Modeling a Competence-based Industry Role Selection System for University Graduates Using Machine Learning

Fullgence Mwachoo Mwakondo

Institute of Computing and Informatics, Technical University of Mombasa, P.O. Box 90420 – 80100, Mombasa, Kenya

Email: mwakondopoly@gmail.com

Abstract

This paper presents a design of a system for industry role selection, representing both its structure and behavior. Knowing the right industry role that suits a graduate based on their competences on graduation has remained a critical matter for graduates when searching for jobs after graduation. Thousands of university students graduate each year and enter the market to search for jobs that are limited. Searching without prior information on the most appropriate industry role one is suitable for leads to blind search. Blind search not only puts graduates at risk of long-term unemployment and job mismatch but also overloads employers with many applications during job selection. Therefore, this paper addresses 2 objectives: 1) to model the system's structure, and 2) to design the algorithm for the system's behavior. Since object-oriented programming is currently the dominant programming paradigm, object modeling technique was selected to model both the system's structure and the algorithm for the system's behavior. To realize object modeling and represent the system's artifacts in a highly simplified form, Unified Modeling Language (UML) was adopted as the standard modeling toolkit. More specifically, UML class diagram was used to represent the structural model of the system where the underlying objects of the model were exactly similar to those of the problem domain. Finally, use case diagram of the UML toolkit was used to represent the system's behavior in selecting industry role for graduates. To ensure that the system improves performance of its behavior through experience in selecting industry roles for graduates, Machine Learning (ML) algorithm was designed. Two machine learning techniques, naïve Bayes and Support Vector Machines (SVM), were used as the algorithm's criteria for selecting industry roles for graduates. Experiments to evaluate performance of the system were conducted using data collected from Software Engineering industry domain. The end product was design of an intelligent industry role selection system with relevant structure and behavior to easily work with both in the academia and industry. Findings reveal the system improves performance of its behavior in selecting industry roles for graduates much better under SVM (67%) than naïve Bayes (57%). On the same benchmark dataset, the system recorded better performance (85%) than reported performance (82%) in the benchmark system. These findings will benefit industry by getting evaluation tool for revealing graduate's suitability for employment which they can use as prior information for decision making when filtering candidates for interview. Besides, this will provide researchers with a digital platform to study and bridge the gap between industry and academia. Lastly, this will attempt to reduce both low job satisfaction and long-term unemployment that is one of the causes of social and economic pain both in Kenya and around the world. This paper has revealed competence based industry role selection system with relevant structure and behavior can improve searching of jobs by providing a fairly accurate prior information. However, this paper recommends testing this approach with other alternative machine learning techniques as well as other alternative industry domains.

Key Words: Industry-academia gap, Machine learning, Object-oriented modeling, Selecting industry roles

Introduction

Thousands of university students graduating each year enter the market to search for jobs that are limited. This is due to inadequate capacity of most economies to create jobs both in developed and developing countries in the world. As a result, for new university graduates getting a job is characterized by long search for employment opportunities in the market. In order to maximize their employment chances, graduates send as many applications as possible to as many organizations they think are potential employers. However, this

Published: December 2020

method of searching without prior information on the most appropriate industry role one is suitable for leads to blind search. Blind search not only puts graduates at risk of long term unemployment and job mismatch but also overloads employers with many applications during job selection. Therefore, knowing the right industry role that suits a graduate based on their competences on graduation remains a critical matter for graduates when searching for jobs after graduation.

As a result, matching competences a graduate possesses with competences required by a given industry role through skills mapping is the new technique that links graduates skills with industry roles. Skills mapping ensures the right match of graduates' skills to industry jobs through application of analytical methods. Analytical methods do better where independent and dependent features of a problem are simple and linearly related to each other. However, where complexity and non-linearity characterize the problem as it were in skills mapping, analytical methods get overwhelmed. Although industry roles are defined by unique patterns of skills in terms of main competence, specific competence, and proficiency (CWA16458) their complexity is evidenced by the way they are organized hierarchically into specialized groups defined by organizational structures. Four organizational structures used to organize industry roles are functional, geographical, product, and matrix. As a result, recognizing skills patterns for various industry roles organized in these hierarchical structures is difficult for analytical methods and makes them inefficient.

However, technology such as Artificial Intelligence (AI) can be used to support analytical methods and help improve their efficiency. AI provides a technology for designing programs that can learn such complex patterns from grouped data and be able to automatically recognize these patterns in newly collected data. This involves representing the underlying structure of the groups in the program as the data structure and empowering the program to learn the pattern rules for each group that can be applied to classify a new data item into any one of the existing groups. This approach is known as supervised learning and requires availability of large volumes of data. Coincidentally, due to the availability of large volumes of data, data driven AI technology known as Machine Learning (ML) is gaining traction. ML is used to design such programs that can learn and improve their performance when carrying out a task through experience derived from learning.

Therefore, in this paper, the design of an intelligent system for industry role selection using machine learning as the data-driven, AI technology, representing both its structure and behavior are discussed. Object modeling technique was selected to model both the system's structure and the algorithm for the system's behavior. To ensure that the system improves performance of its behavior through experience in selecting industry roles for graduates, ML was used to design the algorithm. The system is expected to solve the problem of blind search which involves searching for job without prior information on the most appropriate industry role one is suitable for. Blind search not only puts graduates at risk of long term unemployment and job mismatch but also overloads employers with many applications during job selection. As a result, the specific objectives of this paper were: 1) to model the system's structure, and 2) to design the algorithm for the system's behavior.

Situation Analysis

There exists little information towards improving graduates employability especially using machine learning techniques (Jantawan & Tsai, 2011; Chien & Chen, 2008). Zaharim *et al.,* (2010) applied requirements of professional bodies and accrediting bodies to construct an engineering employability skills framework for Malaysian graduates. Chien & Chen, (2008) built a classification system using data mining techniques for prediction of employee retention of new job applicants. Jantawan & Tsai (2011) presented a classification system based on decision trees and naiveBayes for predicting graduate employment 12 months after graduation based on attributes that influence graduate employment identified from actual data collected from graduates. None of the studies attempts to empower new graduates with information for quick job search or attempts to improve the search technique.

Case study: Software Engineering Industry

Software Engineering (SE) is a typical case of occupational industry domain. This domain has been widely studied in research literature (Moreno et al., 2012; Shashidhar et al., 2015). SE as an industry occupation is concerned with development of software that is reliable, efficient and economical. Software developers or engineers refer to the entire community of people involved in software development or working in the SE industry in various roles. Role activities of software development demand specialized cognitive competences as prerequisites for superior performance (Winterton et al., 2005). Industry roles have specific areas of competence as aspects of the job which an individual can perform competently. Each industry role demands certain levels of skills capacity in terms of main competence area, specific competence area, and proficiency. Currently, industry roles in SE demand 3 types of main competence (designing, coding, managing) and 6 specific competences include, analyst programmer (AP – application programmers, MP – mobile programming), test programmers (TE – test engineers), software architecture designer (SA – software architect), web designers (WP – web programmers), systems managers (SAD - systems administrator) and project managers (PM – project management), and broadly focused to either mobile applications or desktop applications. Table 1 presents areas of competence for software developers and their relative demand or prevalence in each industry role as well as percentage proportions in each industry role of software developers' focus towards mobile applications or desktop applications. Competences that drive superior job performance are derived from knowledge and skills.

Table 1. Areas of Competences for Software Engineering (SA, software architect; AP, application programming; TE, test engineering; WP, web programming; MP, mobile programming; SAD, systems administration; and PM, project management) (Mwakondo, 2018)

	TYPE	SE Industry Roles							TOTAL [Rank]
		SA	AP	TE	WP	MP	SAD	PM	
Main Competence [% prevalence]	D (design)	50.00	22.12	42.16	40.51	34.58	17.61	28.21	36.00 [2]
	P (coding)	33.20	61.06	52.61	40.51	42.36	29.55	34.29	42.72 [1]
	M (manage)	16.80	16.81	5.22	18.99	23.05	52.84	37.50	21.28 [3]
Specific competence [% numbers]	Mobile Developers	26.30	48.30	57.10	48.40	0	7.70	14.30	38.90
	Desktop Developers	73.70	51.70	42.90	51.60	0	92.30	85.70	61.10

Technical skills required of software developers were revealed by a study carried out by Surakka (2005) which grouped these skills into 5 categories: platform skills, programming skills, networking skills, database skills and distributed technology skills. To learn the skills for software development, graduates must be trained in the academia. However, they are trained along with other ICT practitioners through a number of degree programs that are offered in the academia such as Computer Science, Information Technology, Software Engineering, Mathematics and Computer Science. This makes it difficult to recognize skills patterns that are unique for various SE industry roles. The universally recommended source of knowledge and skills for software engineers is known as Software Engineering Body of Knowledge (SWEBOK). This content provides a standard to academia for creating academic programs. As a result, the academic programs in which graduates are trained are created in collaboration of both employers and academia in the industry. Fig. 1 indicates the interaction between employers, institutions, and graduates as stakeholders in the domain industry.

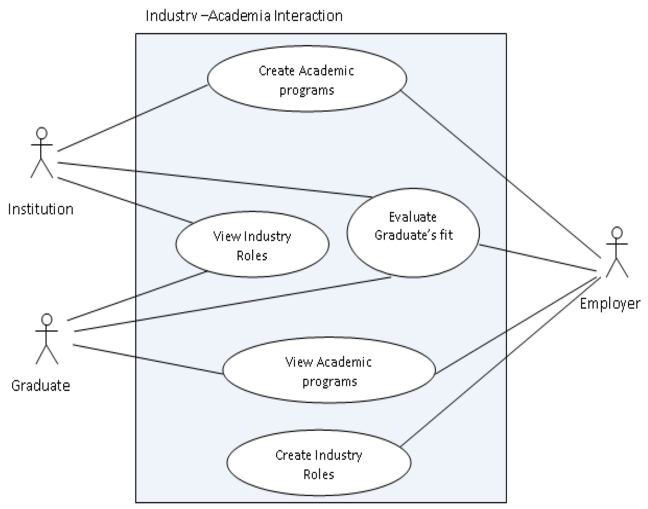

Figure 1. Graphic presentation of industry-academia interactions

Each year institutions advertise programs that are viewed and attract potential graduates. Before they are enrolled for training, graduates are evaluated to check their readiness to undertake the program. Each institution in the academia has its own eligibility criteria to admit the students for the program. This makes graduates' skills from one institution biased towards some of industry roles possibly different from another institution. Therefore, employers must interact with the programs in academia through which they can select ones that whose graduates' skills are closely biased towards their needs and can be potential employees. Successfully trained graduates are evaluated by not only learning institutions to determine their level of success but also employers in the industry to determine their suitability for industry roles before they are employed.

Structural Modeling of the Industry Role Selection System

Unified Modeling Language (UML) is the standard modeling toolkit used in object-oriented modeling. Object modeling technique was preferred over structured modeling to model the system's structure because currently the dominant programming paradigm is object-oriented programming (Tharawani et al., 2016). Object-oriented programming languages, such as Java, .Net, PHP, Python, etc., are widely used to implement the object–oriented design models. More specifically, class diagram of the UML toolkit was used to produce the object model to represent the system's structure in selecting industry roles for graduates. An object model describes the system's objects and their underlying interactions. Therefore class diagram as a representation of object model provides specification for software classes and their interfaces in the application. Fig. 2 presents the object model in the form of a class diagram. The diagram indicates that graduates are trained by institutions whose training targets specific industry sectors. Each sector contains specific roles that are offered by employers. Therefore graduates targeting a particular sector are employed by employers belonging to that sector.

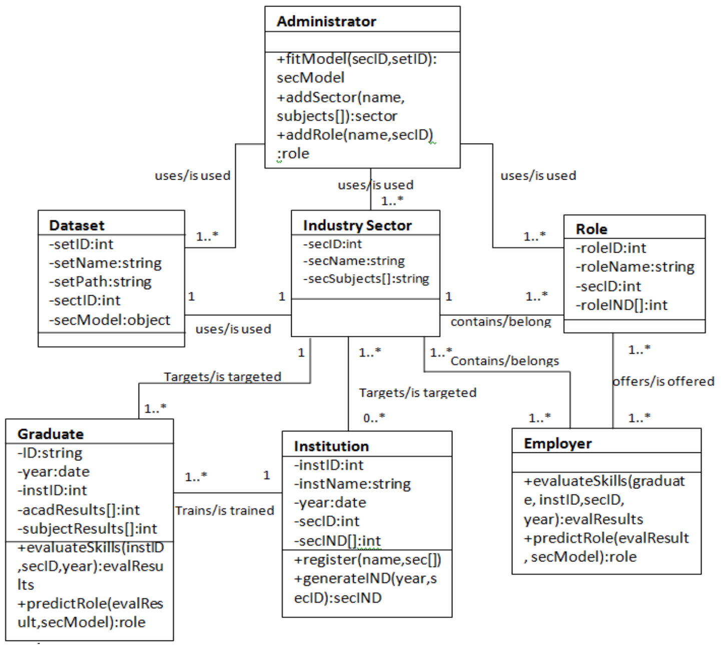

Figure 2. Graphic presentation of a class model

In addition, there is a lot of data available in each sector showing employees' profiles which can be mined to extract rules for prediction models that can be used to evaluate and predict most suitable roles for new graduates by potential employers or the new graduates themselves. Class *Role* models various jobs offered by the industry employers. Each role has a specific set of competences which are conceptualized into values stored as an array field of the class known as *roleIND*. Since each role belongs to a particular named sector characterized by a particular set of subjects, the class *industry sector* represents these details for each role. Each role has a particular threshold for sector subjects whose training is conducted at learning institutions. The class *institution* models various sectors in which graduates are trained through various subjects where they develop required competences for roles in the training sector. Cohorts in the training develop both common competences that apply to all and specific competences that are individual based. Common competences are conceptualized into values stored as an array field of the institution class known as *secIND* while the specific competences are calculated on demand from a graduate through the class graduate where academic results and scores in the sector subjects are derived. The class Administrator models an artificial entity that produces the prediction model through the *fitModel* method that uses the dataset for the sector in the *Dataset* class. The *fitModel* generates the machine learning model from the dataset that is used by both *Graduate* and *Employer* classes to predict suitable industry role through the *predict* method. The *predict* method is called repeatedly by *evaluateSkills* methods of *Graduate* and *Employer* class.

Behavioral Modeling of Industry Role Selection System

Again, Unified Modeling Language (UML) was adopted as the standard modeling toolkit. More specifically, use case diagram of the UML toolkit was applied to produce the use case model to represent the system's behavior in selecting industry role for graduates. A use case model describes the functions of the system as viewed by its users, developers, and testers, and is developed as the initial specification of the system's requirements. Fig. 3 presents the behavioral model in the form of a use case diagram. The use case model envisaged 4 kind of users for the system i.e. administrator, employer, graduate, and university institution. Employers should be able to register industry roles available in various sectors in which they operate, clearly indicating their minimum skills and knowledge index values requirements. Also, they should be able to view academic sector profiles for various institutions based on their knowledge and skills content in the exams each year they examine. Finally, employers should be able to evaluate new graduates on industry roles suitability. Likewise, institutions should be able to register their academic profiles for sectors in which their degree programs are based. Where for each sector, each year they should record knowledge and skills indices derived from their exam's content administered to students. Also, they should be able to view industry roles profiles for various sectors based on knowledge and skills minimum indices required by industry. Finally, institutions should be able to evaluate their graduates on industry roles suitability before they graduate so as to assess themselves against industry requirements. Graduates, as well should be able to evaluate themselves against industry roles requirements to determine their suitability for employment. They should, also, be able to view industry role requirements for various sectors in industry as well as view academic performance profiles in various sectors for various institutions.

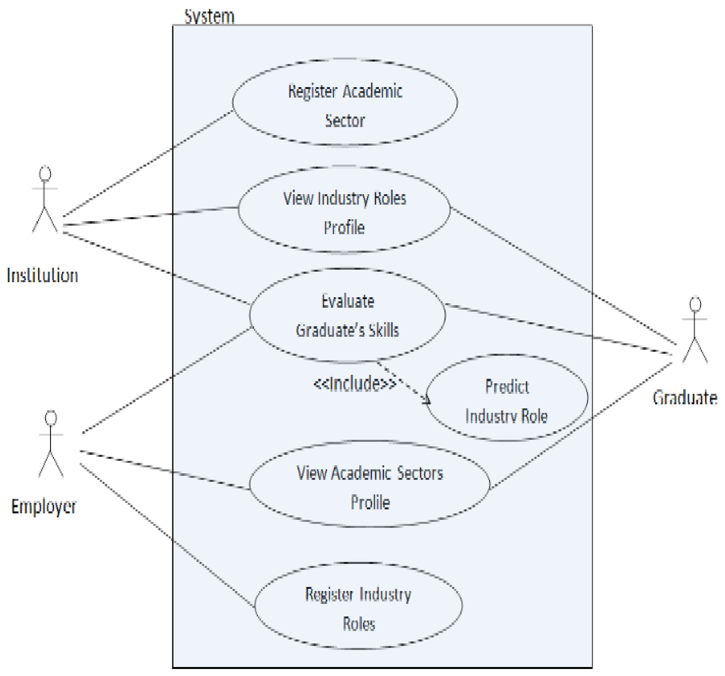

Figure 3. Graphic presentation of use-case model

Algorithm Design and Implementation of the Industry Role Selection System

In this section, the algorithm that generates the machine learning objects from the dataset of employees' profiles in the industry roles as well as algorithm that uses these machine learning objects to predict industry roles for new graduates is described. Also, the results achieved through the proposed choice selection system are presented and discussed.

Fit Algorithm explanation

This method is responsible for fitting the data into the system to learn or to estimate the parameters. Algorithm 1 outlines algorithm of the 'fit' method. This algorithm takes in the taxonomic tree in which the industry roles are organized and the dataset containing graduate employees details to be learned. The algorithm is able to group the dataset content based on their dependent values according to the various sections of the taxonomic tree such as sub-tree (main competence), non-leaf nodes (proficiency), leaf nodes (specific competence), or tree heights. The algorithm is able to learn how items of the dataset belonging to various leaf nodes look like, if they belong to known non-leaf nodes and various non-leaf nodes are distinguished by their height levels in the tree or sub-trees. Finally, the algorithm is able to store the learned knowledge rules for that particular dataset. Therefore, the key aspects of this algorithm are: 1) input, 2) learning, and 3) storing the learned knowledge rules. Input to the function is the dataset, *d*, and taxonomy tree, *t*, which is a hierarchical data structure describing industry roles structure. MLA stands for Machine Learning Algorithm, in this case either naïve Bayes or support vector machines.

Algorithm1: fit model algorithm

```
Function Fit(taxonomy_tree t, dataset d)
    PredictorModel = {}
    levelPredictionObject = {}
    NodePredictionObject = {}
    subTreePredictionObject = {}
    1_Get taxonomy_tree's height/levels
    Height =t.getTreeDepth()
    1_Get subtrees/main_competences
    Subtrees = {}
    Subtrees = t.getSubtrees()
    1_For each subtree/main_competences
    For Each subtree, m in subtrees
        1.1_Get subtree's leaf nodes/classes
        Childnodes = m.getChildrenOf(d)
        1.2_Get other subtrees' leaf nodes/classes
        Othernodes = t.getChildrenOf(d) - Childnodes
        1,3_Create subtree's (main_competences) classifier object
        Testset1, trainset1 = splitDataset(d, childnodes)
        Testset2, trainset2 = splitDataset(d, othernodes)
        Testset = testset1 + testset2
        Trainset = trainset1 + trainset2
        PredictionModel = m.getPrediction(MLA, trainset)
        ModelAccuracy = m.getAccuracy(predictionModel, testset)
        subTreePredictionObject[m] = {[childnodes], [predictionModel],[ModelAccuracy]}
        3_For each subtree's non-leaf nodes/proficiencies
        For Each subtree levels, h in Height
            3.1_Get leaf children
            Leafnodes = m.getLevelNodes(h, d)
            3.2_Get other non-leaf nodes' leaf children
            otherLeafnodes = m.getChildrenOf(d) - leafnodes
            3,3_Create non-leaf node's (proficiency) classifier object
            Testset1, trainset1 = splitDataset(d, leafnodes)
            Testset2, trainset2 = splitDataset(d, otherLeafnodes)
            Testset = testset1 + testset2
            Trainset = trainset1 + trainset2
            PredictionModel = m.getPrediction(MLA, trainset)
            ModelAccuracy = m.getAccuracy(predictionModel, testset)
            levelPredictionObject[h] = {[leafnodes], [predictionModel],[ModelAccuracy]}
            4_For each subtree's leaf nodes/ specific_comptences
            For Each subtree leafnode, n in leafnodes
                4.1_Get leaf node/class
                Currentnode = n
                4.2_Get siblings
                Othernodes = leafnodes - n
                4.3_Create leaf node's (Specific_comptence) classifier object
                Testset1, trainset1 = splitDataset(d, currentnode)
                Testset2, trainset2 = splitDataset(d, otherLeafnodes)
                Testset = testset1 + testset2
                Trainset = trainset1 + trainset2
                PredictionModel = m.getPrediction(MLA, trainset)
                ModelAccuracy = m.getAccuracy(predictionModel, testset)

            NodePredictionObject[n] = {[currentnode], [predictionModel],[ModelAccuracy]}
            End For
            3_Store specific classifier objects in the data structure
            PredictorModel[1] ={ NodePredictorObject}
        End For
        2_Store proficiency classifier objects in the data structure
        PredictorModel[2] ={ levelPredictorObject}
    End for
    1_Store main competence classifier objects in the data structure
    PredictorModel[3] ={subtreePredictorObject}
PredictorModel{ 1: {NodePredictorObject}, 2: {levelPredictorObject}, 3: {subtreePredictorObject}}
return PredictorModel
End function
```

Predict Algorithm explanation

This method is responsible for the prediction function of the system. Algorithm1 outlines the algorithm of the 'predict' method. The algorithm takes in an instance of unemployed graduate's data and taxonomic tree for industry roles in which the graduate is seeking for employment. The algorithm uses the knowledge rules generated by the 'fit' algorithm to decide the role for which the graduate is suitable. The key aspects for this algorithm are: 1) input tree and graduate data, 2) load the knowledge rules from the store, 3) search for the appropriate knowledge rules to process the graduate data and, 4) use the rules to decide the industry role suitable for the graduate.

Algorithm2: predict role algorithm

```
Function Predict(taxonomy_tree t, data d)
    mainCompetencePredictorObjects = {}
    proficiencyCompetencePredictorObjects = {}
    specificCompetencePredictorObjects = {}
    1_Load classifier objects
    File = open("path to PredictorModel")
    Model = Load(File)
    1_Get taxonomy_tree's height
    Height = t.getTreeDepth()
    1_Get taxonomy_tree's subtrees/main_competences
    Subtrees = {}
    Subtrees = t.getSubtrees()
    1_For each subtree/main_competence
    For Each subtree, m in subtrees
        2.1_Get main_competence classifier objects
        mainCompetencePredictorObjects = Model[3]
        currentAccuracy = 0
        correctlabels = {}
        2.2_Predict data's main_competence
        For Each mainCompetencePredictorObject, MO in mainCompetencePredictorObjects
            Labels = MO[0]
            Predictor = MO[1]
            Accuracy = MO[2]
            Result = Predictor(d)
            3.1_Select main_competence of classifier object that predicts +ve
            If Result ==1
                If (currentAccuracy<=Accuracy)
                    currentAccurracy = Accuracy
                    correctLabels = Labels
                    correctSubtree = m
                else
                    skip/continue
            else
                skip/continue
        End For
    End For
```

```
1_For each subtree's non-leaf nodes/proficiencies
        ordered in ascending order of levels
For Each correctSubtree levels, h in Height
        2.1_Get corresponding proficiency
        classifier objects
        proficiencyCompetencePredictorObj
        ects = Model[2]
        currentAccuracy = 0
        correctlabels = {}
        2.1_Predict data's proficiency competence
        For Each
        proficiencyCompetencePredictorObj
        ect, PO in
        proficiencyCompetencePredictorObj
        ects
                Labels = PO[0]
                Predictor = PO[1]
                Accuracy = PO[2]
                Result = Predictor(d)
                3.1_Select proficiency of
        classifier object that predicts +ve
                If Result ==1
                        If
        (currentAccuracy<=Accuracy)

            currentAccurracy = Accuracy

            correctLabels = Labels

            correctLevelNodes = h
                        else

            skip/continue
                else
                        skip/continue
        End For
End For
1_Get current leaf node's specific_competence
        classifier objects
For Each correctLevelNode, n in
        correctLevelNodes
        2.1_ Get specific_competence classifier
        object
        specificCompetencePredictorObjects
        = Model[1]
        currentAccuracy = 0
        correctlabel = {}
        2.1_Predict data's specific_competence
        For Each
        specificCompetencePredictorObject,
        SO in
        specificCompetencePredictorObjects
                Label = SO[0]
                Predictor = SO[1]
                Accuracy = SO[2]
                Result = Predictor(d)
                3.1_Select
        specific_competence of classifier object
        that predicts +ve
                If Result ==1
                        If
        (currentAccuracy<=Accuracy)

            currentAccurracy = Accuracy

            correctLabel = Label
                        else

            skip/continue
                else
                        skip/continue
        End For
    End For
    1_Report industry role = main_competences +
        proficiency + specific_competence
    Role=maincompetence(correctSubtree)+pr
        oficiency(correctLevelNodes)+specif
        ic_competence(correctLabel)
    return Role
End Function
```

Results and Discussion

Table 2 presents the demographic characteristic of the experimental datasets used in the investigation. Dataset2 was used as benchmark dataset where Shashidhar *et al.*, (2015) on the same dataset, using a related system, reported performance accuracy of 82%. Dataset1&3 were used as multiple case studies for different industry domains to validate system's results for generalizability. The system's model was generated using two induction algorithms, hence two versions of the model. The two models were experimented under similar conditions then the results were compared. This involved fitting and testing both models with similar training and validate sets, respectively through 10 iterations of 5-fold cross-validation.

Table 2. Demographic characteristics of experimental datasets

Dataset	Attributes	Instances	Classes	Levels
Dataset1 (SE field)	18	113	12	3
Dataset2 (SE benchmark)	18	279	12	3
Dataset3 (AL field)	14	50	7	3

Table 3 presents results of this experiment that indicated there was a difference in mean performance between SVM and naïve Bayes models based system (56.79, 52.54 in dataset1 and 78.77, 63.93 in dataset2, respectively) which suggested that SVM model was better than naïve Bayes. Further investigation was conducted to test whether the differences (4.25 and 14.84) were real and significant.

This test was conducted using paired sample t-test procedure. A paired sample t-test was conducted to test the hypothesis that system performance difference was not significant. For this type of test to be valid, conditions for tests were checked (homogeneity and normality of data). The results indicate the difference was real and significant based on 10 iterations of 5-fold cross validation tests (paired dataset1: t = -2.09; df = 49; p = 0.042 and paired dataset2: t = -15.02; df = 49; p = 0.000, respectively).

Table 3. Model's mean (± STDev) performance validation

	5-Fold cross validation results (%)			
Test fold	SE Benchmark Dataset		SE Field Dataset	
	Naïve Bayes	SVM	Naïve Bayes	SVM
Fold_1	60.8±3.4 (n=10)	73.3±3.7 (n = 10)	49.5±7.5 (n=10)	55.9±9.6 (n=10)
Fold_2	63.0±3.7 (n=10)	77.8±4.2 (n=10)	48.9±11.3 (n=10)	59.4±5.0 (n=10)
Fold_3	66.7±5.9 (n=10)	80.2±6.0 (n=10)	56.2±7.2 (n=10)	58.1±10.7 (n=10)
Fold_4	63.4±5.5 (n=10)	81.9±2.6 (n=10)	51.2±10.5 (n=10)	53.1±8.5 (n=10)
Fold_5	65.8±6.2 (n=10)	80.7±5.8 (n=10)	56.8±14.0 (n=10)	57.5±12.7 (n=10)
Overall	63.93±5.30 (N=50)	78.77±5.42 (N=50)	52.54±10.56 (N=50)	56.79±9.48 (N=50)

To confirm the difference was not due to any other factor but only machine learning construct difference, ANOVA test was conducted to rule out the effect of fold to fold variations. One-way ANOVA is a statistical procedure used to test whether the means of 2 or more groups, in this case folds, were significantly different. For this type of test to be valid, conditions for ANOVA that must be satisfied are homogeneity of group variance and normality of data was done using the Levene's test where the condition for homogeineity of group variances and the condition for difference of group means were both met (p = 0.495 and p = 0.135, respectively). The results indicate the fold variances were equal and, in fact, means of the fold scores were not different and therefore the seemingly difference between the two models in Table 3 above was not due to effect of fold to fold variations. This was enough reason to select SVM model as the best classifier for the system.

Finally, test of the quality of the system using appropriate quality measures was required. This was after realization that accuracy alone sometimes could be misleading as sometimes a model with relatively high accuracy was likely to predict the 'not so important class labels' fairly accurately while making all sorts of mistakes on classes that were actually critical. As a result, other performance measures such as precision, recall and F1 scores were incorporated. The aim was to study the ability of the model to find all the positive instances correctly (recall) and ability not to label negative instances as positive (precision) or weighted average score of the two (F1). Table 4 illustrates results of the model performance along 4 quality metrics and across 3 datasets, and presents performance results along hierarchical levels across the 3 datasets. In each case, the model reported equal performance in both accuracy and recall. However, its ability not to label negative classes as positive was not as good as its ability to find all positive classes correctly which was equally good (precision = 66%, recall = 69%).

Table 4. Comparison of performance across three datasets

Performance Metric	SE field Dataset (Research)	SE lit. Dataset (Benchmark)	AL field Dataset (Validation)	Mean
Accuracy	0.59	0.85	0.65	0.69
Precision	0.62	0.83	0.54	0.66
Recall	0.59	0.85	0.65	0.69
F1_score	0.57	0.83	0.56	0.65

On average, model performance seemed to improve upward the hierarchy levels consistent with other models (Clare & King, 2003; Barbedo & Lopes, 2006). Model's performance seemed to be very high in the benchmark dataset as a result of having more instances whose classes had very high accuracies (class 10 & 11) and fewer instances whose classes had very low accuracies (class 7 & 8). This was not the case with other two datasets where distribution differences of classes with very high and very low accuracies were not high. In the Benchmark dataset where performance was 85%, high accuracy (100%) class (class11) had the highest number of instances (size=11) while low accuracy (5%) class (class7) had the lowest number of instances (size=2). In Research dataset where performance was 59%, high accuracy (93.4%) class (class7) had moderate number of instances (size=3) while low accuracy (5%) class (class3) had moderate number of instances (size=1). In Validation dataset where performance was 65%, high accuracy (100%) class (class1&5) had moderate number of instances (size=2) while low accuracy (5%) class (class2&7) had moderate number of instances (size=2). Model performance in both Research and Validation datasets seemed to be fairly good (59% and 65%, respectively). These results indicate the best generalization performance as an average performance calculated across the 3 datasets. In this case, along hierarchical levels the best average performance accuracy of the model was 67% while general average performance was 69%. Therefore, it can confidently be claimed that the best performance of the model was 67%.

Conclusion

This paper has revealed that searching for a job without prior information on the most appropriate job one is suitable for leads to blind search. Blind search can put graduates at risk of long term unemployment, job mismatch and as well as overloading employers with many applications during job selection. As a result, this paper has presented a design of a system for industry role selection, representing both its structure and behavior and more specifically modeling the system's structure using class diagram as well as modeling system's behavior using use case diagram. Two important aspects of the system's behavior when selecting appropriate industry role with the help of machine learning techniques were presented as design algorithms. Experimental design was adopted to test the systems relevancy where the findings were promising. Hence, the paper's findings is a greater step forward towards reducing both low job satisfaction and long term unemployment that is one of the causes of social and economic pain both in Kenya and around the world.

Acknowledgement

My great appreciations go to the entire staff of the Department of Partnership, Research and Innovation (PRI) at Technical University of Mombasa (TUM) for proof reading, editing, and final publication of this study. Besides, National Council for Science, Technology and Innovation (NACOSTI) was very instrumental in supporting the research funding of this study as part of my PhD research.

References

Chien C., Chen L. (2008). Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry. *Expert Systems with Applications* 34: 280–290

Jantawan, B. & Tsai, C. (2013). Application of Data Mining to Build Classification Model for Predicting Graduate Employment". *International Journal of Computer Science and Information Security Vol.11 (10): 1–7*

Moreno, A., Sanchez-Segurab, M., Medina-Dominguezb, F. & Carvajal, L. (2012). Balancing software engineering education and industrial needs. *Journal of Systems and Software 85(7):1607–1620*

Mwakondo, F. (2018). A Model for Mapping Graduates' Skills to Industry Roles using Machine Learning Techniques: A Case of Software Engineering. *PhD Thesis. http://erepository.uonbi.ac.ke/*

Shashidhar, V., Srikant, S., Aggarwal, V. (2015). Learning Models for Personalized Educational Feedback and Job Selection. *Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015*

Surakka, S. (2005). Trend Analysis of Job Advertisements: What Technical Skills Do Software Developers Need? *Informatics in Education, 2005. Vol. 4(1): 101-122*

Tharawani, R.K, Nzamani, H.N, Nzamani, Q.A., Khatari, Y., Chandio, F.H, & Abbasi, M.S. (2016). Modelling Choice Selection System of a Public Sector General University in Pakistan, *Sindh University Research Journal (Science Series)* 48: 127-134

Winterton, J., Delamere, F. & Stringfellow, E. (2005). Typology of Knowledge, Skills and Competences: Clarification of the concept and prototype. *Centre for European Research on Employment and Human Resources 2005:1-108*

Zaharim, A., Omar, M.Z., Yusoff, Y.M., Muhamad, N., Mohamed, A., and Mustapha, R. (2010). Practical framework of employability skills for engineering graduate in Malaysia,” *In: IEEE EDUCON Education Engineering 2010: The Future of Global Learning Engineering Education:* 921–927 pp