Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Making personnel selection smarter through word embeddings: A graph-based approach

Nikos Kanakaris [*], Nikolaos Giarelis, Ilias Siachos, Nikos Karacapilidis

Industrial Management and Information Systems Lab, University of Patras, 26504 Rio Patras, Greece

ARTICLE INFO

Keywords:
Natural language processing
Text categorization
Graph representation learning
Issue management
Personnel selection
Word embeddings

ABSTRACT

This paper employs techniques and algorithms from the fields of natural language processing, graph representation learning and word embeddings to assist project managers in the task of personnel selection. To do so, our approach initially represents multiple textual documents as a single graph. Then, it computes word embeddings through representation learning on graphs and performs feature selection. Finally, it builds a classification model that is able to estimate how qualified a candidate employee is to work on a given task, taking as input only the descriptions of the tasks and a list of word embeddings. Our approach differs from the existing ones in that it does not require the calculation of key performance indicators or any other form of structured data in order to operate properly. For our experiments, we retrieved data from the Jira issue tracking system of the Apache Software Foundation. The evaluation results show, in most cases, an increase of 0.43% in the accuracy of the proposed classification models when compared against a widely-adopted baseline method, while their validation loss is significantly decreased by 65.54%.

1. Introduction

Nowadays, artificial intelligence (AI) has changed our life dramatically by improving user experience, recommending products, making predictions, automating repetitive tasks and so on. Entertainment, transportation, e-commerce, fraud detection, online customer support and medicine are just a few examples of the fields where artificial intelligence has been successfully applied on (Abduljabbar, Dia, Liyanage, & Bagloee, 2019; Awoyemi, Adetunmbi, & Oluwadare, 2017; Kanakaris, Giarelis, Siachos, & Karacapilidis, 2021; Powell, Rotz, & O'Malley, 2020; Sezer & Altan, 2021; Vanneschi, Horn, Castelli, & Popovič, 2018). Furthermore, modern Machine Learning (ML) models enable the forecasting of various ever-changing phenomena using time series data and feature selection techniques. For example, ML models may assist in predicting (i) the future price of crude oil (Karasu, Altan, Bekiros, & Ahmad, 2020), (ii) the speed of wind (Altan, Karasu, & Zio, 2021), and (iii) the economic trend of a country using time series analysis (Altan & Karasu, 2019). Despite the radical changes and improvements that AI and ML offer to the aforementioned sectors, there has not been any significant progress as far as project management and issue tracking process are concerned (Dam, Tran, Grundy, Ghose, & Kamei, 2019).

So far, the majority of existing project management systems (e.g. Jira, Wrike) assist users in mainly planning and monitoring their projects and the corresponding tasks. However, they unintentionally hide important information concerning the company itself and do not fully take advantage of the complex multidimensional data found in the hosted projects (Haidabrus, Grabis, & Protsenko, 2021). Moreover, the existence of numerous departments within a company and the diversity of viewpoints among employees make it difficult to identify and absorb the related information (Bjorvatn & Wald, 2018).

By utilizing well-tested and broadly accepted ML techniques, project management systems are able to detect valuable information and provide insights (Mahdi, Mohamed Zabil, Ahmad, Ismail, Yusoff, Cheng, Azmi, Natiq, & Happala Naidu, 2021). Hence, new opportunities can be created ranging from the ability to recommend solutions and automate simple tasks (e.g. reporting) to solving more complex problems such as finding candidate employees for a list of tasks or even automatically assigning the employees to the available tasks (Azzini, Galimberti, Marrara, & Ratti, 2018).

As far as the process of task assignment is concerned, several approaches can be found in the literature (Fatima, Azam, Anwar, & Rasheed, 2020). These approaches are divided into two distinct categories, namely the 'operations research'-based and the ML-based ones. The former view the problem of task assignment as a time-based one, which is tackled through various classical mathematical

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

[*] Corresponding author.
E-mail addresses: nkanakaris@upnet.gr (N. Kanakaris), giarelis@ceid.upatras.gr (N. Giarelis), ilias.siachos@upnet.gr (I. Siachos), karacap@upatras.gr (N. Karacapilidis).

https://doi.org/10.1016/j.mlwa.2021.100214
Received 8 August 2021; Received in revised form 4 November 2021; Accepted 7 November 2021
Available online 19 November 2021

techniques (e.g. combinatorics, optimizations etc.) or resource scheduling algorithms (e.g. genetic algorithms). The latter focus on extracting important text features, which are then fed into supervised ML algorithms in order to (i) estimate the probability for each given employee to work on a given set of tasks and (ii) assign employees to software bugs (Sajedi-Badashian & Stroulia, 2020; Tran, Le, Nguyen, & Ho, 2019). Nonetheless, the above ML-based approaches can work with any type of textual 'project management'-related data.

There is a number of advantages and disadvantages for both of the abovementioned categories. On the one hand, the task scheduling approaches can estimate the maximum number of tasks that can be concurrently processed and predict the minimum amount of time required to complete a project. However, they fail to match employees to tasks relevant to their skills, since they assume that all tasks or employees are similar to one another. On the other hand, ML-based approaches take into account the skill relevance in different tasks but require large amounts of textual data for predicting accurately the most suitable employee for each task.

In this work, we present a novel four-phase approach that assists organizations in the personnel selection process. Our approach is a ML-based one in that it utilizes textual data to assign people to tasks. The outcome of our approach is an ML model that is able to estimate how relevant or qualified a candidate employee is to work on a given task. Our ML model takes as an input a list of documents that describes completed tasks and the list of the assignees of the tasks. Our approach differs from the existing ones in that it does not require the calculation of Key Performance Indicators (KPIs) or any form of structured data in order to operate. Instead, it relies on techniques from Natural Language Processing (NLP), graph representation learning and word embeddings to reveal hidden knowledge that exists in unstructured textual data. It is domain agnostic since no additional information about the employees (e.g. their curricula or profiles) is needed to operate.

For the implementation of our approach, we use the Neo4j graph database, the Python programming language and the TensorFlow deep learning library. To evaluate our approach, we benchmark it against a widely used classification model on a dataset retrieved from the official Jira instance of the Apache Software Foundation. This dataset concerns the development of 168 open source software projects. The experimental results demonstrate a significant improvement in accuracy of the classification models generated by following our approach. The related code, dataset and evaluation results are openly accessible on the GitHub[1] platform.

Generally, project management systems encompass large amounts of unstructured textual data. Our motivation is to build a novel approach that analyzes the aforementioned data and makes recommendations related to task assignment and personnel selection by exploiting information extracted from already completed tasks.

The main contribution of this paper is four-fold:

- We provide the research community with an approach that improves existing ones by exploiting word embeddings, neural networks and graph representation learning techniques in an integrated way.
- We investigate whether the analysis of unstructured textual data benefits the personnel selection process or not.
- We propose a four-phase pipeline that assists project managers in the personnel selection process.
- We provide the research community with a rich dataset containing tasks of the projects of an organization that can be utilized as a baseline in similar research directions.

The remainder of this paper is organized as follows. Background concepts and related work are introduced in Section 2. In Section 3, our approach is presented in a thorough and elaborated fashion, while the experiments carried out to evaluate our approach are reported in Section 4. The novelty, future work directions as well as the limitations of the proposed approach are discussed in Section 5.

[1] https://github.com/imis-lab/personnel-selection.

2. Background and related work

2.1. Graph-related concepts and measures

In graph theory, numerous graph types have been developed, which are primarily differentiated on the types of vertices and edges, the features that their nodes or edges may incorporate and their general architecture (West et al., 2001). For the context of this paper, we introduce the following graph notations.

Definition 1 (*Graph*)**.** Let a graph $G = (V, E)$ be defined as a tuple consisting of a set of vertices (or nodes) V and a set of edges $E \subseteq V \times V$.

A vertex, denoted as v_i belongs to the graph G if $v_i \in V$. Similarly, an edge $e_i = (v_x, v_y)$, connecting two nodes $v_x, v_y \in V$, is part of the graph G if $e_i \in E$. The size of the graph is defined by the number of vertices $|V|$, while the number of the edges is defined as $|E|$.

Definition 2 (*Directed Graph*)**.** A directed graph is a graph $G = (V, E)$, where the edges are directed by arrows. Similarly, if the edges are not directed, the graph is called undirected.

Definition 3 (*Heterogeneous Graph*)**.** A heterogeneous graph is a graph $G = (V, E)$, in which there exists a node type mapping function $l : V \rightarrow L$, which assigns types/labels to its vertices and a link type mapping function $\hat{l} : E \rightarrow \hat{L}$ which assigns types/labels to its edges from a discrete set of types L and $\hat{L}$, respectively.

If only the vertices are labeled, the graph is called node-labeled. Similarly, in the case that only the edges are labeled, the graph is called edge-labeled, while a fully-labeled graph is considered to be a graph with both vertices and edges labeled. In this paper, we consider a heterogeneous graph to be fully-labeled.

Definition 4 (*Neighbor Nodes*)**.** A pair of vertices $v_i, v_j \in V$ of a graph $G = (V, E)$ are called neighbor nodes, if and only if the edge $e = (v_i, v_j) \in E$

For the rest of this paper, the notation $N(v)$ will denote the set of neighbors for a node v.

Definition 5 (*Jaccard Coefficient*)**.** The Jaccard Coefficient index for a pair of vertices $v_i, v_j \in V$ of a graph $G = (V, E)$, denoted by $J(v_i, v_j)$, is a statistical metric used for calculating the similarity or diversity of two nodes. Specifically, it considers the amount of the intersection of their neighbor nodes (the set of common neighbors) over the union of them (the set of nodes that are neighbors with either v_i or v_j) (Jaccard, 1901). It is defined as:

$$J(v_i, v_j) = \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$$

Definition 6 (*Cosine Similarity*)**.** The Cosine Similarity measure for a pair of vectors $\vec{x}, \vec{y} \in \mathbb{R}^D$, denoted by $cos(\vec{x}, \vec{y})$, is a similarity metric calculated between two non-zero vectors, which is commonly calculated as the Euclidean dot product between vectors x, y. It is defined as:

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \, \|\vec{y}\|} = \frac{\sum_{n=1}^{D} x_i \cdot y_i}{\sqrt{\sum_{n=1}^{D} x_i^2} \cdot \sqrt{\sum_{n=1}^{D} y_i^2}}$$

In this paper, we utilize the cosine similarity to calculate the similarity between two word embeddings (Section 2.3).

2.2. Graph-based text representations

In the *graph-of-words* textual representation (Rousseau & Vazirgiannis, 2013), every document of a corpus is depicted as a single graph. Particularly, each graph node resembles a unique word of a document and each edge that connects a pair of vertices denotes the co-occurrence between the corresponding words in the document. Co-occurrence, as a term, stands for two words appearing simultaneously within a sliding window of text; as suggested in Rousseau, Kiagias, and Vazirgiannis (2015), the length of the sliding window should not exceed four words, since there are diminishing returns in the accuracy of the ML models. This approach (i.e. taking into consideration not only the appearance of words in a document, but also the co-occurrence among them) allows for a more sophisticated feature extraction and selection, compared to the *bag-of-words* representation. However, *graph-of-words* bears a great deal of limitations, as it does not allow the assessment of the importance of a word for the whole corpus, neither does it depict multiple documents in a single graph. In addition, there is no intuitive way to support more complex data architectures, thus reducing the expandability of this particular representation.

In an effort to tackle these restrictions, Giarelis, Kanakaris, and Karacapilidis (2020b) suggested the *graph-of-docs* representation, where a single graph can incorporate multiple textual documents. The construction of a heterogeneous graph, where words and documents are represented with different nodes, offers the following advantages: (i) we can easily calculate the importance of a term in a whole set of documents and (ii) we can configure the graph effortlessly, in order to include more complicated data structures.

2.3. Word embeddings

Generally speaking, word embeddings is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of a word such that the words that are closer in the vector space are expected to be similar in meaning. A wide variety of techniques for determining word embeddings have been proposed in the literature (Almeida & Xexéo, 2019). The most popular ones include *Word2Vec* (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013), *GloVe* (Pennington, Socher, & Manning, 2014) and *fastText* (Joulin et al., 2016). These techniques are extensively utilized in a plethora of typical NLP tasks, such as text classification and sentiment analysis, or even in more complex tasks, such as spam detection and question-answering.

The employment of pre-trained word embeddings guarantees accuracy improvement of an ML model, mitigates overfitting and facilitates the generalization of the model (Andreas & Klein, 2014; Kholghi, De Vine, Sitbon, Zuccon, & Nguyen, 2016). Practically, large datasets, which are usually domain or language specific, are used for training word embeddings. Hence, the produced word representations are able to capture statistical correlations between words and conclusively, generate embeddings for a specific NLP domain/task (e.g. finding animals similar to a mustang horse or finding cars similar to a Mustang). Usually, pre-trained word embeddings are used as a starting point, which are then adjusted for the specific task in hand. *GoogleNews* using Word2Vec and *Common Crawl* using GloVe are some of the most favored public pre-trained word embeddings. For an in-depth analysis as far as the differences and use-cases of various pre-trained word embeddings are concerned, we refer to van der Heijden, Abnar, and Shutova (2020).

Additionally, we note that most of the techniques for extracting word embeddings have been used as core elements for producing sentence, paragraph and document embeddings, correspondingly (Kusner, Sun, Kolkin, & Weinberger, 2015; Ye, Shen, Ma, Bunescu, & Liu, 2016). For example, *Doc2Vec* (Le & Mikolov, 2014) depends on Word2Vec to map each document of a corpus as a vector.

2.4. Graph embeddings

Feature extraction and selection in graph-based ML techniques, such as node classification (i.e. predicting the labels of nodes in graphs), have always been a challenging task. The nature of graph representations makes it impossible to create features automatically, as they usually depend on specific graph properties. However, graph embeddings seek to remedy this shortcoming by mapping graphs to a vector or a set of vectors, thus reducing the feature extraction task to a representation learning one. Graph embeddings can be applied on node level (i.e. producing a vector representation for each node of a graph) or graph level (i.e. mapping a graph to a vector representation). In this paper, we focus our interest on the former. The basic concept behind node embeddings is to create a vector representation for a node by capturing high-dimensional information regarding its neighborhood and utilizing dimensionality reduction techniques to compress this information into a low-dimensional feature space.

A variety of techniques for producing node embeddings has been proposed in the literature. The majority of these methods suggest a two-step process, as it was first introduced by *DeepWalk* (Perozzi, Al-Rfou, & Skiena, 2014). *DeepWalk* is inspired by language modeling techniques and proposes the construction of a set of random walks on a graph. These random walks are fed to a *SkipGram* model (Mikolov, Chen, Corrado, & Dean, 2013a) to produce a matrix of vector representations (node embeddings), in a similar fashion to how a *SkipGram* model maximizes the co-occurrence probabilities among the words that appear within a window.

Node2Vec (Grover & Leskovec, 2016) is a semi-supervised representation learning graph algorithm that maximizes the probability of maintaining the structure of neighborhoods of nodes. It bears a great resemblance to *DeepWalk*, with the process of generating random walks being the factor that differentiates each other. Specifically, *Node2Vec* establishes second order random walks, which can be perceived as random walks that each step relies on input parameters. These parameters introduce a bias, as far as different graph traversal strategies are concerned (e.g. Breadth-first Sampling, Depth-first Sampling). *Node2Vec* manages to learn feature space representations for the nodes, based on the graph properties and the neighborhoods they share.

Graph Embedding techniques that utilize *SkipGram* or other deep neural techniques, such as *DeepWalk*, *Node2Vec* and *LINE* (Tang et al., 2015), usually suffer from a performance bottleneck, because of their computationally expensive nature, which results in an increased execution time. *FastRP* (Chen, Sultan, Tian, Chen, & Skiena, 2019) aims to alleviate this downside without downgrading the quality of the produced node embeddings. This is achieved by capturing the implicit relationships among nodes and then applying normalization of the similarity between each node in the graph. The normalization procedure uses the node degrees and a scalable dimension reduction algorithm, i.e. Very Sparse Random Projection to produce node embeddings in a time efficient way, compared to *SkipGram* and *SVD* (Mikolov, Chen, Corrado, & Dean, 2013b).

GraphSAGE (Hamilton, Ying, & Leskovec, 2017a) is an inductive algorithm for producing node embeddings. Unlike the abovementioned algorithms that inherently calculate the node embeddings for a fixed graph (and thus producing poor results for newly added vertices, that did not participate in the training phase), *GraphSAGE* learns an embedding function, which can be generalized to include newly added nodes. The embedding function depends on node features, such as textual properties and node degrees, although the adaptability of this technique enables users to apply *GraphSAGE* to graphs that may or may not contain node attributes, as well as to fixed or dynamically expanding graphs.

2.5. Human resource allocation

Human resource allocation (i.e. assigning the most qualifying personnel to the most appropriate task) is admittedly a difficult and complex process, with bad selections resulting in major time and cost repercussions for a company. In the recent years, the concept of automating human resource allocation has gained interest, leading to a variety of approaches originating from diverse research fields.

A wide range of existing human resource management tools rely on concepts and methods from the area of recommendation systems. For instance, Alkhraisat (2016) proposes the utilization of document similarity techniques and ontology, and semantic similarity measures for the generation of an automated issue tracking system. The proposed approach discovers similar issues and recommends candidate experts as well as related materials to address each issue. Heyn and Paschke (2013) describe an extension for the Jira platform, which works in a similar fashion as the abovementioned work, enabling the re-usability of similar work and the proper dissemination of work to the right developers. Another expertise recommendation engine can be found in Truica and Barnoschi (2019). This work utilizes text mining and NLP techniques to infer knowledge from resumes. In the next phase, the system produces a recommendation score for each candidate by semantically matching the retrieved information with job skills and requirements that are stored in a database.

The employment of appropriate Machine Learning algorithms has gained interest lately for manufacturing innovative personnel selection applications. For instance, the authors in McDonald and Ackerman (2000) propose a novel approach for evaluating job candidates by utilizing ML algorithms to extract a given job's requirements and the skills of the applicants by their LinkedIn profile. Having extracted these features, a ranking of the applicants is created based on the semantic similarity of the candidates' skills with the job prerequisites. In addition, the authors in Azzini et al. (2018) describe a knowledge extraction process about academic candidates from a semi-structured interview. To this end, they employ various ML classifiers in order to discover the soft skills of individuals and match these skills with job requirements. Their empirical results demonstrated that Support Vector Machine (SVM) and naive Bayes (NB) approaches produce better results than k-Nearest Neighbors (k-NN), decision trees and random forests. Another application of ML classifiers can be found in Wowczko (2015), where job advertisements in Ireland in 2014 were processed to extract job titles and descriptions, which then served as input data. A categorization of jobs was eventually developed, affiliating each job categorization with a set of skills.

2.6. Task assignment

Helming, Arndt, Hodaie, Koegel, and Narayan (2011) built an automatic assignment ML system for work items (e.g. tasks and bug reports). Their system models relationships between work items as functional requirements (e.g. work A needs to be implemented before work B), extracts significant textual features from work items using TF-IDF, while the combined relationships and features are used for training various supervised learning classifiers such as NB, SVM and neural networks. Their implementation relies on known ML-oriented java libraries and WEKA.

Jonsson et al. (2016) built an automated bug assignment system that follows a similar approach as the one proposed in Helming et al. (2011). Their work differs in that they only extract textual features from successful bug reports, using TF-IDF as provided by the WEKA Java library. Then, they train multiple classifiers that assign bug reports to developer teams. After the classifiers are trained, they are merged into a generalized classifier, which takes into account predictions from all former classifiers. Overall, they report an accuracy ranging from 50% to 89% with higher accuracy scores correlated to more training data.

Mo et al. (2020) built an automated staff assignment ML system for building maintenance workers using NLP techniques. Particularly, they utilize a bag-of-words implementation to extract meaningful features from textual work descriptions and then predict which work group of employees is assigned for this work. The unigrams extracted from the bag-of-words implementation are fed into three supervised learning classifiers, namely Linear Regression (LR), NB, and SVM. Overall, their experimental results indicate an accuracy of 77% and 88% for predicting correctly the assigned workforce and work priority, respectively.

Hassanien and Elragal (2021) utilize Word2Vec embeddings that are fed to a siamese long short-term memory neural network, which calculates the semantic similarity between texts of work descriptions found in enterprise resource planning systems. Although this work is not strictly related to the task assignment problem, it is the most similar to our approach since it utilizes word embeddings to capture the contextual information between important text features.

3. Our approach

In this section, we first define the preliminary concepts and the mathematical notation needed to describe our methodology. Then, we analytically present the proposed approach.

3.1. Preliminary concepts

Let $I = [i_1, i_2, \dots, i_N]$ be a list of textual descriptions of issues or tasks of an organization, where $N \in \mathbb{N}$ (terms *issue* and *task* are used interchangeably throughout this paper). Let $A = [a_1, a_2, \dots, a_K]$ be a list of assignees of an organization, where $K \in \mathbb{N}$. Let $W = [w_1, w_2, \dots, w_M]$ be the list of unique words from the textual descriptions I, where $M \in \mathbb{N}$. Let $E = [\vec{e_1}, \vec{e_2}, \dots, \vec{e_M}]$ be a list of word embeddings, where $\vec{e_x} \in \mathbb{R}^D$ is a vector of D dimensions that corresponds to the vector representation of word w_x. Let $F = [f_1, f_2, \dots, f_n] \subseteq W$ be a feature space, where f_x denotes that the word w_x has been selected as a feature and the number of features $n \leq M$.

Let $GD = (V, E)$ be a heterogeneous *graph of docs*, where V is the set of vertices and E is the set of edges of the graph. An edge $e_i = (v_x, v_y)$ links a vertex v_x to a vertex v_y. Each vertex v_i is associated with a label $l_i \in \{word, issue\}$ and each edge e_i is associated with a label $\hat{l}_i \in \{includes, co\_occurs\}$. An edge $e_i = (v_x, v_y)$ with a label $\hat{l}_i = co\_occurs$ can only connect two vertices when $l_x = l_y = word$. Similarly, an edge $e_i = (v_x, v_y)$ with a label $\hat{l}_i = includes$ can only connect two vertices when $l_x = issue$ and $l_y = word$. Let $WSG = (V', E')$ be a homogeneous word similarity graph, where $V' \subset V$ is the set of vertices with a label $l'_i = word$ for each vertex $v'_i \in V'$ and E' the set of edges of the graph with a label $\hat{l}_i{}' = is\_similar$ for each edge $e'_i \in E'$. An edge $e'_i = (v'_x, v'_y)$ links a vertex v'_x to a vertex v'_y and denotes a similarity greater than a predefined threshold between the words w_x, w_y as far as the *cosine similarity* of their embeddings $\vec{e_x}, \vec{e_y}$ is concerned. Let $ISG = (V'', E'')$ be a homogeneous issue similarity graph, where $V'' \subset V$ is the set of vertices with a label $l''_i = issue$ for each node $v''_i \in V''$ and E'' the set of edges of the graph with a label $\hat{l}_i{}'' = is\_similar$ for each edge $e''_i \in E''$. An edge $e''_i = (v''_x, v''_y)$ links a vertex v''_x to a vertex v''_y and denotes a *Jaccard index score* greater than a predefined threshold between the textual descriptions i_x, i_y.

3.2. Methodology

The aim of our work is to build an ML model that is able to estimate how relevant or qualified a candidate employee is to work on a given task i_x. Descriptions of past completed tasks and the names of assignees are required as input. Our approach consists of four main phases which are described in depth in the following sections and can be summarized as follows (see also Fig. 1 for a more visualized summarization):

- **Representing assignees and issues as a graph-of-docs** (Section 3.2.1). Transforming a list of issues I and a list of assignees A into a graph-of-docs GD, as proposed in Giarelis et al. (2020b).
- **Calculating graph-based word embeddings** (Section 3.2.2). Training a Word2Vec model on the list of descriptions of the issues I in order to construct a similarity subgraph between words appearing as nodes in the GD. In the next step, we employ algorithms for representation learning on graphs (e.g. Node2Vec and GraphSAGE) to produce a list of word embeddings for each word using the similarity graph WSG as input. The abovementioned process enables us to fine-tune the trained list of word embeddings E.

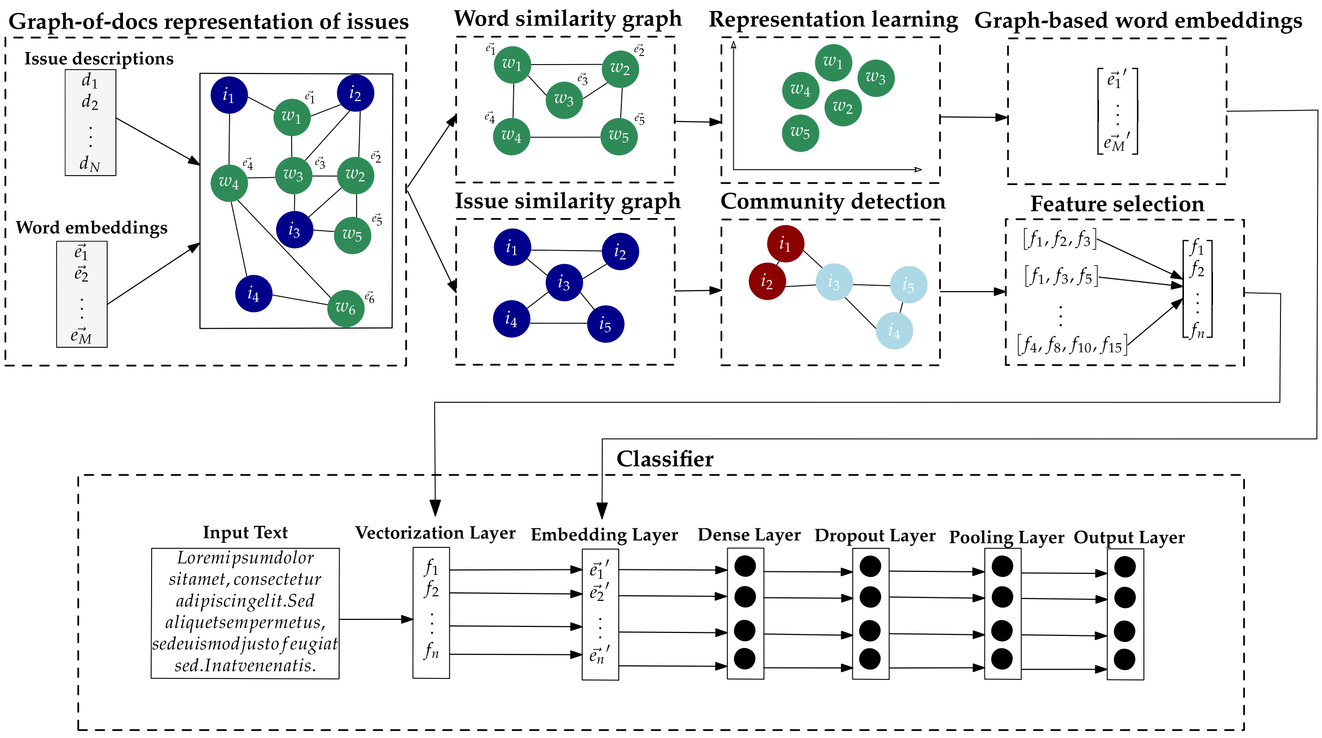

Fig. 1. An overview of the proposed approach. i_x denotes a node that corresponds to an issue or task. w_x denotes a node that corresponds to a word from the descriptions of the issues or tasks. The word embedding of a word (w_x) is denoted by $\vec{e_x}$, while the enhanced version for the same word is denoted by $\vec{e_x}'$. f_x denotes a word that is selected as a feature for the final neural network classification model. The color of the nodes in the community detection step denotes different groups of textually similar issues. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- **Feature selection** (Section 3.2.3). Defining a feature space F by identifying important terms in communities of textually similar issues, as proposed in Giarelis, Kanakaris, and Karacapilidis (2020a).
- **Estimating the relevance of an employee** (Section 3.2.4). Training a neural network, where the input is the description of an issue i_x and the output is the probability of how capable an assignee a_x is in completing the given issue.

3.2.1. *Representing issues as a graph-of-docs*

The first phase consists of integrating the list of textual descriptions I and the list of assignees A into a single graph, according to the graph-of-docs representation (Giarelis et al., 2020b). The construction of a heterogeneous graph-of-docs representation allows nodes with different labels (i.e. '*issue*', '*word*' and '*assignee*' labels) to co-exist into a single graph. Similarly, the graph incorporates relationship edges with various labels (i.e. '*co_occurs*', '*includes*', '*is_similar*' labels). As mentioned above, the graph-of-docs representation facilitates the investigation of the global importance of a word, as a word can be linked to multiple documents. Finally, the existence of similarity edges between issues and words enables us to calculate well-established metrics, as far as their similarity is concerned. Before the construction of the *GD*, a training step is performed. Specifically, the descriptions are used to train a Word2Vec model in order to calculate a word embedding for each word that appears in the graph.

The constructed heterogeneous graph GD contains all the connections between the assignees, issues and the words of a corpus. Edges with an '*includes*' label are utilized to link each unique word of the corpus to every issue node which contains this particular word. Two word nodes are connected through a relationship edge with a label '*co_occurs*', which captures their co-existence within a sliding text window of fixed length. An assignee is connected with an issue node through a relationship edge with a label '*is_assigned_to*'. Last but not least, the word similarity subgraph WSG connects two highly similar word vertices based on their pairwise cosine similarity score, which is calculated by their trained Word2Vec embeddings.

The above representation of a corpus of issues as a heterogeneous graph enables us to reduce the human resource management problem to well-known and already studied graph tasks. By exploring important structural characteristics of a graph, such as node centrality and frequent subgraphs, these methods could identify important features, discover similar issues, generate issue clusters based on their similarity and even enrich existing word embeddings, through graph representation learning on the associated similarity subgraph.

3.2.2. *Calculating graph-based word embeddings*

This phase takes as an input the generated instance of the similarity subgraph *WSG*, which captures the similarity between words through the list of word embeddings *E*. The aim of this phase is to produce the enriched word embeddings list E' by fine-tuning the list E. To do so, we employ graph representation learning algorithms, which consider neighboring nodes as semantically similar. This enables us to utilize the existing word embeddings by translating their semantic similarity into a graph structural similarity. The graph representation algorithms take as input the word similarity subgraph *WSG*. Each edge of the graph also has a '*score*' $\in [0.0, 1.0]$ property, which reflects the pairwise cosine similarity percent based on context provided by the initial list of word embeddings *E*. For instance, for two synonym words w_x, w_y the score property is ≈ 1, whereas for two antonym words w_x, w_y the score property is ≈ 0. Edges with a score value less than a predefined threshold are removed, aiming to avoid any information exchange between dissimilar nodes during the representation learning step. In this paper, we employ *Node2Vec* and *GraphSAGE* as the main graph representation learning algorithms. Section 4.3 provides a detailed explanation of how we utilize the aforementioned algorithms. Finally, an in-depth review of graph representation learning algorithms can be found in Hamilton, Ying, and Leskovec (2017b).

3.2.3. Feature selection

The task of the feature selection step of our pipeline is to produce a feature space F from the produced instance of graph-of-docs representation GD. This phase can be distinguished into four steps: (i) creating an issue similarity graph ISG, (ii) clustering (i.e. discovering communities) textually similar documents, (iii) performing feature selection for every one of the discovered clusters and (iv) producing an overall feature space F by combining the separate feature spaces created in the previous step (see also Algorithm 1).

The core idea behind the creation of an issue similarity graph ISG is that issues with textual similarities are expected to share a fair amount of common word nodes and structural attributes. Thus, by employing typical data mining similarity measures, we are able to recognize the underlying patterns of the graph and calculate the (textual) similarity between two issues i_x, i_y. Having calculated the similarity for each pair of issue nodes, the construction of the similarity graph ISG is a straightforward process. The issue similarity graph consists of issue nodes and edges with an '*is_similar*' label. The edges are also weighted with a '$score$' $\in [0, 1]$, which denotes how similar two nodes i_x, i_y are as far as their textual descriptions are concerned.

In the second step, the issue similarity graph facilitates the discovery of communities of textually similar issues. To this end, we define a distance value for every pair of issues, which is equal to the '*score*' property of the edge connecting the corresponding issue nodes. As far as clustering is concerned, a plethora of community detection algorithms can be found in the literature, such as Louvain (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008), Weakly Connected Components (Levorato & Petermann, 2011) and Label Propagation (Zhu & Ghahramani, 2002). In this paper, we employ Louvain as the main community detection algorithm.

In the next step, we assume that issues that are part of the same community are also likely to share common features. To extract the most common features from every community, we prioritize the *top-N* terms of each cluster, firstly by their descending score of document frequency in the community, and secondly by their descending *PageRank* score. These *top-N* words serve as the feature space for every discovered community.

Finally, by merging the *top-N* terms of each cluster, we derive the global feature space F. The final feature space derived from the above process is bound to be significantly smaller than the original feature space. This dimensionality reduction (i) accelerates the feature extraction and selection process, (ii) restricts the repercussions of the '*curse-of-dimensionality*' phenomenon, and (iii) increases the overall robustness and reliability of the ML models.

Algorithm 1: Feature selection

```
Input: an instance of the graph-of-docs representation $GD = (V, E)$
Result: a feature space $F$
// Step 1
IV ← get_all_issue_nodes($GD$);
ISG ← initialize_graph();
for each node $v$ in $IV$ do
    for each node $u$ in $IV$ do
        similarity_score ← calculate_similarity($v$, $u$, Jaccard);
        add_nodes_to_graph(ISG, $v$, $u$, similarity_score);
    end
end
// Step 2
communities ← [];
for each node $v$ in $ISG$ do
    add_node_to_a_community(communities, $v$, Louvain);
end
// Step 3
communities_topN ← [];
for each community $c$ in communities do
    $WV$ ← get_all_word_nodes_of_community($c$);
    for each node $v$ in $WV$ do
        scores.add(calculate_centrality_score($v$, PageRank));
    end
    topN ← find_topN_words($WV$, scores, N);
    communities_topN.add(topN);
end
// Step 4
$F$ ← [];
for each topN in communities_topN do
    $F$ ← unique($F \cup topN$);
end
return $F$;
```

3.2.4. Estimating the relevance of an employee

This phase takes as an input the list of the selected features F, the produced graph-based word embeddings E' and the list of the textual descriptions of the issues I. It trains a neural network that is capable of predicting how suitable an employee is to undertake a specific issue. The proposed neural network is composed of 7 layers: (i) an *input* layer; (ii) a *vectorization* layer, where the input text is transformed into a list of features, retaining only the terms that are included in the feature space F; (iii) an *embedding* layer, where the terms are converted into their equivalent embedding representations inferred from the list E'; (iv) a *dense* layer; (v) a *dropout* layer, which mitigates overfitting; (vi) a *pooling* layer, where the embedding for the whole text of an issue is calculated by aggregating the embedding representations of the features, using a predefined pooling function such as *max* or *average* pooling; (vii) an *output* layer, where the final relevance scores are calculated by an activation function such as *softmax*.

At this point of our pipeline process, the relevance classification problem has been reduced to a text classification one with multiple classes. The names of the candidate employees serve as the set of the model classes. A classification algorithm is utilized to accurately predict the class, which will be assigned to every issue (based on the probability scores produced by the classifier). Despite the fact that the algorithm assigns only one class to every issue, we can handle this limitation by examining the probabilities generated for the rest of the classes and thus inferring how relevant each candidate employee is with a given issue. In other words, the relevance of each employee with a task can be quantified as the probability score calculated by the ML classifier for the corresponding class.

3.3. Comparative assessment

Contrary to the work of Helming et al. (2011) and Jonsson et al. (2016) which use a statistical approach, namely TF-IDF, we propose a graph-based approach along with word embeddings to extract the most significant terms. The statistical approach of the aforementioned approaches measures the significance of terms across multiple documents; however, it fails to capture relationships between the candidate terms in the feature extraction phase. These relationships are captured by graph-based approaches that rely on co-occurrences such as the one proposed in this paper. TF-IDF also fails to capture context between terms, as it does not produce word embeddings, which generally capture the similarity between terms.

Moreover, compared to the work described in Mo et al. (2020), where simpler classifiers such as LR, SVM, and NB are used, our approach utilizes a neural network classifier. This results in an improvement in the overall performance, regardless of the selected text representation. Finally, we further expand the approach presented in Hassanien and Elragal (2021), which utilizes Word2Vec embeddings to calculate the similarity between two different tasks. In particular, we utilize the semantic similarity of our trained Word2Vec embeddings as a preparatory step, for training our classifier for the task assignment

Table 1
The attributes of each sample of the dataset.

Attribute	Description	Values	Attribute type
Assignee	The name of the assignee of the issue	e.g. {john_doe, jane_doe}	String
Description	The description of the issue	unstructured text	String

Table 2
Descriptive statistics of each subset of the dataset.

Dataset	Number of unique classes	Number of samples	Evaluation results
Dataset 1	3 (most frequent)	37,695	Table 3
Dataset 2	4 (most frequent)	39,259	Table 4
Dataset 3	21 (most frequent)	57,798	Table 5
Dataset 4	300 (all)	228,969	Table 6

process. The utilization of the aforementioned state-of-the-art techniques enables the generation of ML models that are capable of deeply understanding unstructured textual data, which can be found in project management systems. We argue that the proposed approach is highly suitable for the problem of task assignment since it (i) captures complex relationships between the words of a document, (ii) identifies synonym terms and words, and (iii) utilizes neural networks, which are essential for building robust ML models for the analysis of multi-dimensional textual data.

4. Experimental evaluation

As far as the evaluation of our approach is concerned, the Python programming language and specifically, the TensorFlow deep learning library (Abadi et al., 2015) were utilized to build our classification models. In addition, the initial word embeddings list represents each word as a one-hot vector. Storing the graph data, as well as the procedure of enhancing the initial word embeddings was accomplished through the Neo4j graph database[2] using representation learning techniques. The full code of the experiments is freely available on the GitHub repository[3] of the paper.

4.1. Dataset

The original dataset used for the evaluation of this paper consists of *168* software projects, including *Spark*, *Hadoop* and *Airflow*. The information contained to this dataset concerns *228,969* Jira issues, with every issue incorporating the attributes *description* and *assignee* (see Table 1). The *assignee* attributes act as the class labels for the issues of the dataset. The dataset is available on Google Drive[4] and was fetched from the publicly accessible Jira instance of *Apache Software Foundation*.[5]

Concerning the analysis of how each classification model reacts with respect to the number of samples as well as unique employees (classes), we created four subsets of the original dataset, where we conducted 10-fold cross-validation classification experiments. The details (number of classes, number of samples and evaluation results) for every subset are provided in Table 2.

[2] https://neo4j.com/.

[3] https://github.com/imis-lab/personnel-selection.

[4] https://drive.google.com/file/d/19PnIT-hJtGk60ntdymo3vsw8o2Qs4Zsn/view?usp=sharing.

[5] https://issues.apache.org/jira, retrieved at: 20 December 2019.

4.2. Evaluation metrics

Two well-established metrics, namely *accuracy* and F_1 score are used to evaluate our approach. These metrics are defined as follows:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

where *TP* stands for true positives, *TN* for true negatives, *FP* for false positives, *FN* for false negatives predictions of each classification, and finally *precision* and *recall* are defined as follows:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Specifically, accuracy calculates the percentage of correct classifications, whereas F_1 expresses the harmonic mean of the *precision* and *recall*.

For a more elaborate evaluation of our model, we also calculate the average difference between the training and test loss of every neural network, where the sparse categorical cross-entropy was utilized as the loss function. The fact that this metric remains unaffected from the order of the data used to train the classifier is the main benefit of taking into consideration such a metric. In particular, sparse categorical cross-entropy captures how dissimilar the distribution of the real values of class labels is compared to the predicted values of the model, which in turn indicates whether the given classifier generalizes well or not.

Finally, to examine whether the improvement in the abovementioned metrics of our approach is statistically significant, we perform a micro sign test with a *p-value* = 0.05 for every classifier.

4.3. Classification models

In this section, we present the classification models used to evaluate the proposed approach. We consider the *bow_100x50* classifier as our baseline model, since it is the simplest and the most used model as far as the task of text classification is concerned.

- **bow_100x50**: A neural network model with 2 hidden dense layers with 100 and 50 units, respectively, and a dropout layer of 0.5. It also utilizes the bag-of-words model as a text representation. During the training phase, we use 15 epochs, a batch size of 128, the sparse categorical cross entropy as a loss function, the ReLU activation function for the 2 hidden layers, the softmax activation function for the output layer and the Adam optimizer.
- **graphsage_{100, 200, 300}**: A neural network model with a trainable embedding layer with 100, 200 or 300 dimensions, a hidden dense layer with 100 units, a dropout layer of 0.5 and a global max pooling layer. The initial weights of the embedding layer have been calculated by running the *GraphSAGE* algorithm on the corresponding word similarity graph *WSG* for 5 epochs. During the training phase of the neural network, we use 15 epochs, a batch size of 128, the sparse categorical cross entropy as a loss function, the ReLU activation function for the hidden layer, the softmax activation function for the output layer and the Adam optimizer.
- **node2vec_{100, 200, 300}**: A neural network model with a trainable embedding layer with 100, 200 or 300 dimensions, a hidden dense layer with 100 units, a dropout layer of 0.5 and a global max pooling layer. The initial weights of the embedding layer have been calculated by running the *Node2Vec* algorithm on the corresponding word similarity graph *WSG* for 5 iterations. During the training phase of the neural network, we use 15 epochs, a batch size of 128, the sparse categorical cross entropy as a loss function, the ReLU activation function for the hidden layer, the softmax activation function for the output layer and the Adam optimizer.

Table 3
Classification *accuracy* and F_1 score (± standard deviation) of each model on the subset of the dataset concerning the 3 most frequent assignees (Dataset 1). All the models have statistically significant loss (last column) improvement against the *bow_100* × *50* baseline. Bold font indicates the best score value for each column.

Method	Accuracy	F_1	Train loss	Validation loss	Abs loss difference
bow_100 × 50	95.11(±0.52)	95.11(±0.52)	**0.0023**	0.4029	0.4005
graphsage_100	94.12(±0.46)	94.12(±0.46)	0.0037	0.2767	0.2730
graphsage_200	94.10(±0.48)	94.10(±0.48)	0.0040	0.2839	0.2799
graphsage_300	94.04(±0.46)	94.04(±0.46)	0.0048	0.3040	0.2991
node2vec_100	95.20(±0.32)	95.20(±0.32)	0.0218	**0.1252**	**0.1034**
node2vec_200	95.16(±0.27)	95.16(±0.27)	0.0186	0.1359	0.1173
node2vec_300	**95.28(±0.25)**	**95.28(±0.25)**	0.0166	0.1426	0.1260

Table 4
Classification *accuracy* and F_1 score (± standard deviation) of each model on the subset of the dataset concerning the 4 most frequent assignees (Dataset 2). All the models have statistically significant loss (last column) improvement against the *bow_100* × *50* baseline. Bold font indicates the best score value for each column.

Method	Accuracy	F_1	Train loss	Validation loss	Abs loss difference
bow_100 × 50	94.42(±0.38)	94.42(±0.38)	**0.0039**	0.4487	0.4448
graphsage_100	93.50(±0.44)	93.50(±0.44)	0.0048	0.2788	0.2740
graphsage_200	93.33(±0.36)	93.33(±0.36)	0.0058	0.3028	0.2970
graphsage_300	93.15(±0.37)	93.15(±0.37)	0.0068	0.3233	0.3165
node2vec_100	**94.49(±0.32)**	**94.49(±0.32)**	0.0297	**0.1446**	**0.1149**
node2vec_200	94.42(±0.45)	94.42(±0.45)	0.0243	0.1542	0.1299
node2vec_300	94.43(±0.38)	94.43(±0.38)	0.0223	0.1624	0.1402

4.4. Evaluation results

To evaluate our approach, we benchmark the classification models presented in Section 4.3 against the *bow_100x50* classifier on a list of datasets with different numbers of samples and classes. The obtained results (Tables 3–6) indicate that classifiers, which are based on embeddings produced by the Node2Vec algorithm, are the ones that generally perform better (concerning accuracy and F_1 metrics), regardless of the properties of the given dataset. In addition, a statistically significant improvement in accuracy has been recorded for the majority of the proposed classifiers on *Dataset 4*.

It is also observed that the dimension of the embeddings does not affect significantly the performance of the classifier, since models relying on the same representation learning algorithm produce similar results. However, we notice that the dimension of the embeddings affects the generalization process of the classifier, since all models have statistically significant loss improvement against the *bow_100x500* classifier on each one of the four datasets (see Fig. 2). Moreover, among all models, *node2vec_100* achieves the lowest absolute loss difference on each dataset. Finally, another aspect that affects the performance of a classifier is the size and the number of the unique classes of a dataset. This happens mostly due to the fact that the datasets are not balanced and, in many cases, there is a limited number of samples for a specific class, which in turn does not allow a classification model to generalize easily. This last observation is also noted in Jonsson et al. (2016).

To provide an illustrative evaluation of our approach, we visualize the embeddings generated from (i) the pooling layer of the proposed approach, and (ii) the last hidden layer of the baseline model (see Fig. 3). To do so, we first reduce the dimensions of the embeddings to 20 using the principal component analysis process; then, we use the t-SNE tool (Maaten & Hinton, 2008) to visualize the final embeddings in a two-dimensional space. For the sake of clarity, we only visualize the embeddings learned from *Dataset 2* (Table 4) through the *node2vec_100* (highest accuracy) and *bow_100x50* (baseline) models. As illustrated in Fig. 3, the proposed approach produces embeddings that results in a better separation of the documents with respect to the associated class. In other words, documents with the same label are close to each other and the different groups of documents are better separated using the proposed approach. This explains why the majority of the proposed models achieves an improvement in accuracy and F_1 scores.

Considering the above observations, we conclude that the proposed approach is more likely to produce efficient classifiers with a better generalization process and better document or word embeddings.

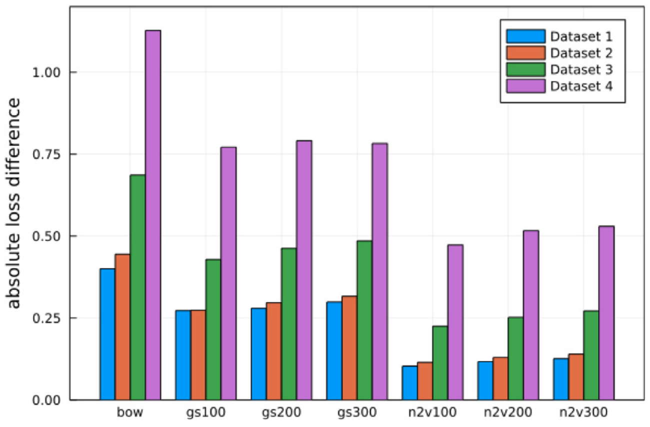

Fig. 2. Absolute loss difference of the classification models on each subset of the original dataset concerning the 3 (**Dataset 1**), 4 (**Dataset 2**), 21 (**Dataset 3**) or 300 (**Dataset 4**) most frequent employees.

5. Discussion and conclusion

In this paper, we propose a novel approach that exploits techniques from various research fields, such as natural language processing, graph representation learning and word embeddings, to assist project managers in making smarter and evidence-based decisions in the personnel selection process. By properly assigning the most qualified personnel to the appropriate tasks, the chance of success of a project is expected to significantly increase. To this end, our approach estimates a relevance score between a given task X and an employee Y, by estimating the probability that the employee Y possess the skills required by task X. The neural networks developed for the calculation of this probability reveal hidden knowledge residing in unstructured textual data, which otherwise would remain untapped. In addition, our approach is domain agnostic, as it does not require additional information about the tasks (e.g. keywords that reflect the necessary skills and competences to perform a given task) or the employees (e.g. one's curriculum or social media profile).

For our experiments, we retrieved data from the Jira issue tracking system of the Apache Foundation, which we enriched with information from word embeddings. As demonstrated by the evaluation results, the proposed approach contributes to the increment of the classification accuracy (compared to a widely-adopted baseline method), which in

Table 5
Classification *accuracy* and F_1 score (± standard deviation) of each model on the subset of the dataset concerning the 21 most frequent assignees (Dataset 3). All the models have statistically significant loss (last column) improvement against the *bow_100 × 50* baseline. Bold font indicates the best score value for each column.

Method	Accuracy	F_1	Train loss	Validation loss	Abs loss difference
bow_100 × 50	**87.11**(**±0.28**)	**87.11**(**±0.28**)	0.0638	0.7501	0.6863
graphsage_100	85.70(±0.48)	85.70(±0.48)	0.0595	0.4885	0.4289
graphsage_200	85.59(±0.53)	85.59(±0.53)	**0.0537**	0.5165	0.4628
graphsage_300	85.27(±0.62)	85.27(±0.62)	0.0565	0.5421	0.4856
node2vec_100	86.95(±0.61)	86.95(±0.61)	0.1607	**0.3858**	**0.2251**
node2vec_200	86.96(±0.54)	86.96(±0.54)	0.1354	0.3874	0.2520
node2vec_300	86.78(±0.30)	86.78(±0.30)	0.1281	0.4000	0.2719

Table 6
Classification *accuracy* and F_1 score (± standard deviation) of each model on the dataset concerning all the available (300) assignees (Dataset 4). * indicates statistical significance in accuracy improvement against the *bow_100 × 50* baseline. All the models have statistically significant loss (last column) improvement against the *bow_100 × 50* baseline. Bold font indicates the best score value for each column.

Method	Accuracy	F_1	Train loss	Validation loss	Abs loss difference
bow_100 × 50	50.59(±0.38)	50.59(±0.38)	1.4043	2.5318	1.1275
graphsage_100	51.00(±0.53)	51.00(±0.53)	1.3063	2.0776	0.7714
graphsage_200	50.91(±0.30) *	50.91(±0.30)	**1.2925**	2.0836	0.7912
graphsage_300	50.91(±0.36) *	50.91(±0.36)	1.3114	2.0940	0.7826
node2vec_100	**51.64**(**±0.27**) *	**51.64**(**±0.27**)	1.5557	**2.0287**	**0.4729**
node2vec_200	51.32(±0.29) *	51.32(±0.29)	1.5184	2.0350	0.5166
node2vec_300	51.25(±0.60) *	51.25(±0.60)	1.5131	2.0429	0.5298

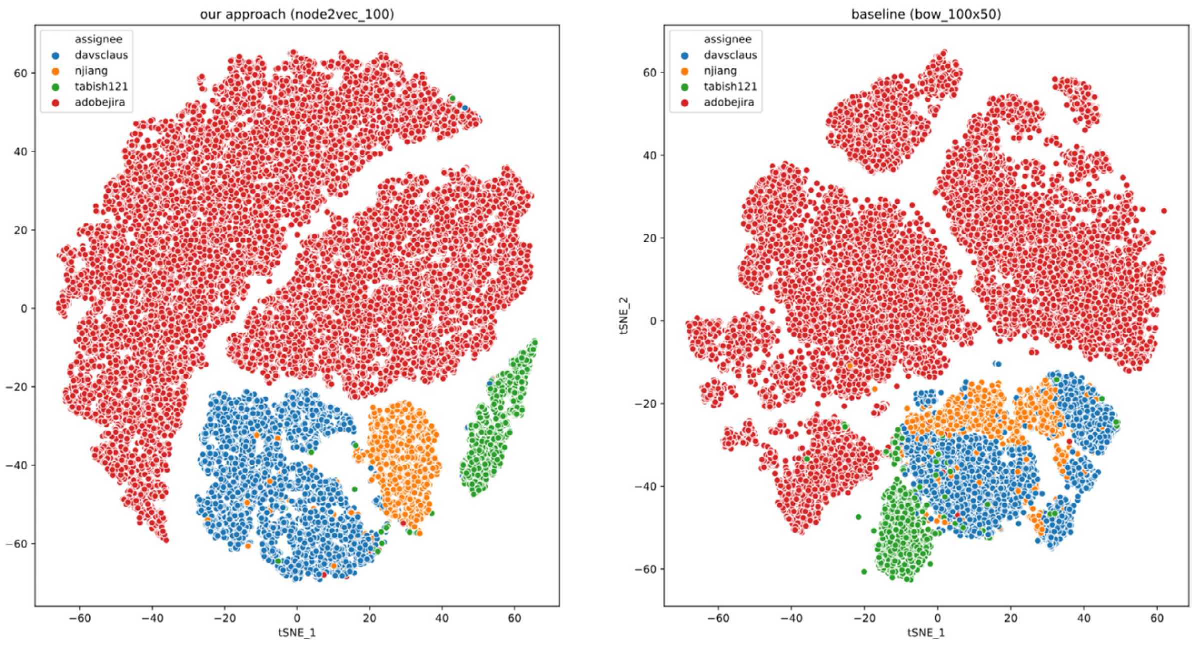

Fig. 3. The t-SNE visualization of the embeddings learned by the proposed approach (**left**) and the baseline model (**right**) on *Dataset 4*.

turn denotes that it is able to calculate the relevance score precisely. It is also noted that the performance of our approach is not affected by the majority of the characteristics of the dataset. On the contrary, it is only affected by the size and the number of the unique classes of the given dataset, something that occurs as a result of the imbalance of the dataset and the limited number of samples for specific document classes.

The examples given in Section 4.4 demonstrate that the proposed approach produces classification models for predicting the best fit for each employee. From a practical perspective, our approach enables project managers to assign employees to tasks without any human bias and alleviates the need for calculating KPIs or other time-consuming metrics.

The main contributions of our paper are: (i) we implemented and evaluated an approach that meaningfully integrates concepts and techniques from the fields of word embeddings, neural networks and graph mining; (ii) we investigated whether the analysis of textual data is able to assist in the personnel selection process; (iii) we proposed a novel ML-based pipeline that assists project managers in the personnel selection process; (iv) we provided the research community with a rich dataset containing tasks of the projects of an organization that can be utilized as a baseline in similar research directions.

A list of remarks has been collected during the analysis of the evaluation results, which can be summarized as follows: (i) the utilized technologies (i.e. word embeddings, neural networks and graph representation learning) can be used for efficiently analyzing textual data related to personnel selection; (ii) the combination of the aforementioned technologies advances the document classification process and assists in producing well-generalized models; (iii) the utilization of graph-based text representations is essential for capturing relationships between the words of a document; (iv) graph representation learning techniques produce better word embeddings as far as the task of identifying synonym words is concerned; (v) neural networks are of great value when dealing with the analysis of high-dimensional textual data.

Aiming to further advance the performance of our approach, future work will focus on improving its classification phase through graph mining techniques such as node classification with graph neural networks and graph attention networks (Veličković et al., 2018). We also plan to further enrich the initial graph-of-docs representation by introducing new edge types among '*issue*' nodes (e.g. '*dependencies*', '*priorities*' etc.). Another future work direction is to expand the proposed pipeline with an additional phase that utilizes techniques from the field of operations research; this will allow the inclusion of constraints and optimization rules, enabling an organization to optimize the allocation of tasks to the available employees (Gaspars-Wieloch, 2021; Kanakaris, Karacapilidis, & Kournetas, 2020).

CRediT authorship contribution statement

Nikos Kanakaris: Conceptualization, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Project administration, Methodology, Validation. **Nikolaos Giarelis:** Conceptualization, Software, Data curation, Writing – review & editing. **Ilias Siachos:** Conceptualization, Software, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Validation. **Nikos Karacapilidis:** Conceptualization, Writing – review & editing, Supervision.

Acknowledgments

The authors would like to thank Haroula Baliaka, Ph.D. candidate at the California Institute of Technology, for her assistance in proofreading and improving the readability of this article.

The work presented in this paper is supported by the inPOINT project (https://inpoint-project.eu), which is co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (Project id: T2EDK- 04389).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL https://www.tensorflow.org/ software available from tensorflow.org.

Abduljabbar, R., Dia, H., Liyanage, S., & Bagloee, S. A. (2019). Applications of artificial intelligence in transport: An overview. *Sustainability*, *11*, http://dx.doi.org/10.3390/su11010189, URL https://www.mdpi.com/2071-1050/11/1/189.

Alkhraisat, H. (2016). Issue tracking system based on ontology and semantic similarity computation. *International Journal of Advanced Computer Science and Applications Word Embeddings: A Survey*, CoRR abs/1901.09069. URL arXiv:1901.09069.

Almeida, F., & Xexéo, G. (2019). Word embeddings: A survey. CoRR abs/1901.09069 arXiv:1901.09069.

Altan, A., & Karasu, S. (2019). The effect of kernel values in support vector machine to forecasting performance of financial time series and cognitive decision making. A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer. *Applied Soft Computing*, *100*, Article 106996. http://dx.doi.org/10.1016/j.asoc.2020.106996, URL https://www.sciencedirect.com/science/article/pii/S1568494620309352.

Altan, A., Karasu, S., & Zio, E. (2021). A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer. *Applied Soft Computing*, *100*, 106996. http://dx.doi.org/10.1016/j.asoc.2020.106996, URL https://www.sciencedirect.com/science/article/pii/S1568494620309352.

Andreas, J., & Klein, D. (2014). How much do word embeddings encode about syntax? In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 2: short papers)* (pp. 822–827). Baltimore, Maryland: Association for Computational Linguistics, http://dx.doi.org/10.3115/v1/P14-2133, URL https://www.aclweb.org/anthology/P14-2133.

Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 International conference on computing networking and informatics (ICCNI)* (pp. 1–9). http://dx.doi.org/10.1109/ICCNI.2017.8123782.

Azzini, A., Galimberti, A., Marrara, S., & Ratti, E. (2018). A classifier to identify soft skills in a researcher textual description. In *EvoApplications*.

Bjorvatn, T., & Wald, A. (2018). Project complexity and team-level absorptive capacity as drivers of project management performance. In *Proceedings of the 28th ACM international conference on information and knowledge management International Journal of Project Management Fast Unfolding of Communities in Large Networks. Journal of Statistical Mechanics: Theory and Experiment Fast and Accurate Network Embeddings Via Very Sparse Random Projection*, 399–408. http://dx.doi.org/10.1145/3357384.3357879.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*, P10008. http://dx.doi.org/10.1088/1742-5468/2008/10/p10008.

Chen, H., Sultan, S. F., Tian, Y., Chen, M., & Skiena, S. (2019). Fast and accurate network embeddings via very sparse random projection. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 399–408). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3357384.3357879.

Dam, H. K., Tran, T., Grundy, J., Ghose, A., & Kamei, Y. (2019). Towards effective ai-powered agile project management. In *2019 IEEE/ACM 41st international conference on software engineering: new ideas and emerging results (ICSE-NIER)* (pp. 41–44). http://dx.doi.org/10.1109/ICSE-NIER.2019.00019.

Fatima, T., Azam, F., Anwar, M. W., & Rasheed, Y. (2020). A systematic review on software project scheduling and task assignment approaches. In *Proceedings of the 2020 6th international conference on computing and artificial intelligence* (pp. 369–373). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3404555.3404588.

Gaspars-Wieloch, H. (2021). The assignment problem in human resource project management under uncertainty. In I. Maglogiannis (Ed.), *Risks an innovative graph-based approach to advance feature selection from multiple textual documents*. Cham: Springer International Publishing.

Giarelis, N., Kanakaris, N., & Karacapilidis, N. (2020a). An innovative graph-based approach to advance feature selection from multiple textual documents. In I. Maglogiannis, L. Iliadis, & E. Pimenidis (Eds.), *Artificial intelligence applications and innovations* (pp. 96–106). Cham: Springer International Publishing.

Giarelis, N., Kanakaris, N., & Karacapilidis, N. (2020b). *On a novel representation of multiple textual documents in a single graph*. In Singapore: Springer Singapore.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864).

Haidabrus, B., Grabis, J., & Protsenko, S. (2021). Agile project management based on data analysis for information management systems. In *Design, simulation, manufacturing: the innovation exchange* (pp. 174–182).

Hamilton, W. L., Ying, R., & Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 1025–1035). Red Hook, NY, USA: Curran Associates Inc..

Hamilton, W. L., Ying, R., & Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. CoRR abs/1709.05584. URL arXiv:1709.05584.

Hassanien, H. E.-D., & Elragal, A. (2021). Deep learning for enterprise systems implementation lifecycle challenges: Research directions. Informatics A comparison of architectures and pretraining methods for contextualized multilingual word embeddings. In *Proceedings of the AAAI conference on artificial intelligence automatic assignment of work items*. In Berlin, Heidelberg: Springer Berlin Heidelberg.

Helming, J., Arndt, H., Hodaie, Z., Koegel, M., & Narayan, N. (2011). Automatic assignment of work items. In L. A. Maciaszek, & P. Loucopoulos (Eds.), *Evaluation of novel approaches to software engineering* (pp. 236–250). Berlin, Heidelberg: Springer Berlin Heidelberg.

Heyn, V., & Paschke, A. (2013). Semantic jira - semantic expert finder in the bug tracking tool jira. CoRR abs/1312.5150. URL arXiv:1312.5150.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, *37*, 547–579.

Jonsson, L., Borg, M., Broman, D., Sandahl, K., Eldh, S., & Runeson, P. (2016). Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts. *Empirical Software Engineering*, *21*, 1533–1578. http://dx.doi.org/10.1007/s10664-015-9401-9.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). Fasttext.zip: Compressing text classification models. CoRR abs/1612.03651. URL http://arxiv.org/abs/1612.03651. arXiv:1612.03651.

Kanakaris, N., Giarelis, N., Siachos, I., & Karacapilidis, N. (2021). Shall i work with them? a knowledge graph-based approach for predicting future research collaborations. Entropy on the exploitation of textual descriptions for a better-informed task assignment process. In *Proceedings of the 9th international conference on operations research and enterprise systems - ICORES* (pp. 304–310). INSTICC SciTePress, http://dx.doi.org/10.5220/0009151603040310.

Kanakaris, N., Karacapilidis, N., & Kournetas, G. (2020). On the exploitation of textual descriptions for a better-informed task assignment process. In *Proceedings of the 9th international conference on operations research and enterprise systems - ICORES* (pp. 304–310). INSTICC, SciTePress, http://dx.doi.org/10.5220/0009151603040310.

Karasu, S., Altan, A., Bekiros, S., & Ahmad, W. (2020). A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series. Energy The benefits of word embeddings features for active learning in clinical information extraction. In *Proceedings of the australasian language technology association workshop 2016* (pp. 25–34). Melbourne, Australia: URL https://www.aclweb.org/anthology/U16-1003.

Kholghi, M., De Vine, L., Sitbon, L., Zuccon, G., & Nguyen, A. (2016). The benefits of word embeddings features for active learning in clinical information extraction. In *Proceedings of the australasian language technology association workshop 2016* (pp. 25–34). Melbourne, Australia: URL https://www.aclweb.org/anthology/U16-1003.

Kusner, M. J., Sun, Y., Kolkin, N. I., & Weinberger, K. Q. (2015). From word embeddings to document distances. In *Proceedings of the 32nd international conference on international conference on machine learning - Volume 37* (pp. 957–966). JMLR.org.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st international conference on international conference on machine learning - Volume 32*. JMLR.org, II–1188–II–1196.

Levorato, V., & Petermann, C. (2011). Detection of communities in directed networks based on strongly p-connected components. In *2011 international conference on computational aspects of social networks (CASoN)* (pp. 211–216). http://dx.doi.org/10.1109/CASON.2011.6085946.

Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. Journal of Machine Learning Research Software project management using machine learning technique—a review. *Applied Sciences*, *11*, 5183.

Mahdi, M. N., Mohamed Zabil, M. H., Ahmad, A. R., Ismail, R., Yusoff, Y., Cheng, L. K., et al. (2021). Software project management using machine learning technique—a review. *Applied Sciences*, *11*, 5183.

McDonald, D. W., & Ackerman, M. S. (2000). Expertise recommender: A flexible recommendation system and architecture. In *Proceedings of the 2000 ACM conference on computer supported cooperative work* (pp. 231–240). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/358916.358994.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. CoRR abs/1301.3781. URL http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781.

Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013b). Efficient estimation of word representations in vector space. URL http://arxiv.org/abs/1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th international conference on neural information processing systems - volume 2* (pp. 3111–3119). Red Hook, NY, USA: Curran Associates Inc..

Mo, Y., Zhao, D., Du, J., Syal, M., Aziz, A., & Li, H. (2020). Automated staff assignment for building maintenance using natural language processing. *Automation in Construction*, *113*, Article 103150. http://dx.doi.org/10.1016/j.autcon.2020.103150, URL https://www.sciencedirect.com/science/article/pii/S0926580519314529.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics, http://dx.doi.org/10.3115/V1/D14-1162, URL https://Www.Aclweb.Org/Anthology/D14-1162.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2623330.2623732.

Powell, M., Rotz, J. A., & O'Malley, K. D. (2020). How machine learning is improving us navy customer support. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 13188–13195). volume 34.

Rousseau, F., Kiagias, E., & Vazirgiannis, M. (2015). Text categorization as a graph classification problem. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: long papers)* (pp. 1702–1712). Beijing, China: Association for Computational Linguistics, http://dx.doi.org/10.3115/v1/P15-1164, URL https://www.aclweb.org/anthology/P15-1164.

Rousseau, F., & Vazirgiannis, M. (2013). Graph-of-word and tw-idf: New approach to ad hoc ir. In *Proceedings of the 22nd ACM international conference on information and knowledge management* (pp. 59–68). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2505515.2505671.

Sajedi-Badashian, A., & Stroulia, E. (2020). Guidelines for evaluating bug-assignment research. *Journal of Software: Evolution and Process*, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.2250.E2250smr.2250.

Sezer, A., & Altan, A. (2021). Detection of solder paste defects with an optimization-based deep learning model using image processing techniques. *Soldering & Surface Mount Technology*.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077). Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, http://dx.doi.org/10.1145/2736277.2741093.

Tran, H. M., Le, S. T., Nguyen, S. V., & Ho, P. T. (2019). An analysis of software bug reports using machine learning techniques. *SN Computer Science*, *1*, 4. http://dx.doi.org/10.1007/s42979-019-0004-1.

Truica, C., & Barnoschi, A. (2019). Innovating HR using an expert system for recruiting IT specialists - ESRIT. CoRR abs/1906.04915. URL arXiv:1906.04915.

van der Heijden, N., Abnar, S., & Shutova, E. (2020). A comparison of architectures and pretraining methods for contextualized multilingual word embeddings. In *Proceedings of the AAAI conference on artificial intelligence, vol. 34* (pp. 9090–9097). http://dx.doi.org/10.1609/aaai.v34i05.6443, URL https://ojs.aaai.org/index.php/AAAI/article/view/6443.

Vanneschi, L., Horn, D. M., Castelli, M., & Popovič, A. (2018). An artificial intelligence system for predicting customer default in e-commerce. *Expert Systems with Applications*, *104*, 1–21. http://dx.doi.org/10.1016/j.eswa.2018.03.025, URL https://www.sciencedirect.com/science/article/pii/S0957417418301702.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. arXiv:1710.10903.

West, D. B., et al. (2001). *Introduction to graph theory, vol. 2*. Prentice hall Upper Saddle River.

Wowczko, I. A. (2015). Skills and vacancy analysis with data mining techniques. Informatics From word embeddings to document similarities for improved information retrieval in software engineering. In *Proceedings of the 38th international conference on software engineering* (pp. 404–415). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2884781.2884862.

Ye, X., Shen, H., Ma, X., Bunescu, R., & Liu, C. (2016). From word embeddings to document similarities for improved information retrieval in software engineering. In *Proceedings of the 38th international conference on software engineering* (pp. 404–415). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2884781.2884862.

Zhu, X., & Ghahramani, Z. (2002). *Learning from labeled and unlabeled data with label propagation*: *Technical report*.