{Y}$ and "either $\boldsymbol{X}$ or $\boldsymbol{Y}$.

IV. Methodology and Fine-Tuning Semantic Information

The methodology adopted in this study is pragmatic research philosophy. We make use of design science research to simulate a word embedding model for intelligent classification of the IoT patterns. The study also makes use of document analysis and corpus linguistics for data normalization.

A. *Information Extraction Techniques*

We have examined 109 IoT patterns, taken from both published articles and informal web pages. Of this total, 76 were peer reviewed published articles and 33 were taken

[2] One hot encoding is a binary representation of categorical data in which only a single value takes an integer value of 1 at a time (i.e. the `hot' value in the list), and the other corresponding values takes integer values of zeroes (i.e. the `cold' values in the list).

from informal web pages. First, we developed an attribute extraction system which extract the most important sentences in a document which describe the essence of each pattern. The discussion of this attributes extraction system is beyond the scope of this paper. In this paper, we only use the extracted attributes as an input to our word embedding model. Secondly, we used a Merriam-Webster thesaurus to extract synonymous words for each major part of speech word found in the extracted attributes. Finally, we harvest data from mid-sections of all the 109 documents sampled in the study. This approach is based on a hypothesis that the purpose or functions of patterns in a document are, generally, discussed in the middle section of the document. This helps to exclude superfluous data under such headings as **Introduction, Acknowledgments**, **Conclusion**, etc. which generally will not contain useful information with regard to describing the essence of a pattern.

B. Data Injected into the Word Embedding Model

As stated above, our word embedding model is injected with data from three main sources: (i) contextualized data in the form of extracted IoT attributes, (ii) enriching data taken from other secondary sources such as a thesaurus of synonyms, and (iii) data taken from generic contexts in the sampled IoT documents. Regarding the use of contexualized data which describe the IoT pattern attributes, the aim is to capture the relevant information in the context since we know that not all the information in a document or generic context describes the essence of the pattern. Data taken from secondary sources serves to enrich the extracted atrributes of the patterns. For instance, we know that due to the richness of language, different authors will express the same attribute using different word choices. This requires us to have multiple sentences expressing the same attribute which can be aligned using any technique for sentence similarity quantification. To counteract this problem, instead of adopting a sentence similarity quantification method which can be arbitrary, we opted to harvest synonymous word units from a thesaurus for each word found in the contextualized sentences (i.e. extracted attributes). In the selected thesaurus, these words are indexed by the <syn> tags. In our case, generic contexts refer to data taken from the mid-section of each document sampled in our study. The consideration of these generic contexts justifies the use of the CBOW model since it would make no sense to build a word embedding model on very restricted textual data. Nonetheless, the impact of the vocabulary taken from these generic contexts have little to no impact on the identification of a given pattern. Instead, such surrounding words give the `connotation' of the pattern.

V. Vector Space Model

Our vector space model aims to maximize the probability $\boldsymbol{p_\theta(c|w_p)}$ of the target word $\boldsymbol{w_p}$ given its context words. In this model, the context words $\boldsymbol{c}$ are supervised trained words which represents the attributes of the given pattern. The target word $\boldsymbol{w_p}$ is the pattern name manually extracted from the document and injected into the word embedding model with its respective attributes. The essence of this vector space model is this: extract the pattern name from each document (e.g. `Proxy'), extract the words from the document that describe the pattern's core attributes $\boldsymbol{c \backsim p_\theta (c|w_p)}$ (e.g. `communication', `server', `gateway', etc.), extract synonymous words for these attributes from a thesaurus (e.g. `message', `agent', `intermediary' etc.), and finally draw connotation words from generic contexts (e.g. `internet', `protocol', `cookies') . The vector space model is based on the Word2vec CBOW model, and predicts the pattern name given a set of attributes words, or connotation words for the attributes words. The power of this word embedding model lies in its capability to make a distinction between the words that describe the core essence of a given pattern and connotation words. Intuitively, this vector space model can be formalized with the below function, in which each trained context word targets the pattern name as its neighbour embedding:

$$\frac{1}{V}\sum_{v=1}^{V}\sum_{w_p}\log p_\theta\left(w_t \mid w_p\right) \tag{1}$$

In this function, $\boldsymbol{V}$ is the vocabulary size, and $\boldsymbol{w_t}$ is any trained context word that is semantically related to another word (i.e. pattern name) $\boldsymbol{w_p}$, either equivocally or unequivocally. In principle, the model also aims to compute the probabilities $\boldsymbol{p_{ij}}$ that are proportional to the similarity of words $\boldsymbol{w_i}$ and $\boldsymbol{w_j}$ in a set of $\boldsymbol{N}$ high-dimensional words $\boldsymbol{w_1,\ldots,w_N}$ as follows:

$$p_{ij} = \frac{\exp\left(-\| w_i - w_j \|^2 / 2\sigma_i^2\right)}{\sum_{k\neq i}\exp(-\| w_i - w_k \|^2 / 2\sigma_i^2)}, \tag{2}$$

Specifically, similar words that describe the same pattern are modelled by nearby points and dissimilar words are modelled by distant points with high probability. As stated above, the model draws from three contexts, semantically significant contexts (i.e. extracted attributes) denoted by $\boldsymbol{p_\theta (w_a|w_p)}$, synonymous words contexts $\boldsymbol{p_\theta(w_s|w_p)}$, and generic contexts $\boldsymbol{p_\theta(w_g|w_p)}$, in which $\boldsymbol{w \in N|w_a = w_s = w_g}$. Intuitively, the model aims to maximize the log-likelihood which sums up all three contexts, one per each window as defined by equation 3.

$$\log p_\theta(c \mid w_p) = \log\int\prod_{j=1}^{C} p_\theta(c_j \mid w)p_\theta(w_a \mid w_p)p_\theta(w_s \mid w_p)p_\theta(w_g \mid w_p)dw \tag{3}$$

In function 3, $\boldsymbol{p_\theta (c_j |w)}$ represents the artificial neural network while $\boldsymbol{c_j}$ is a context word for the target word $\boldsymbol{w}$. In this function, $\boldsymbol{C}$ is the size of the *context window*[3] and $\boldsymbol{\theta}$ is a model parameter. Due to the complexity of training neural networks, computing integration over the latent space becomes intractable. We therefore rely on objective training for the different contexts.

For each word pair, the loss is minimized using a maximum likelihood estimation technique in the form of cross-entropy. The loss function to train our neural network is thus defined as

[3] The size of the context window determines how many words before and after the target word are considered as context words.

$$L(\bar{w} \mid w) = -\sum_{j=1}^{N} w \log \bar{w} \tag{4}$$

In this function, ***N*** is the number of classes, and ***w*** sums up the outputs for the words in the class while $\bar{w}$ is the softmax probability value of the target word which must be maximized. Intuitively, that is

$$Maxp(\bar{w} \mid w) = \max \bar{w} \tag{5}$$

$$= \max \log \bar{w} \tag{6}$$

Our approach to computing the loss function as outlined above is very similar to the original Word2Vec CBOW model.

A. Training Setup

1) Corpus

The training dataset contains more than 1000 000 word tokens collected from a list of diversified sources. Specifically, these sources are documents addressing the various IoT patterns from published articles and other informal sources such as Wikipedia. Although this is a relatively small corpus, we deem it adequate for this study to demonstrate the proof of concept for our vector model. Indeed, using a larger corpus in our case will be immaterial and would not yield different results. The use of a large corpus is still, however, recommended to avoid cases of overfitting.

2) Preprocessing

The training begins with text normalization as a necessary step for data preparation. This is a data cleansing process which involves a pre-processing procedure that converts scholarly documents to sequences of text blocks at a sentence level. The IoT patterns in our dataset were mostly available as HTML files or schorlary PDF articles. Both the HTML web pages and the schorlary PDF files were generally not standardized for ready processing as they did not contain block level elements or IDs. In addition, document formatting was inconsistent across the majority of the documents. For instance, other documents contained images, and others used italics or bold text to emphasize important points and themes. To address these challenges, manual preprocessing was performed on the documents. This included converting each HTML and PDF document into XML format that recognizes all the line breaks and white spaces from the original documents. In addition, preprocessing included removing images and tagging each sentence with a unique ID or vector representation. This was done without compromising the experimental results. According to Saldanha's [10] view of corpus linguistics as a methodology, data altering for the purposes of fitting the experimental setup is an acceptable practice so long as it does not affect the authenticity or natural occurrence of the text.

3) Parameter configuration

A multi-threaded architecture is used to separately train words from specific contexts and generic contexts. In this case, words from both threads jointly and asynchronously update the output word embeddings and determine the final word vectors. In both threads, we set the symmetric context window size to 10 (i.e. 5 words to the right and 5 words to the left of the target word). We use differential learning to manipulate and control each thread's contribution to the final word embeddings. Specifically, we control the output of the final word embeddings by controlling the behavior of the stochastic gradient descent function

$$\downarrow \theta_j := \theta_j - \alpha . \frac{\partial}{\partial \theta_j} J(\theta) \tag{7}$$

We set a small learning rate, ***α = 0.01***, in specific contexts to effect better contributions from these contexts. To marginalize the contribution of generic contexts, we set a higher learning rate, ***α = 0.5***, to reduce optimization of the thread to converge to good local minima [11]. Intuitively, this results in less contribution from the generic contexts.

VI. Simulation Results

In this section, we present the evaluation results to demonstrate the performance of our vector space model. Since this paper presents a case study exploring new insights regarding the use of word embeddings in a specific domain, i.e. the Internet of Things, using pre-trained datasets or generic datasets such as Wikipedia, MEN, and Wordsim-353 would not fit the experimental setup. Instead, we compare the performance of three variations of the CBOW model. The intuition behind our experimental approach is the following: to prove our hypothesis that certain contexts exhibit more semantically related information than others, we feed these models with three variations of our dataset: (a) first, we feed the vanilla version of the CBOW model without fine-tuning the information, (b) secondly, we feed the CBOW model with fine-tuned information but without the use of differential learning, and (c) finally, we feed the differential learning CBOW model with fine-tuned information.

TABLE I: Average Cosine Similarities between Words for the CBOW Model Variation

Average Cosine Similarity Values			
Vector Model	$\boldsymbol{cos_{sim}(wp \mid wa)}$	$\boldsymbol{cos_{sim}(wp \mid ws)}$	$\boldsymbol{cos_{sim}(wp \mid wg)}$
CBOW[a]	-	-	0.47
CBOW[b]	0.84	0.83	0.81
CBOW[c]	0.84	0.83	0.68

[a] Vanilla version.

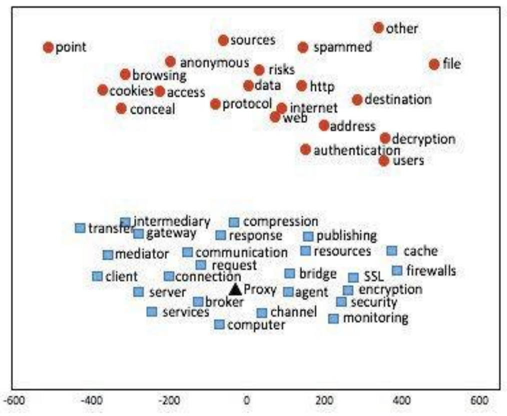
[b] Fine-tuned information without differential learning.

[c] Fine-tuned information with differential learning.

Table I presents similarity results based on the average word vectors across the three CBOW variations. First, we observe that the CBOW model which uses both fine-tuned information and differential learning has a significant edge in performance compared to the other CBOW models. In fact, the CBOW baseline model (i.e. vanilla version) is considerably weaker and provides no value-add in addressing the problem statement. The CBOW variation model which uses only fine-tuned semantic information have the highest cosine similarities for $\boldsymbol{cos_{sim}(w_p \mid w_g)}$ (0.47 vs. 0.81 vs. 0.68). However, this shows the model is ineffective in differentiating semantic properties of words in specific and generic contexts. The CBOW model containing fine-tuned information and using differential learning outperforms the other models, and possesses the capability to differentiate the pattern attributes from its generic connotation information. Figure 1 provides an example of a pattern in which the data sourced from specific contexts (attributes and synonymous words) appears relatively close to the pattern name (i.e. Proxy). The use of connotation words in our model also resolves the challenge of word sense disambiguation.

VII. Conclusion

We have presented an intelligent classification method to categorize the IoT patterns using neural word embeddings. The vector space model proposed in this paper draws from the idea that certain contexts exhibit more syntactic and semantically related information than others. Specifically, we proved the hypothesis that certain extracted information from documents allow us to mine better semantic relationships between words. In contrast, generic contexts provide coarse relationships between words, and also contain a lot a noisy data which makes training the vector space model more expensive. Unlike previous vector space models that tend to use generic data, our paper focused on using data generated for a specific domain to show the usefulness of word embeddings to solve a real-world problem. Our case study to classify the IoT patterns in vector spaces suggest the benefits of word embeddings in similar settings. Future work will focus on dimensional embeddings of common words found in both specific and generic contexts.



*Note: The data in the t-SNE visualization plot presented here has been trimmed for visibility and better presentation of the model performance results.

Fig. 1. A sample CBOW embedding visualization for the proxy pattern.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Vusi Sithole performed the experiments, organized the layout for the paper, and wrote the manuscript.

Linda Marshall reviewed and edited the original draft, provided supervision for the work and verified the results of the experiments.

Acknowledgements

The authors would like to send a sincere word of thanks and appreciation to the anonymous reviewers for their thoughtful insights into this work.

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient Estimation of Word Representations in Vector Space*, arXiv preprint arXiv:1301.3781, 2013.

[2] X. X. Chen *et al*., "Joint learning of character and word embeddings," presented at Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[3] A. Mandelbaum and A. Shalev, "Word embeddings and their use in sentence classification tasks," arXiv preprint arXiv:1610.08229, 2016.

[4] T. Scheepers, E. Gavves, and E. Kanoulas, "Analyzing the compositional properties of word embeddings," *Univ Amst*, 2017.

[5] Wikipedia, *Seventh-Day Adventist Church*.

[6] N. Tandon and G. Melo, "Information extraction from web-scale n-gram data," *Web N-gram Workshop*, vol. 7, 2010.

[7] I. Markov, G. A. Helena *et al*., "Author profiling with doc2vec neural network-based document embeddings," presented at Mexican International Conference on Artificial Intelligence, pp. 117–131, Springer, 2016.

[8] R. Collobert *et al*., "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, pp. 2493-2537, 2011.

[9] R. Schwartz, R. Reichart, and A. Rappoport, "Symmetric pattern based word embeddings for improved word similarity prediction," in *Proc. the Nineteenth Conference on Computational Natural Language Learning*, 2015, pp. 258-267.

[10] G. Saldanha, "Principles of corpus linguistics and their application to translation studies research," *Tradumàtica: traducció i tecnologies de la informació i la comunicació*, no. 7, 2009.

[11] V. Veeriah, N. Zhuang, and G. J. Qi, "Differential recurrent neural networks for action recognition," in *Proc. the IEEE International Conference on Computer Vision*, 2015, pp. 4041–4049.

Vusi Sithole is a Ph.D candidate at the University of Pretoria, Department of Computer Science with a master of information technology from the same University in 2016. He obtained a bachelor's degree in information sciences from the same institution in 2009. His research interests are in fields of software design patterns, internet of things, machine learning, graph theory, and computational linguistics. The focus of his PhD thesis is on establishing a multifaceted approach for organising patterns for the Internet of Things. His research interest include subjects such as attributes extraction, building concept hierarchies for the IoT patterns, definition formation for the patterns, and engineering a pattern locator technique for the known patterns. He has published several peer-reviewed papers in his areas of research, and continues to do ground-breaking research to advance humanity.

Linda Marshall is a senior lecturer at the University of Pretoria, Department of Computer Science. She is the head of the Computer Science Education (SCEDAR) research group at the Computer Science Department, University of Pretoria. She has been the director of the ACM ICPC South African Regional contest since 2001. Her research interests are in the areas of generic programming, graph comparison, software engineering and more particularly computer science education. She has published extensively in these areas, and continues to supervise numerous students undertaking research projects in many of these areas and other emerging fields such as artificial intelligence and machine learning.