sustainability

MDPI

Article

Align My Curriculum: A Framework to Bridge the Gap between Acquired University Curriculum and Required Market Skills

Ahood Almaleh, Muhammad Ahtisham Aslam *, Kawther Saeedi and Naif Radi Aljohani

Faculty of Computing and Information Technology, King Abdulaziz University, 21589 Jeddah, Saudi Arabia; aalmaleh0001@stu.kau.edu.sa (A.A.); ksaeedi@kau.edu.sa (K.S.); nraljohani@kau.edu.sa (N.R.A.)

* Correspondence: maaslam@kau.edu.sa; Tel.: +966-5633-21977

Received: 7 April 2019; Accepted: 30 April 2019; Published: 7 May 2019

Abstract: With the advancement of technology, academics and curriculum developers are always under pressure to provide students with skills that match the market's requirements. A systematic and continuous examination of the market is needed, to stay up to date with the required skills, and then to update the curriculum to train the students with required market skills. In this article, we present a framework referred to as Align My Curriculum (AMC). The AMC framework aims to facilitate alignment between acquired university curriculum outcomes and required market skills. It can be used to classify, compare and visualize the data of a university curriculum and job vacancies in the market. The presented framework benefits academics and curriculum developers by improving the courses and therefore bridging the skills gap. Stakeholders from both academia and industry can gain insights into the predominant required and acquired skills. In addition, it may be useful for analysts, students, and job applicants. This article describes the architecture, implementation and experimental results, with visual analysis to help decision and policy-makers.

Keywords: Naïve Bayes; cosine similarity; text mining; word cloud; classification; comparison; job postings; curriculum

1. Introduction

In recent years, the Internet has changed many aspects of our lifestyle, from how we communicate to how we look for a job. This change has led to a revolution in information and the appearance of quantities of online data, to the extent that finding information became a non-trivial task [1]. With this advancement of technology, academics and curriculum developers are always under pressure to provide students with skills that match the market's requirements. The study of the job market is an area of particular and increasing interest, using innovative data sources and analytical methods [2,3]. This pressure is more relevant in applied areas, such as computing [4]. Staying up to date with the demands of the job market involves retrieving, sifting, and analyzing data of online job notifications [5]. Job 'postings' are viewed as an important source of information for the examination of the required skills, and hold great promise for job market research [6–8]. In the 1990s, job postings on the Internet began to prevail because the costs and time plunged, relative to traditional job advertisements. This provided opportunities to access and analyze job postings to better understand the market trends and demands [6] to bridge the gap between the education offer and market demands [9], and is considered to be a tool to align the education curriculum to the market, preparing students for employment [3]. However, the investigation into the information in online job postings by manual content analysis is a hard task, involving much time and effort due to its dynamic nature, rapid change, and scale of the job market data. To align their curricula (acquired skills) continuously to these requirements,

academics and curriculum developers require a continual assessment of the job market (required skills). There are many types of research into how to analyze the job market using different methodologies, starting by manual analysis through data mining techniques [10,11], and a few researchers have attempted to compare job postings to curricula using manual analysis techniques [4]. In this article, we present a framework referred to as Align My Curriculum (AMC) for use by academics and curriculum developers to bridge the skills gap. To the best of our knowledge, this approach is unique, in the sense that it automates the approach to analyze and compare required and acquired skills.

The remaining article is organized as follows. Section 2 describes related work in the computing disciplines that have used techniques to analyze and examine skill requirements and job market demands. Section 3 describes the architecture of the AMC framework. In Section 4 we describe the experimental results of implementing the framework in the IT domain. Finally, we discuss our findings and conclude this work in Section 5.

2. Related Work

Many studies have been conducted to assess computer-related skills and knowledge using various frameworks and approaches. The study of computer-related requirements started in the 1960s using traditional methods such as surveys, questionnaires, face-to-face meetings, and interviewing experienced figures to define the computing skills required by the market [10,12–14]. In the mid-1990s, researchers developed tools to assess and determine the skills need periodically and to update curricula according to the demands. This process is known as bridging the skills gap [10,15–17]. Due to the importance of analyzing job postings to determine the skills required in the market, to bridge the skills gap, there is a considerable literature on the many techniques in somewhat similar methods. These studies start by collecting job data from online sources [4,5,10,11,18–22], using web crawlers or manually or from newspapers [13]. Then, most of these studies code (label) the jobs into categories, whether by the deductive method [8,13,18], inductive method [4,10,11,19], or both [22]. Finally the analysis of the datasets based on these categories uses manual content analysis [4,13,18] or computerized analysis, such as data mining techniques (clustering, classification, LSA) [5,8,10,11,20] or other techniques [19,22]. Richard and Rachida's (2016) [4] study is slightly different from the above studies, as it examined the online computing job postings and, at the same time, compared them to the Model Curriculum by manual content analysis [4]. As is apparent in the literature, that minimizing dissimilarity between skills needs and graduates is a primary concern of all academics, educators, and many others.

Most previous studies have analyzed job postings to determine the skills using manual analysis steps, in varying proportions. We consider this point to be a limitation and it was our motivation to work on the development of a new approach and framework to improve the identification of the required skills to help academics and curriculum developers to bridge the skills gap between the job market and curricula. Our AMC framework starts by collecting, at the same time, job posting data and curriculum data in the computing field. Then, it uses the Naïve Bayes algorithm to classify the datasets on the basis of predefined classes derived from the literature using the deductive method. After that, it uses cosine similarity techniques to measure the match between the job data and curriculum data. Finally, it uses word cloud techniques to visualize each class of the two datasets. In the next section, we describe the architecture and functions of our ACM framework.

3. Architecture of the Align My Curriculum (ACM) Framework

The ACM framework is designed to enable the classification, comparison and visualization of online computing jobs and computing curricula. The framework consists of four modules using computing techniques: A data collection and pre-processing module, a classification module, a data comparison and similarity module, and a visualization module (as shown in Figure 1). The data collection and pre-processing module obtains the datasets from two sources and puts them into a format suitable for internal processing of the ACM framework.

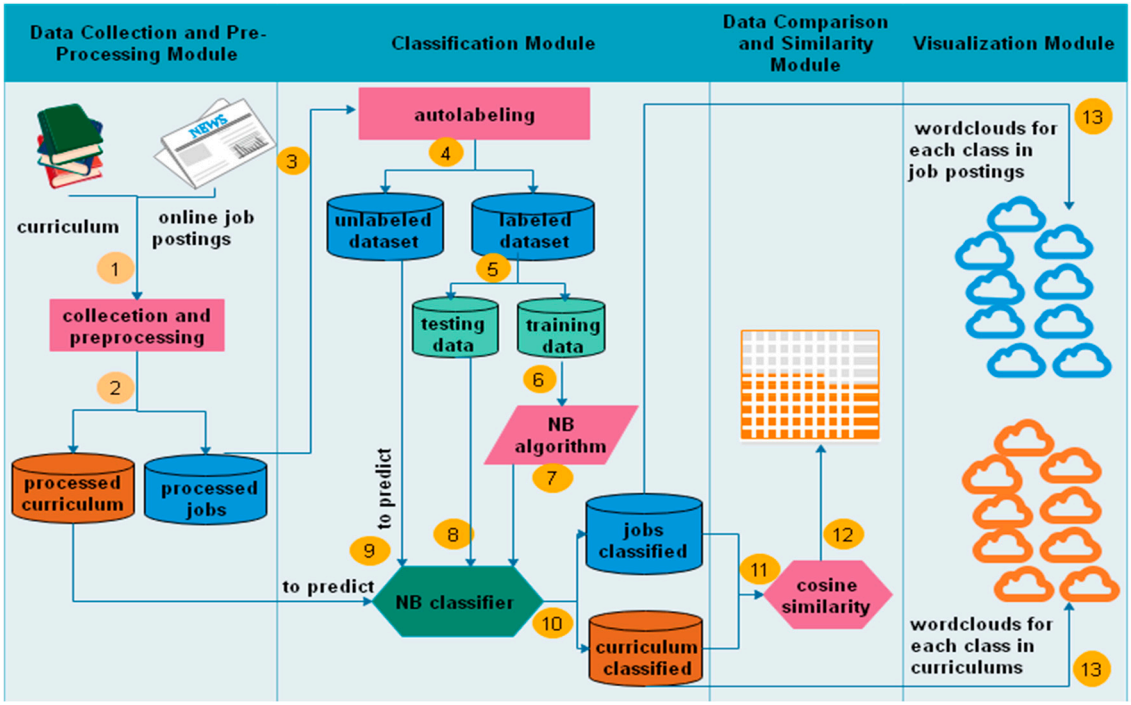

Figure 1. Architecture of the Align My Curriculum (ACM) framework.

The classification of processed data takes place in classification module. This module defines classes to label the data and uses the Naïve Bayes algorithm for classification of the data. The results of this module are forwarded to the last two modules. The data comparison and similarity module is for measuring the similarity between the classified datasets. The visualization module displays the content of the classified datasets. Here, we further describe the technical details of each module in the ACM framework.

3.1. Data Collection and Pre-Processing Module

This module is used to collect the data and pre-process it. The data used in this study refer to two data sources: online job postings and curricula (as shown in Figure 1). First, the job postings are extracted from popular job websites by making use of the freely available web-scraping service provided by Gresper (www.grepsr.com). The computing curriculum datasets are from the courses run by departments, and are downloaded directly from the curriculum repository and saved to two separate documents. After collecting the two datasets from the two sources, they are pre-processed by transforming the raw data, with noisy items, into clean data to be analyzed efficiently. The pre-processing includes tasks such as sampling and feature extraction to determine if it is the necessary to clean the data.

3.2. Classification Module

In this module, the job postings and the curricula are classified on the basis of a common factor called a label, according to the included skills and knowledge. The classification employs the technique of text mining using a supervised algorithm to predict the label as the output for data instances based on their content [23]. Additionally, a common factor is needed for automated comparison between jobs and curriculum. The classification algorithm helps to classify each job posting and course to just one label. These labels are then used to compare job postings to the curricula. Building a model for text classification needs us to label most records to create a labeled dataset as a training dataset and to test (as a testing dataset) the model. Due to the main work objective of computerizing the process

of data analysis, the datasets are labeled by using the keyword-search function in R Studio. Defined classes are needed to label the data.

The challenge in this module involves the focus on the computing skills identified in the general classes for computing skills. We state our basic hypothesis that all computing jobs require a mix of technical skills, business skills, and problem-solving skills. In this work, we refer to all these as computing skills. As the next step, the ACM framework uses this labeled dataset to train and test the model so it can classify all the data on the basis of their content into a single pre-defined class, as mentioned above. In this work, we have two classification problems. There are more than two classes (a multi-class text classification problem), and we have to classify each record into just one to obtain mutually exclusive classes (a classification problem). Accordingly, the Naïve Bayes algorithm was chosen to meet these requirements due to its simplicity and efficiency [24–29]. The algorithm is trained on the labeled data to determine the best class for each record by finding the maximum a posteriori class [29]. After training, we have to evaluate whether the classifier has learned (classified) correctly or not by using different metrics appropriate to the classification problems in this work. If we find that the classifier performance is satisfactory, we can use it to classify the new data. In this module, the Naïve Bayes classifier uses knowledge acquired from the building process as input in classifying the new data [30,31]. The implementation of Naïve Bayes is adapted from the "quanteda" package in R. The results of this module are discussed in Section 4.2.

3.3. Data Comparison and Similarity Module

In this module, two datasets are compared, and their similarity is identified. The process of data comparison and similarity identification takes place on the basis of the label attributes identified by the classification module. By comparing different sources of data, we can understand the interaction between them and reduce the gap [32,33]. This module uses the output of the two above modules (i.e., data collection and pre-processing module, and classification module) as inputs. After cleaning and classifying the datasets, their similarity, or closeness, is measured. The algorithmic question is whether the two documents are similar or not. The simplest way to determine similarity is to use cosine similarity algorithms to measure the angle between two datasets according to their words. The ACM framework measures the closeness between jobs and courses in the same class by using cosine similarity. For example, security courses are compared with security jobs to find the similarity, and the first step in this technique represents the documents as vectors, which involves converting the text of the document to a list where the elements are floating numbers and lose the information on the words' order [34]. It can show how many times each word occurs in the datasets. To achieve efficient results, the cosine similarity needs an additional pre-processing technique to be performed on the datasets, such as converting the words to their root form (stemming function). This helps to establish the term frequency (TF) and the weighting factor number of each word and to compare the occurrence of the words in the two datasets. For example, if we have the word "manager", "management", and "managers", all of them have "manage" as their root. After that, we need to compute the cosine similarity between the words [35].

The above steps are repeated for each pre-defined classes. In text similarity, the values of cosine are always between zero and one, which means that the angle can never be greater than 90 degrees (text vectors are usually positive). If the cosine value is near to one, the datasets have almost the same words and direction, whereas if the cosine value is near to zero, the datasets do not have the same words. The results of this module establish the similarity between curricula and job postings, and are undertaken by automatic comparison (Section 4.3).

3.4. Visualization Module

One of the key contributions of the framework is to provide a method to visualize the major content of curricula and job datasets to help academics and educators to understand the skills acquired and required in each, respectively [11]. There is a plethora of text mining and visualization tools

to facilitate the innovative process of uncovering hidden information on curricula and jobs. In this work, we use word clouds to visualize the content of the job posting and curriculum classes. This can assist in the analysis of the text by identifying words that frequently appear in a set of documents [36]. Initial modules of the ACM framework are used to clean, pre-process and divide input datasets into exclusive classes on the basis of their content. In this module, these classes are visualized to discover more detailed information about the required skills from job postings, and acquired skills from the curricula. To obtain an accurate word cloud, the ACM framework makes use of a tokenizing technique whereby text is tokenized into a two-consecutive-word bigram to help to understand the text's content. By contrast, most word clouds involve the use of stemming, putting all forms of the words together to increase the frequency of those words, however, this technique leads to misunderstanding the words' meanings, especially with computing vocabulary [37]. After tokenizing, a numeric weight is assigned to each word. Our framework makes use of the Term Frequency-Inverse Document Frequency weight (TFIDF) for each word to find the most important words in the two datasets. The most important bigrams for each class are displayed on the basis of their TFIDF scores by using the packages "word cloud" and "RWeka". Finally, the ACM framework draws each word accordingly, allocating the font size and color on the basis of the magnitude of various constants.

4. Experiments and Discussion

In this section, we describe the experimental results of each module of the ACM framework, the data used and analysis of resulting datasets.

4.1. Data Collection and Pre-Processing Module

As mentioned before, the collected datasets in this work are from two sources: curriculum websites and job websites. Data about jobs is collected from well-known job portals and websites such as bayt.com, naukrigulf.com, gulftalent.com and linkedin.com/jobs. These portals and websites group jobs according to various parameters, such as geography, department and expertise. We extracted the job postings related to information technology, information system, and computer science departments in Saudi Arabia. In addition, we used keywords from computing fields such as information, network, software, database, and computer. The data collection process extracted two months' jobs data, from April to June 2018. The results contain basic information on each, such as the job title, company, location, description, and so on. We carefully examined the job postings, deleting duplicates and irrelevant entries. In total, we had a sample of 2550 job postings. The collected dataset was loaded into R and the attributes of interest were selected, in a process known as future extraction. This process helps to improve the analysis results. There is no fixed template for job postings; every company uses its template, as needed. This leads to many difficulties in processing such postings and in extracting the right data entities [38]. Most formats are divided into several sections, such as the nature of the environment and the business, the job benefits and offer such as salary information, and describing the nature of the job, including the skills and knowledge required. This section of the post contains the information of interest to this study, and it is always referred to as the job description or job summary [13]. This study focuses on two types of information to analyze job postings: Job title and job descriptions for each job posting, and eliminates all other attributes. As a next step, it saves to a new file (i.e., job data) for further processing by using R functions. Each job posting represents one record in a dataset with three attributes: ID, job title, and description.

By contrast, the curriculum datasets are downloaded directly from curriculum websites into R. The collected curriculum data are structured, unlike the job data, and this facilitates the process of selecting the attributes that include information valuable to this work. The two types of information to analyze curricula are the course title and course description, and all other attributes were eliminated. We had 118 records for computing curricula from the website of King Abdulaziz University for all computing departments (as a case study). The datasets have three columns: ID, course title, and course description. Then, this information is saved into a new file (i.e., curriculum data) for further

processing in R. The two datasets in a structured form were processed to improve the performance of the classification model [39].

In this work, datasets are varied in structure, length, and type, and need extensive cleaning as the functions and techniques change depending on the content of the two datasets. The preprocessing techniques are used: tokenize the text into a single word, we changed capital letters to lower case and removed all punctuation, stop words and common English words (e.g., the, a, an, or, etc.), words that appear rarely "the frequency of words = 1", words that appear too rarely or too frequently in the document to contribute to identifying the class of the document, all whitespace through "if there is more than one space between words" and numbers "any numbers within the text" [40]. Finally, in this module, we arrive at two cleaned datasets to save for use in the next module (2550 job postings and 118 computing curriculums). After processing, the two datasets were used in all modules in this work. For some modules, we needed further cleaning techniques, in which case we used temporary measures to achieve the task then returned the data to their original form.

4.2. Classification Module

This module has various outcomes, ranging from labeling the documents to predicting classes for new data. Labeling is the first step in classifying the dataset after the pre-processing stage. This work advocates auto-labeling the datasets to save time, rather than labeling each document manually. The defined classes for labeling the datasets were obtained from the most comprehensive and latest research conducted in most computing fields [4,11,41]. We chose to apply this function in the Job title attribute [11]. Curriculum data are disregard in this step because we needed only a sample of all datasets to train the model. For this purpose job postings were more appropriate, due to their varied size and greater detail. Ultimately, it makes the model learn easily. The deductive method (previously used categories in the literature) is used to identify the work classes and these are used to categorize the datasets into independent groups. The aim is to classify every related skill into one group to determine the skills of each class. The predefined classes in this work are: Development (keyword: develop, programmer), Analysis (keyword: analyst), Testing (keyword: test, quality), Network Administrate (keyword: network), Database Administrate (keyword: database), Security Administrate (keyword: security), Management (keyword: manager), Support (keyword: support), and Design (keyword: design). After creating the classes of skills, we retrieved the job postings for which the job title matched the defined keywords in each class, In other words, the keywords were used to search on the title of all job postings [11]. For example, we had two job postings with the job title 'system analyst' and 'ASP Dot Net Programmer'. The labeling of these two job titles is analysis and development, respectively, because of the words analyst and programmer. The result added a new column for the class to assign the labels for each record, and such data are called labeled data. Each record must have only one class (label) to create an exclusive subgroup for discriminating between the skills of each group. To ensure this, we merged all subgroups and removed duplicate records, which have multiple classes due to their job titles matching several keywords. For example, 'network analyst' is duplicated in the 'network' class and the 'analyst' class. Also, we removed the fuzzy records where the job title did not match any of our specific keywords. The fuzzy and duplicated job titles were saved to a separate file called 'unlabeled dataset'. The result of this step was 1173 job postings, labeled into distinct classes, and 1386 job postings that were unlabeled. Figure 2 shows the distribution of the classes of labeled data. What can be seen is the inequality of classes and the dominance of development and management jobs. At this point, we had three datasets: labeled jobs, unlabeled jobs, and curriculum. The labeled dataset was used to train and test the model, and after that, the model could predict the classes for all data based on their content. The 1173 labeled jobs, divided into two subsets, were our training and testing datasets. We trained the model using 80% of the labeled jobs and used the remaining 20% to test its performance. We used the random sampling technique, which is useful to reproduce the starting point of the sequence in random numbers to ensure that all the labels are present in the training and testing datasets [42].

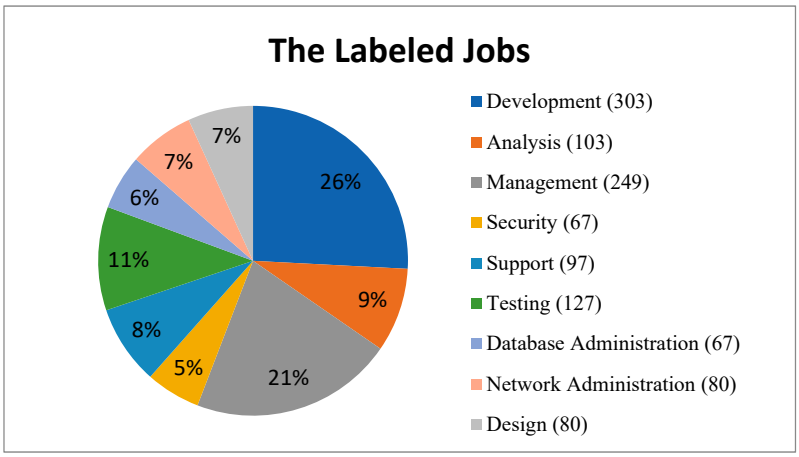

Figure 2. Classes distributed in labeled dataset.

As seen in Figure 2, the labeled datasets are imbalanced, so we used the function of *createDataPartition()* in R to create a balanced dataset by selecting random sampling from every class to preserve similar proportions in the overall distribution of the dataset. This ensures that all classes are divided equally, and the model can be trained for all types of class. We had 942 records in training data and 231 records in testing data, and the distribution of classes showed the same proportions in the two datasets, as in Figure 3. The classifier uses the main bodies of data, the job description attribute, to classify the text, discarding the job title attribute [11]. The model computes the probability of each class label based on the words in the description attribute, and picks the class with the greatest probability [31].

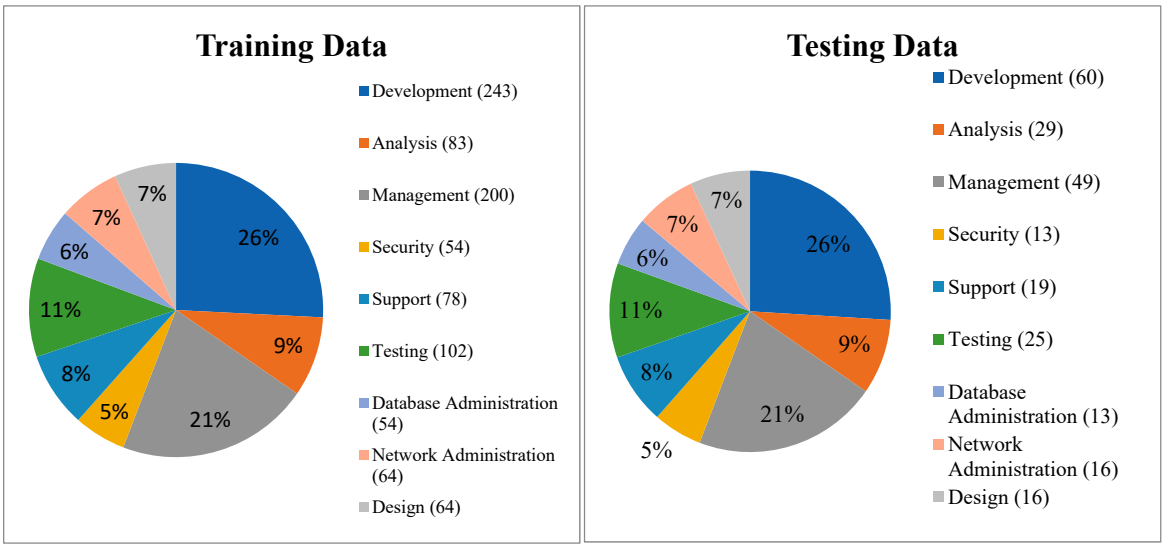

Figure 3. Classes distributed in training and testing datasets.

The model can make sense of training and testing data by transforming them into a word frequencies format. First, it represents the datasets as a corpus then constructs a document term frequency matrix (DFM) to represent the records as word counts; this is called a 'bag-of-words' approach. The model creates the frequency table for each word in the training data against each class and determines the initial weight for every record [43]. The model calculates the probability of each word in the class and the probability of the class. Next, the classifier learns what each class looks like, and then classifies the new data into the predefined classes based on the content of the data. Here, we use a Naïve Bayes classifier. After training, the model is tested by comparing the predictors' classes for

test data with the actual classes to evaluate whether the model learned correctly or not. The model's performance has been computed using the average of precision, recall, and f1-score metrics [30,44,45] for each class (as shown in Table 1). The formulas for these metrics shown in Equations (1) and (2).

$$\text{all Precision} = \frac{\sum_{i=1}^{k} \text{TPi}}{\sum_{i=1}^{k} (\text{TPi} + \text{FPi})}, \text{ all Recall (sensitivity)} = \frac{\sum_{i=1}^{k} \text{TPi}}{\sum_{i=1}^{k} (\text{TPi} + \text{FNi})} \quad (1)$$

$$\text{F1score for the model} = 2 \times \frac{\text{all Precision} \cdot \text{all Recall}}{\text{all Precision} + \text{all Recall}} \quad (2)$$

where TP "True Positive" is the number of correct classifications, FP "False Positive" is the number of incorrect classification, and FN "False Negative" is the number that is not recognized as a class.

Table 1. Model Performance.

Class / Metrics	Development	Analysis	Testing	Network	Database	Security	Management	Support	Design	Average
Precision	0.88	0.78	0.88	0.92	0.84	0.78	0.80	0.80	1	0.85
Recall	0.90	0.55	0.92	0.81	0.84	0.84	0.97	0.63	0.87	0.81
F1-score	0.89	0.64	0.90	0.86	0.84	0.81	0.88	0.71	0.90	0.83

We conclude that the identified algorithm has the best performance. It should be noted that the results could vary according to the applied sampling and pre-processing techniques. In the end, the classifier that has been built can predict the label for new data in the datasets of unlabeled jobs and computing curricula. The result of this module is that all datasets be classified into one of the predefined classes. Figure 4 shows the proportion of classes in curriculum and job posting records. These two charts can yield valuable information when read by curriculum developer or academics. As seen in Figure 4, more than 33% of courses are classified to be in the analysis class, whereas the design and support classes comprise just 2% of all courses. Table 2 displays some example from the two datasets after classification for additional verification of the classifier predicts.

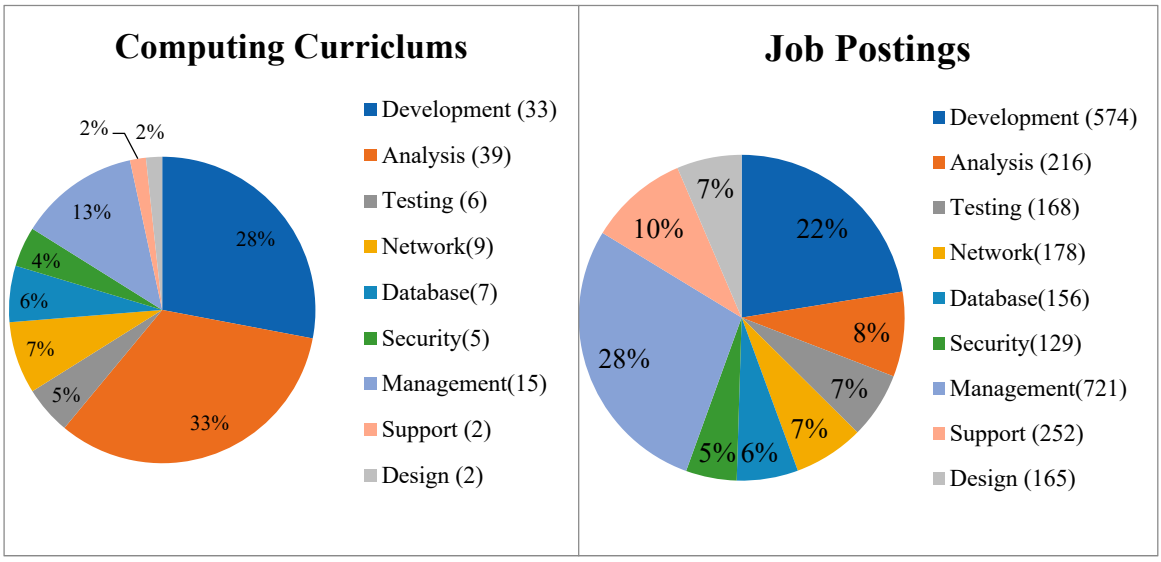

Figure 4. Classes distributed in the datasets.

Table 2. Sample of records after classification by AMC model.

ID	Course Title	Label
3	Principles of Operating Systems	Support
16	IS Applications Design and Development	Developer
30	IS Strategies and Policies	Manager
34	Distributed Systems	Network Administrator
44	Computer Architecture	Support
56	Database Administration	Database Administrator
ID	**Job Title**	**Label**
54	Agile Scrum Master Banking	Manager
136	Big Data Consultant	Analyst
211	Call Center Agent	Support
225	Chief of Cyber Defense	Security
460	Erp Oracle Sr Consultant Ksa	Developer
487	Fiber Optic Technician	Designer

4.3. Data Comparison and Similarity Module

The results of this module help us to understand the relationship between the data on curricula and jobs. This relationship is investigated by employing the cosine similarity algorithm, combined with the term frequency (TF) using the lsa package. To simplify the calculation of TF, words are stemmed in a temporary cleaning step. Calculating the cosine similarity helps to find similar pairs of the class or sets of similar classes. The results of the cosine similarity between the two datasets are presented in Figure 5. The classes of job postings are assigned to the x axis, and the classes of courses to the y axis, thus the blue gradient indicates the value of cosine similarity. The figure clearly shows which documents are more similar than others, and the two datasets are related but not close to each other. Dark blue means near to 0, which signifies that the data are dissimilar, whereas light blue means there is closeness. For example, the content (the description of skills) in the analyst class is similar to courses in the analyst class, and the color of the intersection cell shows this in light blue. The analyst jobs are somewhat similar to developer courses, and while design jobs are different from the skills in design courses, according to the color of the intersection cell (dark blue). In the end, as we see from the cosine results, there is always a gap between courses and jobs datasets.

Figure 5. Heatmap of Cosine Similarity.

4.4. Visualization Module

The visualization module takes the result of the first two modules as input and processes it, for better visualization. The word cloud technique is applied to visualize the content of each class in the two data sources (e.g., the content of design courses and design jobs) to extract further valuable information. To generate an efficient word cloud, the text is tokenized into bigrams and to count the TFIDF. The words in a larger size have a higher TFIDF value in the text. The results of this module are word clouds for each class in job postings and each class in computing curricula. For example, the word clouds for development jobs and development courses show the content of jobs and courses in the development class (as shown in Figure 6). The development jobs (A) need excellent knowledge on the development of software, mobile and web services, and applications with an emphasis on problem-solving. The development jobs want the following programming languages: SQL, Java, oracle, html, VISUAL studio and asp.net. Also, the required skills are business intelligence and working as a team, while the development courses (B) provide students with the skills in systems, and in mobile and web development. In the labs of these courses, PHP and HTML programming languages are taught, as appearing in the word clouds. From this, we conclude that the word clouds are a useful way to extract information from the texts.

Figure 6. Word Cloud of Development jobs and courses. (**A**) The development jobs. (**B**) The development courses.

5. Conclusions

One of the most pressing topics these days in both academia and industry is training staff according to the needs of the industry (job market). To do so, educational institutions are continuously under pressure to find and bridge the gap between their curricula and the job market. A framework referred to as the Align My Curriculum (ACM) framework has been presented to help curriculum developers to identify the skills gap and make suggestions to fill this gap between curricula and the skilled staff required in the industry/market. The ACM framework can be used to analyze, in an automated fashion, the required skills in the market and the acquired skills of the curriculum. The objectives were to use various computer-related techniques to analyze and compare job postings and curricula to help academics and curriculum developers to bridge the skills gap and design more effective courses. To the best of our knowledge, no existing research has provided an automated framework to analyze and compare job postings with curricula at the same time. Use of Naïve Bayes model in the AMC framework resulted in precision of 85%, recall of 81%, and an F1-score of 83%. Additionally, improved results of cosine similarity (i.e., results of the cosine similarity are near to zero) increased the accuracy of the framework. Finally, the use of word cloud with TFIDF and bigram techniques in the AMC

framework improved the visualization of each class in the two data sources. This technique gave us a valuable summary of the content of each class.

We applied our ACM framework to the Saudi job market and the curricula of the Faculty of Computing and Information Technology (FCIT). The framework performed the entire process of determining the required skills in the market, comparing it with the skills acquired from the curricula to identify the gaps between them, suggesting possible points to bridge the gap. The framework has several strengths. First, the use of online data facilitates analysis over time. Second, it can be used for other purposes besides curricula and job posting comparisons. There are also some limitations to this work: We excluded postings written in Arabic, because of our focus on the English language, and the data in this work are multidimensional. Despite this limitation, the presented framework is an efficient tool not only for curriculum developers or academics but also for analysts, students, job applicants, and many others. Additionally, it provides insights into the predominant jobs and skills in computing fields.

Author Contributions: A.A., M.A.A. and K.S. worked on the theoretical concepts, methodology, implementation and experiments. A.A., M.A.A. and N.R.A. prepared the setup for experiments and analyzed the data. A.A., K.S. and N.R.A. did the comprehensive literature review.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jin, X.; Wah, B.W.; Cheng, X.; Wang, Y. Significance and Challenges of Big Data Research. *Big Data Res.* **2015**, *2*, 59–64. [CrossRef]
2. Askitas, N.; Zimmermann, K.F. The internet as a data source for advancement in social sciences. *Int. J. Manpow.* **2015**, *36*, 2–12. [CrossRef]
3. Kureková, L.M.; Beblavý, M.; Thum-thysen, A. Using online vacancies and web surveys to analyse the labour market: A methodological inquiry. *Iza J. Labor Econ.* **2015**, *4*, 2. [CrossRef]
4. Woolridge, R.W. What's In and What's Out: Defining an Industry-Aligned IS Curriculum Using Job Advertisements Rachida Parks University of Arkansas at Little Rock. *J. High. Educ. Theory Pract.* **2016**, *16*, 105–120.
5. Smith, D.; Ali, A. Analyzing Computer Programming Job Trend Using Web Data Mining Literature Review—Web Data Mining. *Issues Inf. Sci. Inf. Technol.* **2014**, *11*, 203–214.
6. Carnevale, A.P.; Jayasundera, T.; Repnikov, D. Understanding Online Job ADS Data, Technical Report. 2014. Available online: https://cew.georgetown.edu/wp-content/uploads/2014/11/OCLM.Tech_.Web_.pdf (accessed on 7 May 2019).
7. Kureková, L.M.; Kureková, L.M.; Anna-Elisabeth, T. *Using Internet Data to Analyse the Labour Market: A Methodological*; Institute for the Study of Labor (IZA): Bonn, Germany, 2014.
8. Müller, O.; Schmiedel, T.; Gorbacheva, E.; Brocke, J.; Müller, O.; Schmiedel, T.; Gorbacheva, E.; Brocke, J. Towards a typology of business process management professionals: identifying patterns of competences through latent semantic analysis. *Enterp. Inf. Syst.* **2016**, *10*, 50–80. [CrossRef]
9. Kim, J.; Angnakoon, P. Research using job advertisements: A methodological assessment. *Libr. Inf. Sci. Res.* **2016**, *38*, 327–335. [CrossRef]
10. Litecky, C.; Aken, A.; Ahmad, A.; Nelson, H.J. Mining for Computing Jobs. *IEEE Softw.* **2010**, *27*, 78–85. [CrossRef]
11. Wowczko, I. Skills and Vacancy Analysis with Data Mining Techniques. *Informatics* **2015**, *2*, 31–49. [CrossRef]
12. Watson, H.J.; Young, D.; Miranda, S.; Robichaux, B.; Seerley, R. Requisite Skills for New MIS Hires. *Sigmis Database* **1990**, *21*, 20–29. [CrossRef]
13. Todd, P.A.; McKeen, J.D.; Gallupe, R.B. The Evolution of IS Job Skills: A Content Analysis of IS Job Advertisements from 1970 to 1990. *Mis Q.* **1995**, *19*, 1–27. [CrossRef]
14. Albin, M.; Otto, R.W. The CIS Curriculum: What Employers Want from Cis and General Business Majors. *J. Comput. Inf. Syst.* **1987**, *27*, 15–19.

15. Eom, M.; Cheehwan, L. Critial skills to be comptetent and relevant IT personnel: Do today's IT personnel possess requisite skills? *J. Inf. Technol. Manag.* **2012**, *23*, 33–49.
16. Gorgone, J.T.; Davis, G.B.; Valacich, J.S.; Topi, H.; Feinstein, D.L.; Longenecker, H.E., Jr. IS 2002 Model curriculum and guidelines for undergraduate degree programs in information systems. *Commun. Ais* **2003**, *11*, 63. [CrossRef]
17. Gorgone, J.T.; Gray, P.; Stohr, E.A.; Valacich, J.S.; Wigand, R.T. Msis2006 Curriculum Preview. *Commun. Assoc. Inf. Syst.* **2005**, *15*, 544–554.
18. Lee, C.K. Analysis of Skill Requirements for Systems Analysts in Fortune 500 Organizations. *J. Comput. Inf. Syst.* **2005**, *45*, 84–92.
19. Sodhi, M.S.; Son, B.-G. Content analysis of OR job advertisements to infer required skills. *J. Oper. Res. Soc.* **2010**, *61*, 1315–1327. [CrossRef]
20. Zhang, S.; Li, H.; Zhang, S. Job opportunity finding by text classification. *Procedia Eng.* **2012**, *29*, 1528–1532. [CrossRef]
21. Debortoli, S.; Müller, O.; Vom Brocke, J. Comparing business intelligence and big data skills: A text mining study using job advertisements. *Bus. Inf. Syst. Eng.* **2014**, *6*, 289–300. [CrossRef]
22. Kim, J.Y.; Lee, C.K. An Empirical Analysis of Requirements for Data Scientists Using Online Job Postings. *Int. J. Softw. Eng. Its Appl.* **2016**, *10*, 161–172. [CrossRef]
23. Westergaard, D.; Stærfeldt, H.-H.; Tønsberg, C.; Jensen, L.J.; Brunak, S. A comprehensive and quantitative comparison of text-mining in 15 million full-text articles versus their corresponding abstracts. *Plos Comput. Biol.* **2018**, *14*, e1005962. [CrossRef] [PubMed]
24. Subramanian, S. Document Classification Using Multinomial Naive Bayes Classifier. *Int. J. Eng. Technol.* **2014**, *3*, 1557–1563.
25. Rajeswari, R.P.; Juliet, K. Text Classification for Student Data Set using Naive Bayes Classifier and KNN Classifier. *Int. J. Comput. Trends Technol.* **2017**, *43*, 8–12.
26. Manning, C.D.; Raghavan, P.; Schütze, H. Text classification and Naive Bayes. In *An Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008; pp. 260–264.
27. Manning, C.D.; Ragahvan, P.; Schutze, H. An Introduction to Information Retrieval. *Inf. Retr. Boston.* **2010**, *16*, 100–103.
28. Roy, S.S.; Kaul, D.; Roy, R.; Barna, C.; Mehta, S.; Misra, A. Prediction of Customer Satisfaction Using Naive Bayes, MultiClass Classifier, K-Star and IBK. In *Soft Computing Applications*; Balas, V.E., Jain, L.C., Balas, M.M., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 153–161.
29. Wang, S.; Jiang, L.; Li, C. Adapting naive Bayes tree for text classification. *Knowl. Inf. Syst.* **2015**, *44*, 77–89. [CrossRef]
30. Ashari, A.; Paryudi, I.; Tjoa, A. Performance Comparison between Naïve Bayes, Decision Tree and k-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool. *Int. J. Adv. Comput. Sci. Appl.* **2013**, *4*, 33–39. [CrossRef]
31. Zhang, H.; Li, D. Naive Bayes text classifier. In Proceedings of the IEEE International Conference on Granular Computing, GrC, San Jose, CA, USA, 2–4 November 2007; pp. 708–711.
32. Shen, Y.; Lin, G.T.R.; Lin, J.; Wang, C. A Cross-Database Comparison to Discover Potential Product Opportunities Using Text Mining and Cosine Similarity. *J. Sci. Ind. Res.* **2017**, *76*, 11–16.
33. Allahyari, M.; Trippe, E.D.; Gutierrez, J.B. A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. *arXiv* **2017**, arXiv:1707.02919.
34. Gomaa, W.H.; Fahmy, A.A. A Survey of Text Similarity Approaches. *Int. J. Comput. Appl.* **2013**, *68*, 13–18.
35. Al-Anzi, F.S.; AbuZeina, D. Toward an enhanced Arabic text classification using cosine similarity and Latent Semantic Indexing. *J. King Saud Univ. Comput. Inf. Sci.* **2017**, *29*, 189–195. [CrossRef]
36. Cherapanukorn, V.; Charoenkwan, P. Word Cloud of Online Hotel Reviews in Myanmar for Customer Satisfaction Analysis. In Proceedings of the 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), Hamamatsu, Japan, 9–13 July 2017; pp. 447–452.
37. Jayashankar, S.; Sridaran, R. Superlative model using word cloud for short answers evaluation in eLearning. *Educ. Inf. Technol.* **2017**, *22*, 2383–2402. [CrossRef]
38. Hirudayaraj, M.; Baker, R. HRD competencies: Analysis of employer expectations from online job postings. *Eur. J. Train. Dev.* **2018**, *42*, 577–596. [CrossRef]

39. Gaigole, P.C.; Patil, L.H.; Chaudhari, P.M. Preprocessing Techniques in Text Categorization. Proceedings published by International Journal of Computer Applications. 2013. Available online: https://pdfs.semanticscholar.org/ff34/7657082e70347a916548a9fe567ab791162a.pdf (accessed on 7 May 2019).
40. Gon, C.A. The impact of Pre-Processing on the Classification of MEDLINE Documents. 2015. Available online: https://www.scitepress.org/papers/2010/30287/30287.pdf (accessed on 7 May 2018).
41. ISCO—International Standard Classification of Occupations. Available online: http://www.ilo.org/public/english/bureau/stat/isco/isco08/index.htm (accessed on 23 January 2018).
42. Jockers, M.L. *Text Analysis with R for Students of Literature*; Springer: Berlin/Heidelberg, Germany, 2014; ISBN 978-3-319-03163-7.
43. Jabbar Alkubaisi, G.A.; Kamaruddin, S.S.; Husni, H. Stock Market Classification Model Using Sentiment Analysis on Twitter Based on Hybrid Naive Bayes Classifiers. *Comput. Inf. Sci.* **2018**, *11*, 52. [CrossRef]
44. Patil, T.R. Performance Analysis of Naive Bayes and J48 Classification Algorithm for Data Classification. *Int. J. Comput. Sci. Appl.* **2013**, *6*, 256–261.
45. Jurafsky, D.; Martin, J. Naive Bayes and Sentiment Classification. In *Speech and Language Processing*. 2017. Available online: https://web.stanford.edu/~{}jurafsky/slp3/4.pdf (accessed on 7 May 2019).