FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Enhanced Methods of Job Offers mining from the World Wide Web

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

**Doktor der Technischen Wissenschaften**

by

**MSc. Ehtesham-ul-haq Dar**
Registration Number 0647752

to the Faculty of Informatics

at the TU Wien

Advisor: Dr. Jürgen Dorn

The dissertation has been reviewed by:

| | |
|---|---|
| Forename Surname | Forename Surname |

Vienna, 10th January, 2018

Ehtesham-ul-haq Dar

# Declaration of Authorship

**MSc. Ehtesham-ul-haq Dar**
**Brigittenauer Lände 224/6838, 1200 Vienna**

I, Ehtesham-ul-haq Dar, declare that this thesis titled, 'Enhanced Methods of Job Offers mining from the World Wide Web' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: _____

Date: _____

I would like to dedicate this thesis to my loving parents ...

# Acknowledgements

And I would like to acknowledge ...

# Abstract

The significance of employment in the setup of a society is quite evident. The methods of employment procurement are gradually changing from conventional to digital. The internet has become a prominent source of job procurement. Online job offers opened the research opportunities to explore different methods for the automation of online jobs classification and retrieval. Classification of web documents as job opportunities required a mechanism from Machine Learning or some other domain. To automate the retrieval of online job opportunities, text classification is an only viable method - in case of machine learning. The Semantic web mining is also a possible solution for job offers classification. We studied different methods for job offers classification, from machine learning and semantic web technologies. More than 5000 job offers were collected from multiple existing job offer websites for this study.

From machine learning discipline, we investigated eight text classifiers to study their effectiveness and generalization performance on new data. Job offers dataset is pre-processed with different available methods and a newly defined method, and arranged into five groups for classification. Classifiers are regularized to avoid high variance, and their effectiveness parameters and generalization errors were evaluated. All the classifiers showed >90% accuracy but generalization errors varied. Ridge Regression and Stochastic Gradient Decent generalized well on new data, for all the groups. On the contrary Random Forest and Perceptron tenacious toward high variance. We found two classifiers that generalized well to new data.

From semantic web technology, we proposed a scalable ontology-based classifier. This enhanced classifier classify generic as well as specific job offers. We used the ontology to: extract concepts from job offers text description, find the minimum threshold for classification, and developed a classification model. We did not use any Machine Learning algorithm to develop this classifier. We evaluated this classifier according to Machine learning evaluation mode - training, and testing dataset. Our classifier showed >90% accuracy, precision, and recall, for both training and testing dataset.

With these promising results of the defined methods we can automate the job offers categorization and retrieval from the World Wide Web.

# Contents

# List of Figures

Chapter 1

# Introduction

Employment remained a social significance throughout all ages in every society. Unemployment is one of the major concerns which is addressed worldwide. Every possible effort due available is put into practice to solve this problem, moreover, new methods based on technologies are proposed to solve this problem. Unemployment is an issue for all developing, emerging, and developed countries and these countries deal with this problem according to their resources, based on the usage of conventional, technological, or both methods. According to International Labor Organization (ILO) the global employment rate is excepted to increase from 5.7 percent to 5.8 percent in 2017, which comprising 3.4 million people worldwide, and total unemployed persons in 2017, according to a modest forecast, would be over 201 million persons(1).

The emergence of the internet and the advent of the World Wide Web changed every aspect of life and society around us, including human resource management. The usage of internet increased extensively in the mid-1990s, which gave birth to the online recruitment industry, and inclination towards this easy to access resource for job search increased ever science (2), and this has phenomenal success in a very brief span of time. The Internet and the World Wide Web changed the conventional employment search methods, and the internet has become a rich and cost-effective source to fulfill the growing needs of employers and employees, and online job searches trends ever since increased. The online recruitment or internet recruiting implies the formal sourcing of employment information online(2).

Nowadays internet has become a major source of recruitment and employment process in both progressed as well as developing countries. A survey conducted by LinkedIn in 2016 showed that 46% of jobs were offered on third party websites or online job boards(3).

## 1.1 Online Recruitment

Online recruitment defines a technological method of recruiting employees with the help of internet-based resources, like World Wide Web. According to Rudman (2010), and Härtel & Fujimoto (2010)(4), online recruitment is the selection of potential applicants, that applied for a job via the internet. In the online recruiting method a candidate, who is applying for a job, send his/her curriculum vitae (CV) to employer through email. That CV along with other received CVs are examined for recruitment(5). Some organization offers online forms and interested candidates fill and send the data to the interested employer. Different organizations recruit employee online or upload their job offers to different well known human resource management websites, to speed up the recruitment process and to get the most appropriate candidate for the opportunity. With the help of different technologies like, database management system, online advertising job portals, and the search engines, employers can recruit the required candidate in a fraction of time, as compared to traditional method of hiring. That process saved a considerable amount of time of the employer. Moreover, due to the worldwide access to the internet, employer organizations websites can be reached from any corner of the world, that provides opportunities for the skilled persons worldwide to take advantage.

## 1.2 Conventional vs Technological job searching

Conventional methods to search for a job include newspaper advertisements, job information referenced by some person, conducting a test for application to screen candidates, call candidate by phone, or post a letter for the interview. On the

contrary technological method for job searching is mainly through different organizations websites, which are interested to find a candidate. These organizations publish their jobs on their websites or use some job boards, like indeed[1], carrerbuilder[2], dice[3], to publish their jobs online. These job boards provide the search facility to users, to search available jobs, or a candidate can also upload its CV for consideration by the employer. Another method for candidate selection is, screening. This process comprises online tests to find out the appropriate candidate. Also, candidates are invited in response to CV shortlisting by email for interview(6). These job boards have become mediator between organizations and potential applicants to get a job(7)(8). Online recruitment attributed to low cost(9)(10), for both employee and employer. According to a survey, online recruitment provides quality candidates because online recruiting agencies provide shortlisted candidates. Another attribute related to online recruitment is its international coverage and these recruitment organizations offer their jobs for international candidates also. According to (11), in 1998 only 15% of unemployed persons searched for jobs through the internet, while that number dramatically increased to 74% in 2008. Approximately 19 million people visited job search sites in December 2011, while more than 24 million people searched jobs through job websites in January 2012, which represents 27% increase of job search trend(11).

## 1.3 Search engines for job search

Search engines are potential mean to find online job offers. However, to find a specific job which matches your requirement is not easy. The problem with search engine, such as Google, is that the keywords based search results come up with irrelevant job offers and the web users have to sift through those results to find out required job offer, which is a time-consuming effort. According to (12), on average only half of the result obtained from job-related queries were relevant. As an alternative, there are many job portals where a number of job offers are listed. Web job portals or virtual social networks have also dramatically changed the

[1] https://www.indeed.com
[2] https://hiring.careerbuilder.com
[3] https://www.dice.com

way people find jobs. Meta search engines are kind of a resource which searches through few portals to collect job offers; some research has conducted in meta search engines for job offers by (13). The coverage range of these portals and meta search engines for job offers is limited, for example, LiveCareer [1] cover more than 500 places to collect information, LinkUp [2] collect information from more than 20,000 company websites, and metasearch engine may be few job portals. Nonetheless, there are thousands of job offers which are available on the World Wide Web, not indexed by these portals or meta search engines, rather a person has to search manually through many portals to find the most relevant job offer. Due to the large volume of the Web [3], available job offers cannot be categorized manually and need a mechanism to automate job offers retrieval.

## 1.4 Job Offers Classifications

Job offers classification and retrieval from the internet required a classification mechanism. This mechanism must be capable of categorizing the unstructured Web documents, that were written in natural language, as job offers and eliminate the irrelevant web pages. The available information related to a job offer consists of many attributes like requirement, salary, location, contract etc., and this information helps to categorize a job offer. A sample of a job offer related to ".Net programming" is shown in figure 1.1.

There are multiple directions possible to categorize information from the World Wide Web. Machine learning or data mining proposes the solution through text classification. Semantic web technology is another potential solution available which can help to classify job offers from the World Wide Web (14)(15)(16). Web documents classification starts with the retrieval of web pages from World Wide Web, with the help of a retrieval system, like a search engine or crawler. Information or text is extracted from these retrieved web pages and preprocessed with different methods, and then a classification procedure is applied on that data to

[1] https://jobs.livecareer.com

[2] http://www.linkup.com/jobsINBox

[3] over 1 billion websites, according to Internet Live Stats (`www.internetlivestats.com`) on September 2014

determine either it is a job offer or an irrelevant document. We investigated multiple methods to classify online job offers. One of the methods used the machine learning approach to achieve the classification task, while other used domain ontology to define the classification model. Following sections described briefly about these procedures.

## 1.5 Data Preprocessing

The task of knowledge discovery consists of multiple steps including: collecting data from a certain domain, preprocessing of that data, transform data into a certain format, analyze data, evaluate and interpret its results(17). Preprocessing is an important and most time-consuming process before classification, or in general, knowledge discovery. Different experimental trials acknowledged that with appropriate preprocessing task, or combination of these methods on dataset, showed the decisive impact on the classification output(18).

Data preprocessing, also known as dimensionality reduction, is a process to make text data ready to use with analytical methods. Data preprocessing in general: is data cleaning, feature selection, formatting and integrating into a format that can be used for knowledge discovery algorithms(19). In other words, the purpose of preprocessing phase is to convert the raw data with the help of Natural Language Processing (NLP) methods into an appropriate input for the next phase, i.e. classification; which is conducted through different classification algorithms.

Text preprocessing contain two main tasks, feature selection, text documents representation into a certain format. Feature selection is a significant step in classification because noisy and redundant features selection degrades the classification algorithms effectiveness like accuracy, precision, recall; and classification model tendency towards overfitting increased(20).

Preprocessing removes some language dependent factors, among these factors are stop words removal, and stemming. Stop words are articles, pronouns etc., and these are always in abundance in text, and if not removed then classifier can biased towards them and cause degradation of classification. A word in a syntax can have different presentations in a language, for example, a verb has three different

presentations of present, past, and future participle, etc. The stemming is a process to convert different presentations of a word into a common base or root (21)(22).

All these language dependent factors must be resolved before classification. The documents for text classifications consists of lots of features and among them are also those features which are irrelevant and noisy(23)(24)(22). We can eliminate these features by applying some available statistical procedure (25), or may define some novel method, according to a certain domain requirements. Application of this procedure reduces the dataset size, and classification learning phase becomes more efficient(26).

Different steps for preprocessing are defined as following,

- **Tokenization:** A text document consists of string and these strings are parsed into words called tokens.
- **Stop Words:** Stop words such as articles, pronouns etc. are removed from the text documents
- **Stemming and Lemmatization:** Conflating, different presentation of same words, into common root with stemming and lemmatization, by applying different algorithms(27)

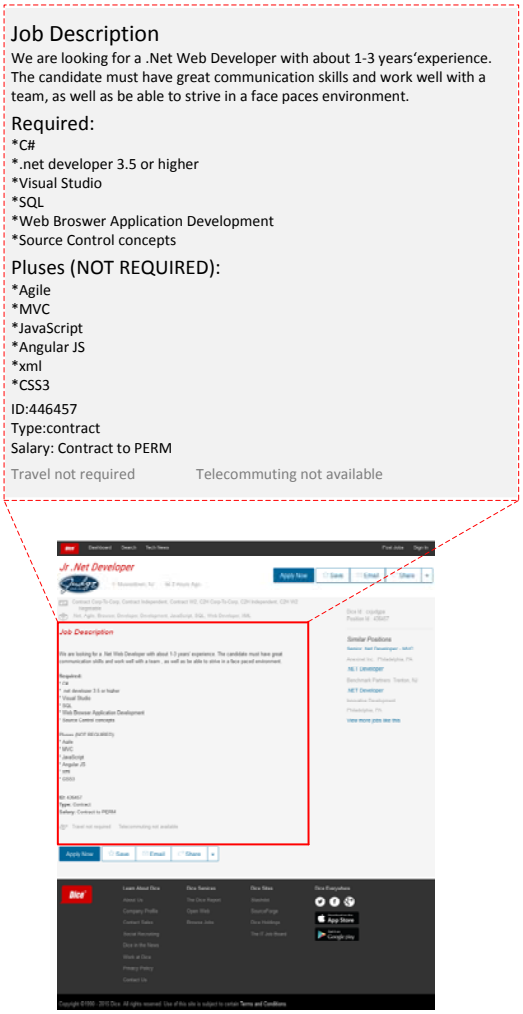


Figure 1.1: A IT job offer sample web page, taken from dice, shows the main textual content of a job offer

Documents are represented into a certain format before classification. Document are transformed into a bag or words, with corresponding word frequency(28), this representation is called vectorization (from vector space model). With the help of a statistical procedure called TF-IDF (term frequency-inverse document frequency), we can weight each word in available text documents(29)(30)(20). This vectorization form of text documents is used for classification procedure.

In summary, Feature selection removes the redundant and irrelevant words, and reduces the high dimensionality of vocabulary, which make classifier training phase more efficient, and improves classification accuracy(24), while vectorization is a document representation, that is used as input for classification algorithms.

### 1.5.1 Domain Related Preprocessing

In this research endeavor of job offers classification, we defined a novel preprocessing method for job offers domain, and study its impact on classification effectiveness. The task of this preprocessing method is to find some specific patterns, that were observed in job offers, and remove those patterns from the job offer text, and eliminate the redundant and irrelevant information from these job offers text. The purpose of this preprocessing method is to further clean the text data from features which are attributed to job offers description, but not contributing in the classification process. This dimensionality reduction makes the classifier training phase more efficient, and moreover it helped eliminate the overfitting problem in some of machine learning algorithms, for job offers classification.

## 1.6 Jobs Classification with Machine Learning

The study for job offers classification is motivated by the hypothesis that Machine learning text classification can assist to automate the classification of job offers from the World Wide Web. The objective of text classification is to build a procedure which is capable of automatically classify documents into predefined category or class. Machine learning already contains many state of the art text classification techniques, which elucidate three step text classification process as: presentation

of text documents, prepare the classifier for text documents, evaluation of classifier (31)(28). This approach to automating text document classification requires labeling text documents with predefined classes to create a dataset, for training a model. That model is then utilized to assign a class to a new text document.

In this method, we find out a classification procedure from machine learning discipline, which can help to classify potential job offers from the World Wide Web. For text classification a detail study can be found in(31)(28)(27). (27) discussed about Naïve Bayes, Support Vector Machine (SVM), K Nearest Neighbor (kNN), and Neural Network, for text classification. Fabrizio Sebastiani(31) discussed decision trees, regression methods, neural networks, kNN, SVM and classifier ensembles, for automation of text classification. (32) also discussed almost all the above mentioned classifiers for text classification. (33) discussed the text classification with the help of Stochastic Gradient Descent (SGD) classifier. These studies helped to select eight classifiers for job offers classification. From generative classifier category, we selected Naïve Bayes, from proximity based classifier we selected kNN and from discriminative category we selected SVM, Perceptron, Logistic regression, Ridge regression, Random Forest (decision tree ensemble)(34) and SGD.

There is no study available which discuss classifier(s) that learn from job offers text data and generalized for the new data, to categorize unseen text as job offers from the World Wide Web. Therefore, automation of job offers classification and retrieval, from the internet, is an opportunity to investigate, and we studied different classifiers that learn from the training data and generalize for new job offers data. For this purpose, we retrieved more than 5000 job offers from different online sources, the list of these online job offer resources are displayed in table 1.1. In this research scenario, we narrow down our classification to IT related job offers and tried to find appropriate text classification algorithm(s), which are free from generalization errors, so that this classifier model can be used to classify job offers from the World Wide Web. We found two classification algorithms, that are free from generalization errors, and are prospective candidates to classify job offers from the World Wide Web. This classification model fits with the paradigm of a binary class text classifier, where IT job offers are designated as positive class and all non-IT job offers, including few text samples related to news and articles, as

negative class. Once classifier(s) are trained and their generalization problem is eliminated, they are capable of classifying the new web documents as a potential job offer or vice versa.

The outcome of classification is mostly dependent upon the preprocessing of input data. We applied different preprocessing methods, or, combination of more than on methods on text data, to investigate its outcome on the classification procedure. To achieve the required classification outcome, we also developed a custom preprocessing method called '*Domain Related Preprocessing*' (Section 1.5.1), that is specific to job offers domain, it helped to reduce irrelevant and noisy data, that reduced dataset size and training phase of classification, and improved the effectiveness of classification model.

## 1.7 Ontology based Job offers classification

To find some specific information required a classification mechanism. As we have already discussed that, to find out the relevant job offers require an automatic process to classify a specific job information. Machine learning has lots of text classification algorithms and preprocessing methods to classify a specific text documents(31)(28)(27). Another possibility to mining the web for specific information is through semantic web technologies. Currently Semantic web technologies are being used for mining the web, and it is one of the fast growing research area(35). Ontology is one of the semantic web technologies, and used for different many purposes. Some ontology-based systems are also used to classify web pages, related to different topics. These systems used the ontology, along with different machine learning methods, to automate the classification of web pages(15)(16). For job specific domain, there are few ontology-based recruitment systems are proposed(36)(37)(38). (38) used deductive model for matchmaking of job offers and job applications, and then similarity measurement was used for ranking. (36) used an ontology to develop a job webportal, that provides recruitment services, while (37) used the German version of HR-XML ontology for the recruitment process. There is no study available which used an ontology for job

offers classification, without the use of machine learning. To fill this gap, we developed an ontology-based classification model. We developed an enhanced scalable ontology based classification model, to classify job offers.

According to Gruber "An Ontology is a formal, explicit specification of a shared conceptualization"(39). The ontology represents the area of knowledge in the form of the vocabulary of terms or concepts that may exist in some certain domain of interest. Ontology has two main advantages of shareability and reusability, to represent domain knowledge. That knowledge or vocabulary of terms can be used by people or multiple applications that need the information related to that domain(36). Concepts are categories which are organized in the form of taxonomy or hierarchy according to a presumed natural relationship, like inheritance in the object oriented programming. Since ontology has many advantages in enhancing computer abilities of understanding domain-specific knowledge, it is a good choice to introduce ontology into the research of information extraction and classification.

This study is the continuation of research for automatic classification of job offers web documents. To classify job offers web documents, an ontology based document classification method is proposed in this phase of research. The tool used to acquire this task is the domain ontology, related to online job offers domain. This ontology not only extracts information (terms or concepts) from a potential job offer text but also performs classification of a web document as a potential job offer.

The first step in specifying the terminology of an ontology is to identify the objects in the domain of discourse(40). We identify the objects or concepts for the ontology by analyzing more than 5000 job offers and excavate required domain concepts for the ontology. We developed domain ontology for job offers into two independent ontology components. One ontology component, for generic job offers classification, and the second component is for more specific job offers classification. For specific jobs category we chose IT jobs offers, and used binary classification paradigm to categorize job offers into IT and non-IT job offers, and used it as case study to evaluate our classifier performance. We designated IT jobs as positive (*pos*) class, and non-IT job offers and any other text like news, article etc., as negative (*neg*) class. An ontology hierarchy or taxonomy is formalized with the most generic concept at the top and more specific to follow, and that hierarchical

| | |
|---|---|
| www.indeed.com | www.linkup.com |
| www.glassdoor.com | www.dice.com |
| careerbuildercareers.com | www.simplyhired.com |
| it-computer.jobs.net | www.jobserve.com |
| www.jobsradar.com | www.careerbuilder.com |
| www.itjobs.com | http://www.workabroad.ph |

Table 1.1: Different Sources of job offers for data acquisition

structure is used for job offers classification. To organize conceptualization into taxonomy it is necessary to establish a logical relation between different job offers concepts, e.g. 'job' *has-a* 'requirements', 'qualification' etc; IT 'job' *has-a* a 'software testing position', 'keywords' *has-a* 'Java' keyword, etc. Our multipurpose ontology helped to: extract concepts, classify job offers, and defined the minimum threshold for classification. In this whole procedure, we used our domain ontology, rather than any Machine Learning algorithms. In the classification procedure we applied only stop words removal preprocessing method, no other preprocessing methods are required to classify job offers through ontology based classifiers.

The internet and the World Wide Web contains tons of job opportunity, that need to explore through an automation system. For this automation, the most critical component is a classifier, which can categorize a job offer from other information. The automation of online job offers categorization and retrieval attributed to a classification system, that is plugged into a crawler, as job offer classifier. That combined system of crawler and classifier pave a way for the retrieval of job offers from the World Wide Web. The classifications mechanism we have developed are the prospective candidate for the job offer classification, to gain intelligence for job offers classification and retireval from the internet or the World Wide Web.

## 1.8 Dissertation organization

The rest of the dissertation is organized as follows.

**Chapter 2** , "Literature Review" contains the general and specific state of the art related to text classification and its application in recruitment process. It starts with the definition of text classification and further describes eight different machine learning text classifiers, we used for classification for job offers. Then machine learning state of the art for the job recruitment, recommendation and classification described. Then Ontology is defined generally. Then the different ontology based classification and information extraction, from Web pages, are described. Then the stat of the art, based on ontology, for the jobs recommendation, procurement of job offers method and system are described.

**Chapter 3** , "Machine Learning Classification Method" defined the data acquisition from different online job sites and a framework used to extract the main textual contents from these online job resources. This chapter further described different existing preprocessing method for text preprocessing and a new method of preprocessing called 'Domain Related Preprocessing? was also elaborated in this chapter. Based on this preprocessing how different dataset groups are organized for classification. Further different classifiers are listed for job offers classifications and classification model errors and regularization are described. Finally, evaluation method is described for generalization error for classifiers.

**Chapter 4** , "Ontology Based Classification Method" contains the information about the dataset of jobs offers, retrieved from different online sources. Then this explains the multiple functions of our proposed ontology. The construction of this ontology is elaborated in detail. This chapter also describes how the ontology is used to extract the job related information from the job offer text. Then we described the detail of ontology based classification model, from training the classification model and then put that model to the test.

**Chapter 5** , "Evaluation", contains the detail evaluation of Machine Learning based job offers classification. This evaluation gives the detail result of the sweet spot, to find optimal regularization value for eight classifiers. Analysis

of classifiers for generalization error and then provide the shortlisted classifiers, appropriate for job offers classification. Next, it evaluates the ontology based classification model for accuracy, precision, recall, and generalization for new data. A comparison of vocabulary extraction from IT and non-IT job offers is also discussed and analyzed.

**Chapter 6** , "Conclusion and future directions" summarize the dissertation and discuss the future work.

Chapter 2

# Literature Review

## 2.1 Text Classification

Text classification, or text categorization is the task of learning from documents $D$, which were already assigned with predefined categories (classes) $C$. In this learning procedure a best functions is searched, and a classification model is developed based on analysis of these provided document, and that classification model or classifier is capable of assigning a document $d_i$ with value T if this document fall under the category $c_j$, otherwise a value of F is assigned in case this document not belong to category $c_j$ (31)(27)(41)(42)(43). This classification model is learning with the provided dataset called training dataset. The classifier model developed based on training dataset is capable to predict a test instance with unknown class. This text document categorization or classification fall under the supervised learning paradigm, in which the document label or class is already defined(44).

### 2.1.1 Document representation and preprocessing

Purpose of text preprocessing is to make input documents more consistent for text representation; It is a mandatory step before text analytics process. This dimensionality reduction process is used to reduce the documents complexity by removing redundant features from that text. Primitive task of text preprocessing are *stop words* elimination and *stemming*. Stop words are more general words, used in a language and meaningless for classification or analysis process; Stemming

converts derived or inflicted words to a common root or stem. Preprocessing varies according to a domain. Some domain analysis required to retain its language syntax structure, and some required one or multiple preprocessing method before analysis or classification(45). Text preprocessing is a critical step for effectiveness of documents classification. Documents are represented in a specific format before classification. Text documents are typically represented with the help of most popular model called, vector space model.

### 2.1.2 Term Frequency - Inverse Document Frequency

The most commonly used method to weight terms or words in a document is based on vector space model called, **TF-IDF** (Term Frequency - Inverse Document Frequency (29)(46). This model represents documents as weight vectors, so that the documents with similar contents have similar vectors.

**TF** (term frequency) is the count of term occurrences, or frequency of a word, in a document. The **TF** assumes that the words with high frequency are more important than others(47)(48). The **TF** deals with the words frequency in an individual document, and cannot deals with the situation where a term appears in multiple documents or in whole corpus (total number of documents) . That is a drawback of term frequency (TF)(41).

**IDF** (inverse document frequency) weights the terms according to its importance with regards to the whole corpus, in other words **IDF** takes into account all the other documents to calculate a term weight. The intuition of **IDF** is that the most common terms in a corpus is not discriminative, e.g. stop words are the most common words in a corpus but not informative. The **IDF** scheme for weight a term **t**, in documents **D** is defined as,

$$\mathbf{IDF} = log(\frac{\mathbf{N}}{\mathbf{DF_t}}) \tag{2.1}$$

Where **N** is the total number of documents, and $\mathbf{DF_t}$ is the document frequency, where term **t** found(49)(48)(47).

To calculate the weight of the term in the vector space model, the word frequency count **TF**, is multiplied by **IDF**, the importance of the word in the whole

corpus. The equation to calculate this **TF-IDF** scheme is defined as follows,

$$\mathbf{TF-IDF} = (1 + \log \mathbf{TF}_{t,d}) \cdot \log \frac{N}{\mathbf{DF}_t} \tag{2.2}$$

Where, $\{\mathbf{d} \in \mathbf{D} : \mathbf{t} \in \mathbf{d}\}$

The intuition of **TF-IDF** weighting scheme is to assign higher weight to the term which has higher presence in a specific document as compared to other documents in a corpus, and make this document distinguishable. It decreases the significance of common terms in a corpus(49)(47).

### 2.1.3 Evaluation Methods for Classification

One of the important phases of the classification is evaluation; It determines the effectiveness of a classifier. The effectiveness is calibrated with precision, recall and F1 and accuracy. Precision is the fraction of relevant instances that are truly related to a class (true positive), this is given by the following equation,

$$\mathbf{Precision} = \frac{(\mathbf{True\ Positive})}{(\mathbf{True\ Positive}) + (\mathbf{False\ Positive})} \tag{2.3}$$

Recall, or, sensitivity is the fraction of relevant instances that are retrieved, its equation is given as follows,

$$\mathbf{Recall} = \frac{(\mathbf{True\ Positive})}{(\mathbf{True\ Positive}) + (\mathbf{False\ Negative})} \tag{2.4}$$

The F1 measure is the harmonic mean of precision and recall scores and its equation is given as follows,

$$\mathbf{F1\ Score} = 2 * \frac{\mathbf{Precision} * \mathbf{Recall}}{\mathbf{Precision} + \mathbf{Recall}} \tag{2.5}$$

**AUC** The Area Under the Curve, or, AUC is the average value of the performance of a classifier. A perfect classifier has AUC value 1.0 and random guessing has value 0.5. When comparing two different classifiers on the same dataset, we compare the AUC values for ranking.

### 2.1.4 kNN Classifiers

k-NN is one of the proximity based classifiers, that uses distance based measurements for classification purpose. The basic concepts of this classification is that, the documents belong to the same class are close to each other, when their similarity is calculated. For similarity distance *'cosine similarity'* or usually *'Euclidean Distance'* is used(50)(51). This classifier decides the category $c_i$ by calculating the similarity of $k$ number of documents, which are most similar to the document $d_i$, and if most of the documents similarity output is positive then these belong to category $c_i$, otherwise they belong to some other class(52)(53)(54)(31). The kNN is a *lazy learner* because during similarity calculation, between documents and $k$ training data, it stores feature vectors and classes of the training set, and when test documents are given then it rank similarity between training and test dataset(32)(42). There is no true training phase in kNN and major computation effort is in classification. In classification phase, distance between stored feature vectors and new test document is computed, and $k$ closest samples are selected and assigned most common class using majority vote(31)(55).

### 2.1.5 Bayesian Classifier

Bayesian classification method based on application of Bayes theorem (56)(57), and it belongs to the generative category of classifiers, that develop a probabilistic classifier based on words or features that are available in different classes. This classifier 'naïvely' assume that all these words or features are independent - without any relation to each other(28)(47)(32), and hence order of the features/words (in a language) become irrelevant, and presence of one word does not affect to others. Due to this naïve assumption the computation of Bayesian approaches is more efficient, and its classification accuracy is not much affected by this assumption(47)(32).

To explain the functionality of this supervised learning classifier, consider the binary classification scenario, (classification with only two classes, let's say positive and negative). The training data is processed into *'Bag of Words'* model - frequencies of terms are captured and saved and then presented into a form called vector.

Posterior probability of these *'Bag of Words'* features distribution is computed with the help of following equation

$$\mathbf{J(C)} = \prod_{k=1}^{m} P(W_k|C)P(C) \tag{2.6}$$

Where, *P(C)* is the prior probability of class $C$, and $P(W_k|C)$ is the likelihood of a word $W_k$ in a class $C$, that is estimated on a labeled training document dataset. A given document is then classified in a class that maximize *J(C)*. If values of *J(C)*, for all the classes are less than a give threshold, then the document is classified as negative.

The naïve Bayes classifier performance is extraordinary for some real world classification applications, such as text classification, email spam filtering, web contents categorization, marketing applications(58)(59)(60)(61)(62)(63). A prominent attribute of naïve Bayes classifier is that it uses small amount of training data to estimate the parameters necessary for classification. Although naïve Bayes has its advantages, however it is not free from problems, such as, when features are highly correlated then due to conditional independence assumption the classifier perform poorly. Another disadvantage of the naïve Bayes classifier is that, its accuracy, as compared to discriminative algorithm, such as Support Vector Machine (SVM) classifier, is lower, i.e. discriminative algorithm outperformed naïve Bayes classification accuracy.

### 2.1.6 Support Vector Machine (SVM)

One of the discriminative classifiers is Support Vector Machine (SVM) (56), it is based on Structural Risk Minimization principle of computational learning theory, which help to find a hypothesis with smallest true error - where, true error = (exact value) - (approximated value)(20). SVM classifier required dataset of at least two classes, lets say positive and negative, such that data points are linearly separable. An SVM model is a representation of the documents terms, mapped as points in space so that the terms of various categories or classes are divided by a clear gap of maximum width, called boundary width(41). The line which separate the

two datasets points is called *separating hyperplane*. If the data points are in two-dimension space then it is a simple line, but for n-dimensional space it is called *hyperplane*, that is the decision boundary for the classifier. The farther a data point from decision boundary more reliable the classification will. SVM classifier separating hyperplane is selected, such that, the distance between the point near hyperplane is maximum, this is known as *margin*. Bigger *margin* makes the SVM classifier more reliable. The points located closest to the hyperplane are known as *support vectors*(64).

The SVM classifier performance is distinguishable as compare to other classifiers(65)(47)(66)(67)(68)(69). The advantage of SVM is that, after training the prediction of new data only required dot product computation(47). According to Joachims (1998), SVM provides two distinct advantages for text classification because,

- SVC can scale up for n-dimensionalities with least tendency of overfitting for term selection
- Default parameter setting for SVM is enough for tuning on validation dataset, that also proved more effective in classification results

### 2.1.7 Stochastic Gradient Descent(SGD)

The Stochastic Gradient Descent (SGD) is one of stochastic approximation algorithms, it has a simple and efficient approach of discriminative learning classification, it is based on convex loss function from convex optimization discipline(70). SGD classifier uses gradient estimate to update the parameter, one instance at a time, that phenomenon is called *online learning*, because as soon as new data comes we update the classifier rather *batch processing* - all data processed at once for parameter update(71). The SGD classifier uses simple stochastic gradient descent learning procedure for classification, with the help of various loss functions and penalties - for regularization. The SGD classifier start with one instance and compute its gradient. This gradient is multiplied by learning rate, let's say 'alpha', and update classifier parameter. This process is repeated for all the instances available in a dataset, to find the optimized parameter for the classifier(64). Due

to this online learning attribute of SGD, it can apply for large dataset with large amount of features(72)(73). SGD has been successfully used in text classification and natural language processing(74). The main advantages of SGD classifier are its simplicity, efficiency, and simple implementation. The disadvantages of SGD classifier are: it required regularization parameter and number of iterations for classification, and in case of feature scaling its accuracy degrade because it is sensitive to feature scaling or preprocessing.

### 2.1.8 Perceptron Classifier

Perceptron belongs to discriminative classifiers category. It is a simple neural network that is capable to linearly separate data through on-line method. Perceptron was used for text classification by (75)(76), and subsequently used by (77) and (78). This classifier initializes by assigning the same positive weights to all of the words $W_i$ of a document $d_i$, which belongs to a category $c_i$ . Next Perceptron iterates over the training dataset, if the result is correctly classified then nothing is updated but if it encounters a misclassification then the weights of the classifier are changed. For example a document $d_n$ belong to positive class $c_+$, misclassified as negative, then the weight of the word $w_k$ is updated by adding a constant value called *learning rate*, on the contrary if document $d_n$ belong to negative class $c_-$ and misclassified as positive then the weight is updated by subtracting *learning rate* constant value(31).

### 2.1.9 Regression Methods for Text Classification

For text classification regression model is already used by (79)(80)(81)(75). Regression is the approximation of a real-values function $y = mx+c$, that fits the training dataset. For text classification, Linear Least-Squares Fit (LLSF) model of Regression methods is used in text classification by (54). The aim of the LLSF method is to learn the values of $x$ and $c$ such that the value of $y$ is minimized. The experimental results by (54) and (65) showed that LLSF is one of the most effective text classifiers. Although the computation cost of matrix manipulation during training is high(31).

**Logistic Regression** A better way of modeling the text classification is *logistic regression* model(82). *Logistic Regression* model provides a mechanism to use linear regression techniques for classification problem(48). In other words logistic regression is a linear classifier whose probability estimates calculated using sigmoid function(83). The sigmoid function is shown by the following equation,

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

The logistic regression classifier work as follows, all the features, along with their weight, are added. The result is used as input into the sigmoid function (equation 2.7), and the output value $\sigma(x)$ is between 0 and 1. A value above 0.5 is classified as positive and a value below 0.5 is classified as negative(64).

**Ridge Regression** In case of linear regression when the data is correlated then least squares estimates are free from biased but show high variance, and least-squares regression tends to overfit(84). The mechanism to overcome overfitting is call *Regularisation*, in which we penalize weight vector - by introducing bias to regression estimates, by apply some additional constraints to weight vector, to reduces the standard error. This kind of least squares regression is called *Ridge Regression*. *Ridge Regression* is one of penalized regression method and it showed good predictive accuracy(85). The bias that introduced, tend to reduce the regression coefficient or weight vector to small magnitude, that process is called *shrinkage*(64)(83). The challenging task in ridge regression is to find the appropriate shrinkage parameter to control the amount of shrinkage of regression coefficient, and reduce the tendency of overfitting for the classifier(86)

### 2.1.10 Random Forest

Random forest is an ensemble learning technique, first introduced by Leo Breman and Adele Culter(87). It is a hybrid model of Bagging algorithm (88) and random subspace method(89)(90)(91). Random forest uses decision tress to build its prediction model. Each tree is constructed from a random or bootstrap sample from the available dataset(92)(48)(34). An important benefit of it is that there is no need to prune the trees - as trees without pruning are prone to overfitting.

The basic concept behind random forest is that, individual tree is more prone to overfitting and if many trees are build which work well collectively then that method potentially decrease the amount of overfitting, by computing average of their results(93).

In Random forest every decision tree make its own prediction, and then for classification a majority vote method is used. In other words, each decision tree provides a probability for each possible output class label. Probabilities of these decision trees are then averaged, and highest probability is used to predict the class label(93)(94).

## 2.2 Ontology

The term ontology, a Greek word, originated from a branch of philosophy called metaphysics, it deals with the subject of existence of things, i.e. what categories of things exist and what are their relationships. The Ontology is three centuries old concept and in recent years it is introduced in computer science as machine readable vocabulary of terms with specification of their meaning and how they are related. According to Grober, "An Ontology is a *formal*, *explicit* specification of a *shared* Conceptualization"(39). According to this definition, an Ontology represents the area of knowledge of a certain domain in the form of vocabulary of terms. More specifically, conceptualization represent the abstract model of concepts (also known as classes or terms) that may exist in some certain domain of interest. From Grober definition, *Explicit* means that concepts or terms must be clearly defined in the form of properties and constraints of a domain of discourse. *Formal* refers to machine readability of these conceptualization which permits reasoning by computer. *Shared* mentions that knowledge gathered in the form of vocabulary can be used by people, computer applications and database that need the information related to same domain. So the basic building blocks of an Ontology is formed with the combination of five components: concepts (also known as classes - as classes in object oriented programming), terms (instances or objects of these classes), relations (categorization in the form of hierarchies - represents classes relations also known as properties), functions and axioms [Grober 1993(39)]. Concepts or classes are categories which are organized in the form of taxonomy

or hierarchy according to presumed natural relationship, like inheritance in object oriented programming. Ontologies encapsulate knowledge of a specific domain and this knowledge may further be shared and reused. The attribute '*shared*' mention a common understanding of information among software or people. Shareability and reusability make ontology a powerful tool to represent domain knowledge (39).

### 2.2.1 Conceptualization

To find the concepts and relations for ontology, in a certain domain, required careful analysis and study of that domain of interest. This analysis guides to find out the required conceptualization and relations for that ontology. These concepts and relations are defined unambiguously and designate each concept a related term.

### 2.2.2 Categorization/Taxonomy

An ontology hierarchy is developed with the help of most generic concept at the top, and more specific to follow, like a tree structure. For examples the most generic concept, for example, considered here are jobOffers, 'ITJobs', 'Computer-ScicneJobs' etc. To formalize conceptualization in to taxonomy it is necessary to establish a logical relation between sub areas of job offers. For Ontology development different relations types are available like 'has-a', 'is-a' relationship. For example, software developer 'is-a' ComputerSicenJobs, software developer 'has-Skill' of java language etc.

### 2.2.3 Axioms and Constraints

Axioms are the rules and/or constraints on the concepts, attributes, relationship and attribute values of an ontology. With the help of Axioms and constraint the meaning of the terms are defined in an Ontology. Hence axioms plays two basic roles in the construction of an ontology

1. First it represent the intended meaning of concepts gathered in an Ontology
2. Second it verify the consistency of the ontology

In short Axioms are small number of rules which are represented in a declarative form and can derive all the facts from them. Axioms verify the consistency of the ontology. Axioms are also very useful to infer new knowledge(95)(96). Following are the few examples of axioms, available in Semantic Web language, are as follows,

- rdfs:domain
- rdfs:range
- rdfs:subClassOf
- rdfs:subPropertyOf
- owl:inverseOf

Constraints are also important part in ontology construction because they help to validate and re-validate the data.

### 2.2.4 Information Extraction and Ontology

According to Riloff information extraction is the process of retrieving a specific text from a given Text corpus (97). Ontology defines the different concepts related to a domain, the instances of the Ontology are used to guide the information extraction process. Information extraction automates the natural language processing by trying to retrieve specific information from natural language text corpus. Ontology based information extraction emerged as a sub-field of information extraction and it guides the process of information extraction from text documents. Domain ontologies are defined for a specific domain of interest and the information extraction (IE) process is also responsible to address a specific domain to retrieve information. This specific domain Ontology is carefully crafted by a competent person, which covers the target data to extract, and the resultant output is also used as feedback to improve the Ontology for more accurate extraction process (98). So due to this common attribute, domain Ontologies can guide the information extraction process to retrieve the information like classes, properties, and instances from text corpus like, a web page or a text document. Ontology based extraction aims to retrieve occurrences of particular class of objects or events and occurrences of relationship among them (99).

## 2.3 Machine Learning State of the Art

The detail discussion of text classification, preprocessing methods, classification algorithms, and evaluation methods are available in (31)(20)(28). (100) discuss the detail of four classifiers for automatic text categorization in different fields, these classifiers are Naïve Bayes, k Nearest Neighbor (kNN), Support Vector Machine (SVM), Neural Network. A comparison of performance evaluation of these algorithms for test documents 'Reuters-21578' is given in this work with elaboration of different advantages and disadvantages of these classifiers. This section describes research which has been carried out by people in the area of automatic document classification with the help of machine learning methods, for job offers domain.

### 2.3.1 Job title classification

An e-commerce web domain system, having human resource data of 'Big Data' proportion, needed to categorize its thousands of job categories to make a metadata or catalog or categorization, to better facilitate customers for searching and browsing, and attract more customer to get better hit for the website to improve their business. With these intentions (101)(102) offer up their work to classify job titles into different categories, for already available job offers on the website of "CareerBuilder"[1]. This categorization serves two main purpose. With the categorization of this large data into predefined job offer titles: facilitate the employer with improved search and products recommendation system, facilitate candidates to easily find the organizations which are offering jobs, and vice versa. The classification and aggregation of this large data into metadata or categories makes easier for analyzers to do analysis.

This research work described a method of job title classification through semi-supervised learning method, with the help of taxonomy called O*NET Standard Occupational Classification(SOC)[2] or simply O*NET. This taxonomy has four level hierarchy with 23 major occupational groups on top level and 97 minor

[1] https://www.careerbuilder.com/
[2] https://www.bls.gov/soc/major_groups.htm

group on next level and so on. Classification with O*NET worked by discovering and then classifying job titles by classifier cascading - used multiple classifier for classification. First, unsupervised learning approach, clustering, is applied to distinguish job titles, out of job dataset. This method used a clustering algorithm to aggregate the training dataset into clusters with the help of Lingo3D - a library based on Lingo clustering algorithm (103). With the application of singular value decomposition (SVD) on the TF-IDF vectorization (46). Documents are arranged into clusters by applying cosine similarity. The k-NN algorithm is used as multi class classifier to classify these clusters and make a catalog or meta documents for further classification. Next support vector machine (SVM) is used to classify jobs to its top level SOC group. The evaluation of this method produced the classification result with averaged precision, recall and F1 score, more than 97%, 94% and 96% respectively, with average classification effectiveness 89.92%.

### 2.3.2 Web text processing for job classification

Labour market has adopted the web services to facilitate both employers and job seeker to fulfill their needs. (104) proposed a method to classify online job offers onto the categories of an already available classifier called CP2011[1] . Different text classification algorithms were applied to classify web job offers dataset onto the ISAT classifier, CP2011. The accuracy of this approach is compared with an already conducted classification by experts of CRISP[2] Research Center. These domain experts retrieve online job offers to study job market activity and variations.

The method defined in this research work consists of feature extractions and then classification. Text was preprocessed by tokenization, converted to small cases, removal of html tags, stop words removal, numbers and extraneous words removal, and finally applied stemming for Italian language. Feature extraction is used to organize ISTAT job category into a pair, containing word and corresponding probability, calculated using job offer titles(105), (106), (107). Then classification task find the relevance between ISAT categories and terms extracted

[1] CP2011 is the Italian standard classifier for Occupations and it is based on the logic of the ISCO (International Standard Classification of Occupations).

[2] CRISP, Inter university Research Centre on Public Services, Italy www.crisp-org.it

from job offers, by using distance-based algorithms. The highest ranked category is selected using Levenshtein similarity metric and proceed for classification.

The dataset was collected from different web sources, and CRISP domain experts assigned classes or labels to these online job offers, these classes or categories were taken from CP2011. This manual classification of domain experts is used to evaluate the text classification, through machine learning approach. Taxonomy based rules are defined, and these rules are used in classification, to classify job offers text. This classification, classify terms used in online job ads and relate them to CP2011 categories. Online job offers titles and qualification codes, defined by domain expert of CRISP Research Centre, are used to train and test different machine learning classifiers. After preprocessing phase the text is vectorized (with the help of Vector Space Model). Two classifiers from Scikit-learn framework (in Python language) are used to conduct text classification; first classifier is Support Vector Machine(SVM) classifier, based on linear kernel LinearSVC, is used; and from neural networks, Perceptron classifier is used(74). Scikit-learn provides facility called grid search, to find the potential optimal parameters for classifiers, to achieve best classifier effectiveness.

### 2.3.3 Job Finding by Text Classification

Web pages are the rich source of textual information, but due to huge size of the web, it is tedious task to extract useful information manually. With this problem in mind the researchers has developed different method to assist search on the web. With this perspective in mind, (108) proposed a solution, through text classification to sort out the job related information. (108) classify job type and job information for different districts of Chinese provinces and cities, and this classification task is performed for disabled person to find job for their domain of interest. Job advertisement data was collected from an official website, specific for Chinese disabled person. There are different types of job in this website, that were collected and used for training and testing for the proposed model.

This research work adopted statistical machine translation methodology called log linear model(109). With two layered architectures, this cascading model consists of perceptron at the core level, and other part consists of language models.

Jobs location or place names, which are extracted from the web document, are used as class labels. The vector space model is used for documents representation and then proceed to classification, according to pre-defined classes. Bayes method is used for classification of the preprocessed text data.

### 2.3.4 Job categorization and recommendation for social network users

One of the challenging problem for organizations, with online job offers website, are to manage and recommend job offers in an efficient and optimized manner, to achieve improved results for job offers and potential candidates matching operation. To fulfill this purpose, (110) proposed a method to identify the relevant information related to a job offers on social network environment and leverage this information to recommend the appropriate candidate. This method is made and evaluated by two e-recruitment organizations, Mutiposting and Work4; Work4 deals with recruitment through Facebook, and Multiposting share job offers on recruitment websites like LinkedIn, Monster, etc. The common aim of these two organization is to offer an efficient recommender system for Facebook and LinkedIn users, and provide a system which help social network candidates to best match appropriate job offers. Additionally, they provided automatic job categorization, based on matching job ads against job categories, such that the social network users can also select and filter these recommendations manually.

Recommendation operation divided job offers text structure, if available, into different fields, like 'Work', 'Education', and 'Interests'. LinkedIn and Facebook users profile text is already structured into field, which helped to apply dimensionality reduction to the text, by eliminating redundant information. The social network user profile is fragmented into fields, and organized into vectors; for this purpose TF-IDF vectorization is used. The purpose of this user representation is to match the user profile with a job offer by applying some similarity algorithm.

The evaluation of the proposed model is performed on three datasets. First dataset contains the social network user, which are candidate for the job offers. Second dataset is the reviews of potential matches, suggest by the developed recommender system and validated by two Work4 teams. Third dataset is the validation

dataset contains potential matches suggested by developed recommender system and validated by one of the team of Work4. Fourth dataset is the combination of last three datasets. The last three datasets are used for performance evaluation of job offer recommendation system. Another dataset which is used for model evaluation called Categorization dataset, it is the combination of job category description and job offer. Job category descriptions is defined in ROME nomenclature, that is equivalent of O*NET(102)(101), but in French language. The proposed method in this research used *field to field* similarity for matching; this *field to field* similarity is computed with the help of cosine similarity(111). According to this method, a field is compared independently to another field. This *field to field* similarity also helped for limited feature selection from the dataset, and hence achieve lower computational time. The drawback of this model is that it required structured documents for recommendation, and hence its scope is limited to only those job offers web pages which have structured text data.

The scope of the above defined system is limited to certain organizations only. These systems already have job offers on their websites and all these systems, except(108), rely on a competence taxonomy, and if that taxonomy is out of date, or a specific job offer description missing then the system may not perform as required. Compared to these systems our approach is to retrieve English language job offers and our scope is World Wide Web, and we do not need any taxonomy for classification, and our procedure produced better results as compared to (108). For the trail, we applied our method to IT and non-IT job offers only, and our job offers classification results are promising to pave a way for automation of job offers retrieval.

## 2.4 Ontology Based State of the Art

Human resource management procedure exploited the internet and the World Wide Web technology to improve recruitment procedure, and trying to recruit the most suitable candidate for a job, that is offered by a specific organization. Job offers classification is a research area for some time now. The mostly classification is

based on recommendation of a job offer to a suitable candidate and for this classification procedure we required either machine learning approach or ontology based or combination of both. Ontology based classification is comparatively a new area of research than pure machine learning classification approach. Ontology is used to represent the domain knowledge of a specific domain and further used by applications or humans to classify domain related tasks. The following section describes research which has been carried out by people in the area of automatic document classification, related to job offers and other domains.

### 2.4.1 Web Pages classification with Domain Ontology

With billions of electronic documents or web pages on the internet, it is impossible to classify specific information from these web pages manually. To sort out this problem, (15) proposed document classification method based on an ontology. Web page are basic unit of a Web site and these web pages consists of html based text with one or multiple subjects. Moreover, these web pages are connected internally or externally and made Bow-Tie structure(112). With this structure in mind, (15) considers that information can be easily extracted from these web sites by calculating similarities(113) of terms from ontology.

This research work used ontology to classify the documents related to: cooperatives, employment, finance, marketing, organizations and trade. Ontology contains vocabulary or concepts and relationship, organized in hierarchical manner, and classification can be achieved by comparing the vocabulary and relationship of ontology. Ontology is constructed based on the information text from web pages. Ontology based classification has some advantages: it provides machine reasoning, vocabulary in the ontology is not only collection of words but also have inherent semantics, ontology is representative of both web page and class.

This study proposed a potential method to build domain ontology semi automatically. First, frequently appeared vocabulary in the document collection is used to build the basic structure of the ontology. Further, more vocabulary is added, to establish relationship between concepts in the ontology. Then for classification, similarities are measured between terminologies (that were extracted

from the webpages) and ontology vocabulary. The classification procedure is conducted in two steps. First, find and extract the key vocabulary in the documents, and then map these vocabulary or concepts in taxonomy (concept or vocabulary hierarchy in an ontology). Before vocabulary extraction, documents are preprocessed with, stop words removal procedure and stemming; and vector space model is used to represent documents, by measuring weight vector with TF-IDF. The text extracted from documents is mapped onto an ontology node, after calculating similarity. The text which produced highest similarity values proceed for the classification, and categorized into a class. For accurate document classification, 'is-a', 'has-a', 'part-of', or 'has-part' relations are used. In case new node is added and its relation established, then its similarity is also calculated for classification. The performance of this model is compared and tested with Bayesian and TD-IDF classifier(114). The ontology based classification effectiveness, precision, recall and F1 were observed and these values are, 89.7%, 95.5% and 92.4%, respectively.

### 2.4.2 Design, implementation and evaluation of ontology based classification of web pages

Due to enormous amount of information on the internet, some search engines, like Yahoo, provides manual classification facility to the users, to access information. To reach out some specific information user has to use some automation system(115). The study in (16) address this problem by improving the accuracy of an automatic classifier, known as Automatic Classification Engine (ACE) (116). This improvement procedure used ontologies to achieve its task. The ontologies were developed with the help of Dewey Decimal Classification (DDC) (117) and Library of Congress Classification (LCC) schemes (118). Domain ontologies are used in this study to assist classification. Ontologies were built with the context of text information from web pages, and classification schemes of DCC and LCC. Domain ontologies(119) were developed with respect to classification schemes and then mapped into the associated classification schemes class representatives. Domain ontology is built in four steps. First, mapping is defined between class representation of DDC and LCC. Next an ontology is defined, that contains set of

"shared classes". The concept of "shared classes" is to create affiliation between domain ontology and classification schemes. Third, domain ontologies are manually built which are related to class representatives. Fourth, redundancy in ontologies terminologies is removed with the help of FaCT(120) reasoner.

Initially ACE creates a dynamic table called "terms table" with four columns, 'name', 'weight', 'tab' and 'position'. This table store terms extracted from web pages. The column, 'name', store the name of the term, 'weight' store the weight of a term, 'tag' store the meta-keyword tag, like html tag, where term was found. The column 'position' store the position of the term. Next, ACE generates a set of triples(121) in RDF, after validating ontologies syntax, and based on these triplets ACE builds domain ontologies (122).

A feed-forward network(122) is used for classification, and to establish relationship between class representatives, shared classes, and conceptual instances. This feed-forward network model contains three layer. Input layer is for input conceptual instances, hidden layer represents shared classes, third layer is the output layer that represent DDC and LCC. The classification process is fragmented into five steps. Initially, the terms are extracted from web pages and assigned a corresponding weight. A higher weight is assigned to a term which is present in the title, heading and in meta keywords, otherwise a small weight is assigned to that term, so that ACE choose the most significant term. Next, ACE compares its domain ontologies conceptual instances with conceptual instance from a web page, these web page conceptual instances are identified by ACE through a method called non-syntactic phrase indexing(123). When a conceptual instance of ACE domain ontology and web page is matched then that conceptual instance is assignment weight by ACE. In third step ACE normalize the weight of the significant instance, with the help of Euclidean norm(124). Next, ACE searches the affiliated class representative with the help of feed forward neural network. This neural network also used by classifier to find the similarity between class representative and a web page. In the last step of classification, metadata is generated. This metadata consists of normalized significant terms and corresponding class representative of DDC and LLC. This metadata is stored in XML based DBMS(125). This domain ontology based classification was conducted to improve the previous classier (116) and the results showed the improved accuracy as compared to (116).

### 2.4.3 An ontology-based recruitment system

Currently, there is no tool available to provide intelligent matching of the online job offer advertisements and employee curriculum vitae. To fill this gap, (36) proposed a method for intelligent recruitment, based on an ontology. This intelligent recruitment system developed an intelligent web portal to serve the needs of employee and employer. The semantic web consists of machine readable data, and technology used for this is the ontology. In this research, ontology is used to introduce semantic capabilities, and an ontological component is used to define search engine that provides more accurate matches between job offers and candidate profile.

Web portals are useful tool to collect huge volume of information. There are several employments provider web portals already working, but these web portals are not harvesting the benefits that semantic technology provides. So in this research work, an ontology based web portal is developed.

This web portal facilitate an ontology based semantic matching between job offers and their corresponding requests. The web portal is already providing services to the labor market in south east of Spain.

Ontologies can help to improve the search accuracy on the web, because, search method specifically find those pages which were referred by the concepts of ontology, and hence retrieve most relevant result. There are two methods available for the development of semantic web portals. First method is based on (126), it is a community web portals based on ontology for, not only accessing and contributing information but also maintaining and developing the portal. Another method is based on Ontoportal(127), that used ontology for the research community portal, to search for queries, and also add the new concepts into that portal. In this work, ontology is used as domain knowledge conceptualization (128). Moreover, ontologies are defined with the help of attributes and these attributes has some values. The concepts relate to each other in the ontology such that taxonomy structure or a tree like structure is developed, just like multiple inheritance in object oriented concepts. The axioms are defined for the ontology concepts relations. Axioms are the constraints or rules between concepts attributes and relationships.

The main purpose of this research work was to develop a web site, to provide easy to access services related to the employment search; for the people living in south east Spain city, Murcia. This website provides unemployed people a facility to search and find a job, and also facilitate enterprises to find specialized skills employees. For this purpose, portal offer advices to the jobless people and provides them the organizations contact - for job procurement. The additional features of this website is that, it provides public access to the people looking for employment, and private enterprise access to organization - to upload employment offer advertisements. The website also contain employment related news, and can also keep record of the applicant profile data and their curriculum vitae. This website is maintained in such a way that profile of employer and employee has common descriptors, which leads to a better match between applicant and employer. A potential employee can also visit the website to find interesting links, that were categorized into groups for easy access. The ontology for this website contain four main concepts, '*Applicant*', '*Employer*', '*offer*' and '*profile*'. '*Applicant*', is the potential employee and '*Employer*' is the organization or enterprise, who is offering job and looking for specific workers. '*offer*' is the job offer presented by '*Employer*' and '*profile*' is the '*Applicant*' qualification data.

### 2.4.4 Recruitment Process based on Ontology

The internet has proven an effective source for Human resource discovery and management. Different efforts have been done to increase employment market transparency, and improved employment procurement. Even high investment could not achieve the required goals till now. Moreover, the redundancy in the data quality produces unappropriated results for recruitment process. With these problems in mind, (37) tried to improve these recruitment problems with the help of Semantic Web technology.

A recruitment process is roughly divided into four main tasks, start with the development of job position requirements, then publish job offer online, and wait for applications from potential employees, and conclude with the selection of required skilled employee(s). These online job offers are usually consists of unstructured,

semi-structured, or structured text with uncontrolled vocabulary. Semantic technology provides the *concepts* from controlled vocabulary; which leads to machine processable data, improved job offer description that is independent of language, and effective matching of job offer and potential candidate.

There are many online job portals available, in public or commercial capacity, where an applicant searches for a job. It is tedious task to search job related information on these portals. Website with semantic technology, which publish their employment information in RDF(129) format, has several advantages. That website is easy to crawl by web spiders, jobs offers reach to more potential candidates, and hence procurement of job is easy. Both employment candidates and employer benefited from available semantic job description, and the information available in machine processable format. Moreover, organization can find best match between job requirements and potential employee.

To develop this system an ontology for job recruitment is created with the help of already available human resource ontology called HR-XML, based on the German version of HR-XML standards by HR-XML consortium[1]. Ontology is constructed in fragments (sub-ontologies), to avoid redundancy for job description and posting. The HR-XML library consists of more than 75 XML schemes for various human resource transactions. For this research work seven sub-ontologies were selected, from German version of HR-XML; these sub-ontologies are '*Job Position Seeker*', '*Job Position Posting*', '*Industry*', '*Organization*', '*Skills*', '*Education*' and '*Person*'. Due to internet, job market has expanded worldwide, so '*Industry*' sub-ontology specify both German[2] and American[3] classification system of Industry. '*Skills*' sub-ontology contains concepts to specify job requirements and candidate skills. '*Person*' sub-ontology is used for potential job candidate profile data manipulation, and '*Organization*' is used for organizations which offer jobs. HR-Ontology is defined in a Semantic Web language, OWL(130), and job posting and data of applicant are stored in another semantic web language, RDF(129). With the help of ontology, the degree of semantic similarity between employer and employee are calculated, and the output of it, is a ranked list of the best matched

[1] http://www.hr-xml.org/
[2] http://www.destatis.de/allg/d/klassif/wz2003.htm
[3] http://www.census.gov/epcd/www/naics.html

candidates for a specific job.

The state of the art, described above, are mostly deals with the job recommendation system, either on social network or a job offers websites. These organizations deal with the job offers which are published on their websites or they have access permission, and website job offers resource are easily available for analysis. The mostly work done based on job offer or job advertisement matching with most appropriate curriculum vita, to sever a specific organization needs. The scope of this research work is limited to an organization level, and there is no research work which describe the classification of job offers for bigger canvas, like the internet and the World Wide Web job offers, as there is huge amount of job offers possibilities available world wide. We tried to fill this gap by developing a classification mechanism with the help of ontology to classify job offer for generic and more specific job offer. Our system is flexible enough to expand without changing already defined system. As, for more specific human resource classification, all we need to do is to define a new ontology, and system need to train only for that new ontology, and it start working without showing any dependence on already defined system. Our classification system classify in two main steps, information extraction through ontology, and then classify that information as potential job offer, or not a job offer. This simple solution development is easily maintainable and scalable, with less effort.

Chapter 3

# Machine Learning Classification Method

We worked on binary classification solution and divided job offers into two main classes, IT related job offers as *positive* class and anything else (a non-IT job offer,a news, or a text article) as *negative* class. We evaluated eight classifiers on the text data, which is collected from 12 different online job sources, related to IT and non-IT job offers, and some different text articles from various news websites. The text is extracted from these web pages and saved into files for preprocessing. We apply different stemming methods or combination of these methods on this data to study classification results. These preprocessing methods are taken from the Natural Language processing (NLP) framework (NLTK), and a custom method is also introduced. The NLP framework provides two kinds of methods for stemming, one is called *'Stemming'* and other is called *'Lemmatization'*. The custom method that we developed, based on some common patterns, which are found in job offers text. These common patterns consist of auxiliary information related to jobs, and their elimination has no effect in classification, rather it helped to reduce the data size and classification time and performance (Section 3.2.1). With the help of these preprocessing methods we organize our text data into six groups (Section 3.2.2). For every group these text files are transformed into a CSV file, after preprocessing. Every CSV file consists of jobs offer text and its corresponding class, *pos* (IT jobs) and *neg* (non-IT jobs), as shown in figure 3.1. In another process some IT specifi text is converged from different form of terms into one common format, e.g. `'C#'` is converged from `'C sharp'`,`'C-Sharp'`, `'c#'`, `'c #'`.

This convergence remove confusion for a classifier because a classifier treats the different expression of the same term as different keywords, which can degrade the classification outcome. This text in files undergoes vectorization (tokenization, counting, and normalization with the help of `TfidfVectorizer` method) before classification. We examined the classifiers with dataset groups and evaluated different metrics and estimated the generalization error to find out the best model for IT job offers classification. Figure-3.2 shows these different steps from data acquisition to classification.

| No. | 1: text<br>String | 2: @@class@@<br>Nominal |
|---|---|---|
| 1192 | Registered Nurse - RN -FT Days-Kindred Hospital Kansas... Job in Kansas City, MO \| | neg |
| 1193 | Retail Services - VP, Program Manager (Sears) Job in Elk Grove Village, IL \| LinkUp.com\n\nBachelor's | neg |
| 1194 | Non Clinical Professional - SENIOR SYSTEM ANALYST Jobs in Georgia at Gwinnett Medical | pos |
| 1195 | Non Clinical Professional - SENIOR SYSTEM ANALYST Jobs in Georgia at Gwinnett Medical | pos |

| No. | 1: text<br>String | 2: @@class@@<br>Nominal |
|---|---|---|
| 1192 | registered nurse rn registered nurse rn planning delivery direct indirect patient care nursing process | neg |
| 1193 | retail services vp program manager sears job elk grove village il strong working knowledge card industry | neg |
| 1194 | clinical professional senior system analyst jobs georgia gwinnett medical center job description | pos |
| 1195 | clinical professional senior system analyst jobs georgia gwinnett medical center job description seeking | pos |

Figure 3.1: Two samples of CSV files: (above) contain text after extracted from World Wide Web through boilerpipe, (below) text preprocessed with some preprocessing method. The CSV files consists of text and its corresponding class, *pos* (IT job) or *neg* (non-IT job)

## 3.1 Data acquisition

Different IT and non-IT job offers are retrieved from 12 different job offers websites, listed in table-1.1. The text is extracted from these web documents and saved into 5512 text files for preprocessing, we called it *raw data*. We used 'boilerpipe' library to extract the text from these different websites.

**boilerpipe** library is designed to extract the main textual content of a web page. It removes the elements like boilerplate, templates and advertisements contents, which are part of a web page(131).

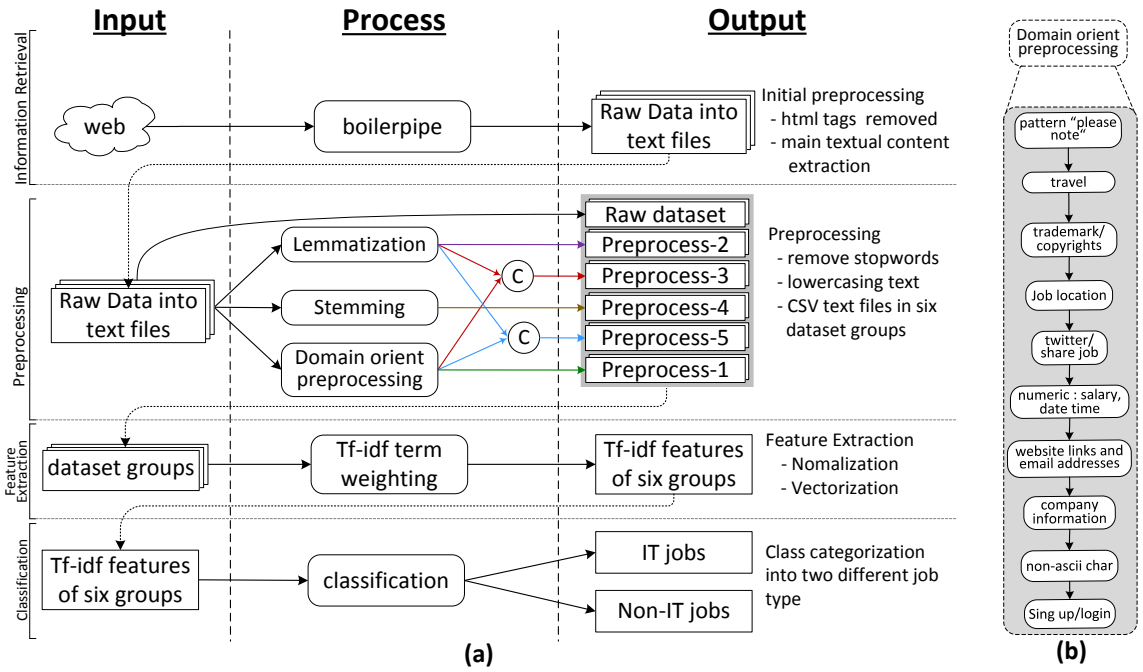


Figure 3.2: (a) The architecture of the process for text data preprocessing and IT jobs classification. It starts with the information retrieval from the web and then extracts the text with the help of boilerpipe framework. In preprocessing step total 6 groups are created, 5 groups with the help of different preprocessing methods or combination of methods(circle with C), and one group consists of text extracted from boilerpipe (Raw dataset). Further, these dataset groups are processed with `TFidfVectorizer` and then classified. (b) is the detail of domain oriented preprocessing method.

Text that related to job offers also contain some irrelevant information that is part of a job offer, but does not contribute to classification. For example 'ID:436457', the numeric value of 'Salary' or 'date time'. This information is removed by the custom preprocessing method, explained in Section 3.2.1. Although 'boilerpipe' removes the HTML tags but still some text documents consists of HTML tags - due to inconsistent HTML code of a web page. These tags are also removed with the help of this custom method.

## 3.2 Data Preprocessing

For data preprocessing we used Natural Language Processing toolkit (NLTK), for stemming and lemmatization and stop words removal, and then converted all the text into small cases. A Custom preprocessing method is used to remove some unnecessary information from job offers text, and we named this method 'domain related preprocessing'.

**Stemming and Lemmatization:** The difference between stemmer and lemmatization is that a stemmer operates on a single word without knowledge of language grammar or vocabulary, so the outcome is not necessary a valid English word. On the Contrary lemmatization process tags sentences into its part of speech (POS) with the help of WordNet lexical database of English, and then convert the inflected word into its related root, which is a valid dictionary word. That is why lemmatization is computationally much costlier than stemmer.

### 3.2.1 Domain related preprocessing:

Text documents D= $\{d_1, d_2, d_3, ..., d_n\}$, from a specific domain, consists of some extraneous information, which exclusion may reduce the text size and improves the classification results. This preprocessing method removes this unwanted text from jobs specific web documents. This text consist of few common text patterns that are observed from job offers text and are displayed as regular expression transition diagram, as shown in figure-3.3. These patterns do not contribute to classification and therefore eliminated from the text. Following is the detail of these patterns,

- **Pattern-1:** this pattern tries to find out words like 'please', 'e-mail' 'CV' etc. as shown in the figure-3.3(a)
- **Pattern-2:** deals with user invitation for a job and, usually text consists of words like 'click', 'apply' (for a job or role etc.), as shown by figure-3.3(e)
- **Pattern-3:** has text that invites users to subscribe the website or 'share' or 'log in', also copyright information, 'twitter' to twit the job information, as shown in the figure-3.3(b)

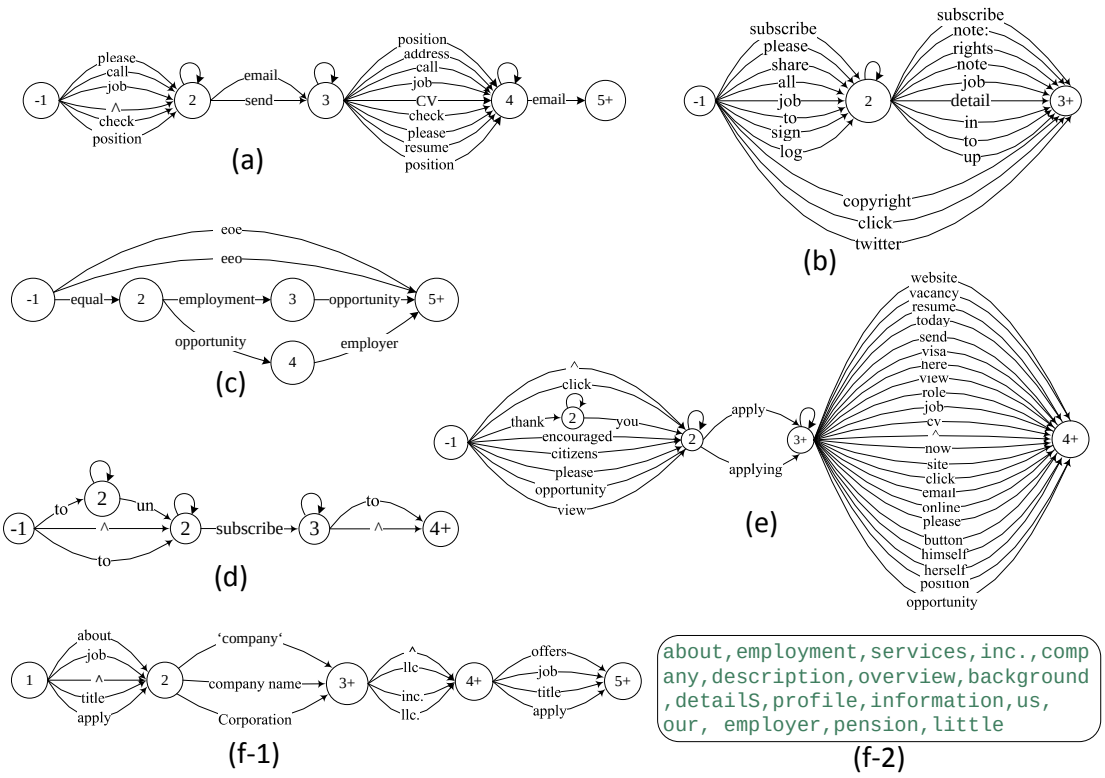


Figure 3.3: Different text patterns found in the IT and non-IT job offers, these common patterns are observed from different job offers and these are removed from the text

- **Pattern-4:** Text relates to organization policy and the environment. It format is like 'equal opportunity employee' or 'eoe'; and words like 'race', 'color', 'religion' and 'gender' helped to trace this pattern, diagram is shown in figure-3.3(c)

- **Pattern-5:** information related to 'subscribe' or 'unsubscribe' to the website, as shown in figure-3.3(d)

- **Pattern-6:** This text consists of organization introduction, which offer that job. This text describe the solutions they offer to their clients. It was observed that in some non-IT job offer, company introduction part has also mentioned about some IT related information, which may be misleading for

the classifier and need to remove. This pattern is shown in figure-3.3(f-1). Also a supplementary pattern, figure-3.3(f-2), of different specific words is used to detect the sentences/paragraph which consists of 'company description or introduction' (Listing 3.1 and 3.2)

Listing 3.1: Example 1

```
18  Copyright ©
19  IntraGroup, Inc. All rights reserved.
20  Reproduction in whole or in part in any form or medium
    without express written permission of IntraGroup,
    Inc. is prohibited.
```

Listing 3.2: Example 2

```
19  Company Information
20  HCL has a strong global presence with facilities in over 35
    countries. HCL has an extensive global network of delivery
    centers to provide seamless service to customers worldwide.
    Our unique culture of ideapreneurship encourages and empowers
    our employees to ideate and implement them generating value
    for the organization. Our 96,000 'ideapreneurs' (idea led
    intrapreneurs) have created a significant business impact by
    generating over 28,000 innovative ideas. This transformational
    culture seeds, nurtures and harvests, business-driven,
    customer-focused innovation at the grass roots level.
21  Dice Id : hcl001
```

'Boilerpipe' also helped to extract the title of a job-related web page, some of these titles consists of multiple information related to job title itself, company name and location etc. This information in title helped to extract the company name, if exists, and this information is further used to remove the traces of company/organization introduction (*C-Intro*) from the main text. Some documents contain the *C-Intro* text which was larger than the job description itself, and its elimination reduced text size such that after this preprocessing and removing stop words, the final dataset size was reduced to half of the original dataset. Example-1 (Listing 3.2) and Example-2 (Listing 3.2) are the two samples of this kind of textual

information. In these examples line 19 and 20 consists of company/organization related information.

The listing of Algorithm-1 shows domain related preprocessing method. First this method tries to find the company name from the title, with the help of pattern-6, usually company name contains postfix like 'LLC.' or 'inc.' etc., pattern-6 exploits this to find company name. If that name found then it is easy to find *C-Intro*, so method traverse a whole text document and try to find traces of this name in every line of the text of a document, and removes it. If the name is empty then pattern-6(f-1) and (f-2) tries to find *C-Intro* from the main text and remove this kind of text, which is shown in Example-3.1 and Example-3.2, by line 19 and 20. It is important to start domain related preprocessing with pattern-6, because if we apply any other pattern then there is a possibility that we lost some text related to the *C-Intro* information and unable to proceed with pattern-6. We initialized variable *keywords* after pattern-6 . These keywords are related to IT jobs and we extracted 134 of these keywords from IT jobs offers documents. These keywords aid the pattern-1 to 5 such that if text satisfies these pattern and also consists of one or more of these keywords then we preferred to keep that text otherwise remove it.

### 3.2.2 Dataset Groups

We organized the data into total six different groups. One group consists of *raw data* while other five are derived from this *raw data*, by applying different preprocessing methods or combination of them. This *raw data* consists of 5512 text files, that text was extracted from different job offer web pages, with the help of 'boilpipe' framework, and saved into text files. We used *lemmatization*, *stemming* and *domain related preprocessing* to the *raw data* and organize them into five groups. These different preprocessing methods took different amount of time to process text data. *Lemmatization* took about 60 minutes to process the 100 text documents. *Stemming* required approximately 12 seconds to process 1000 text documents and *domain related preprocessing* takes approximately 4 minutes for 1000 text documents. These groups helped to find out a specific preprocessing

**Algorithm 1:** Domain Related Preprocessing algorithm

```
1  AlgoLine0 Method: DomainRelatedPreprocessing(docs)
   Data: text documents
   Result: preprocessed document
2  initialization;
3  while till the end of docs list read next document d do
4      Intro ← GetIntroFromTitle(Pattern − 6);
5      if Intro not empty then
6          while not at end of document d do
7              read every line;
8              if line contain Intro then
9                  remove the line;
10     else
11         while not at end of document d do
12             read current line;
13             if contain 'about company' or
14             'company information' then
15                 remove this line;
16                 analyze next line/paragraph for 'company' information with
                    the help of pattern-6 (f-1) and (f-2) ;
17                 if found IntroFromJobTitle then
18                     remove the text;
19             else
20                 analyze the whole document for 'company' related
                    information;
21                 if found the infromation then
22                     remove it;
23             if trademark related information then
24                 Remove line;
```

**Algorithm 1:** Domain Related Preprocessing algorithm (continue...)

```
1  AlgoLine24
2  keywords ← [ tcp, sql, asp, soa, wan, php, qtp, san, pig,
     web services, web developer, web scripting, ios developer,
     ibm mainframe, visual studio, data modelling, ux designer,
     active directory, objective c, pair programming, z/VM,
     restful services, database analyst, Network engineer,
     Embedded software, Junit testing, Project manager,
     Software QA/SQA, .....];
3  while till the end of docs list read next document d do
4      while while not at end of this document d do
           /* We examine each line; if line consists of multiple
              sentences then we parse the line into different
              strings and then try to find pattern-1 to 5 in those
              string and remove only the string which consists of
              pattern-1 to 5 and save the other string to keep the
              potentially useful information safe.                */
5          read current line;
6          if contain pattern-1 to 5 then
7              analyze the line if found a keyword then
8                  skip the line to save text;
9              else
10                 remove line;
11         if found some rule in line then
12             apply that rule;
```

method or combination of them that can produce the most promising classification results. Following is the description of these data groups,

- **raw data:** dataset that consists of the extracted text from the web documents by 'boilerpipe'. This textual data is not treated with any preprocessing method and it is in its original form
- **Preprocessing-1:** To get this dataset we applied only *domain related preprocessing* on *raw data* (Section 3.2.1)
- **Preprocessing-2:** Data is *lemmatized* only with the help of NLTK 3.0
- **Preprocessing-3:** Data is *lemmatized* and also treated with Preprocessed-1 procedure
- **Preprocessing-4:** Data is *stemmed* only, with the help of porter *stemmer* of NLTK
- **Preprocessing-5:** Data is *stemmed* and also treated with Preprocessed-1 method

The data groups formation process is shown in figure 3.2(a)-'preprocessing' section

### 3.2.3 Rules to converge different IT related key terms into single expression

Some IT related terms are expressed in different forms and classifier also consider it different terms. These terms are converged into a common format by using some explicit rules, following are these rules,

**Rule-1:** `c-sharp, c #, c sharp, c#` → `C#`

**Rule-2:** `Dot Net,dot NET, .NET` → `.Net`

**Rule-3:** `Software development life cycle, system development life cycle, sdlc` → `SDLC`

**Rule-4:** `Object oriented programming, oop` → `OOP`

**Rule-5:** `Object oriented design, ood` → `OOD`

**Rule-6:** `Information technology, IT` → `information-technology`

## 3.3 Classification model

Each dataset group is divided into three parts for classification process, 60% is used to train the algorithm, and 20% is used for its validation - total 80% is used to prepare the model. The rest of 20% is new and unseen data for the algorithm, and it is used to test the model. Before classification, data groups are converted into 'bag of words' and then TF-IDF term weight is calculated (vectorization) with the help of Scikit-learn(74) `TfidfVectorizer` method, as shown in figure 3.2. For the experiment, we used Scikit-learn Python framework 0.17 with Python 2.7, on 64 bit quad core machine, having 8GB Ram, with windows 7 installed.

### 3.3.1 Classification Setup

We used eight Scikit-learn framework classifiers named `LogisticRegression`, `RidgeClassifier`, `SVC` (SVM Classifier), `SGDClassifier`, `KNeighborsClassifier`, `MultinomialNB`, `RandomForestClassifier` and `Perceptron`. With 10-fold cross validation and optimal regularization $\lambda$ (Section 3.3.2), as classifiers parameters. We evaluated these classifiers on six dataset groups.

### 3.3.2 Model Errors and Regularization

Prediction estimators inherently have three types of generalization errors: bias, variance, and noise. Noise is an irreducible error. Bias of a learning algorithm indicates its average accuracy across different iteration of the training set. High bias (under-fitting) classifiers unable to capture the relevant relationship between train and validation data and cause high train and prediction error. The variance is the sensitivity of a learning classifier to minor alternation in the training dataset.

High variance (over-fitting) models predict well on training data but do not predict well on new data. A classifier with small variance and high bias underfit the model and the classifier with high variance and low bias overfits the model. Bias-variance trade off minimizes the two errors that prevent learning classifiers from generalizing beyond their training set. To overcome the high bias and variance we used regularization.

**Regularization** is a collection of techniques that helps to prevent over-fitting by penalizing the model complexity (132).

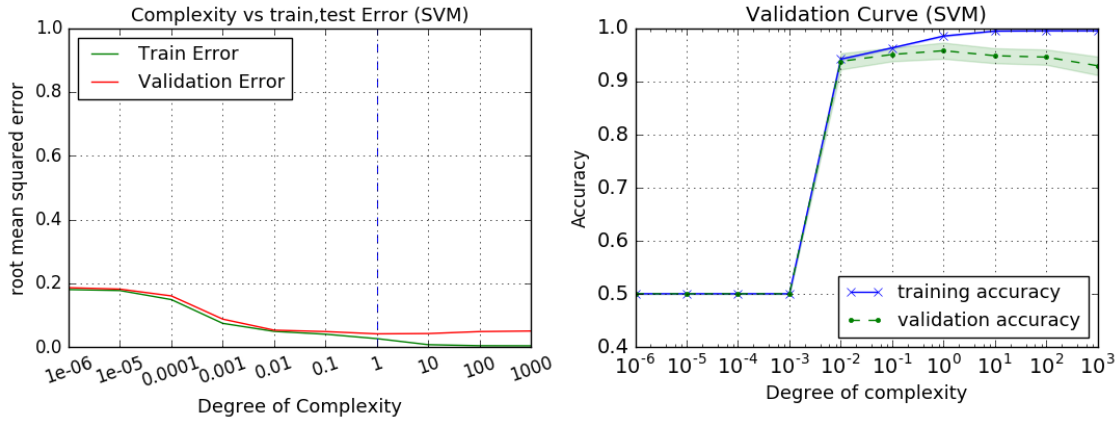


Figure 3.4: description about validation curve

To find out optimal regularization parameter we used Scikit-learn *validation curve* (validation graph), which helped to tune the classifier to find the potential optimal regularization value $\lambda$, to avoid the overfitting problem. We also plot another visualization *complexity vs train test error* (complexity graph). In this we plotted Root Mean Square Error (RMSE) as a function of *Degree of Complexity*. With the help of this visualization we tried to find out the optimal *Degree of Complexity* value $\lambda$, for all eight classifiers. An example of these visualization is shown in figure 3.4, where on the right is a validation graph and left is a complexity graph. In these plots we searched for *sweet spot*(**?** ), that is potential optimal $\lambda$ value ($\lambda$ = *Degree of Complexity*), to tune the classifier for best performance.

## 3.4 Evaluation Measures

To evaluate classifiers output we used Accuracy, Precision, Recall, F1 Score and Area Under the Curve (AUC) score. We evaluate classifiers in two steps. We called these two steps as *initial evaluation* and *generalization error evaluation*.

### 3.4.1 Initial Evaluation

Initially we had eight classifiers that used six dataset groups, that make total 48 experimentation. We observed the behavior of these tests with the help of *learning curve*. We observed two criterion on the *learning curve*, *Accuracy* must be $\geq$90%, and training and cross validation curves must converge, that implies we have appropriate dataset size to train algorithm (this plot is shown in figure-5.2[bottom right]). These criterion helped to select only 12 classifier and their corresponding dataset groups to further examine for generalization error.

### 3.4.2 Evaluation for Generalization error

We evaluated generalization problem with the help of *train test errors* plot (evaluation plot) to examine the classifier model for high-variance. In this curve, we plotted RMSE as a function of training and new data (test data) samples iteration (in %). This visualization helped to find out the generalization error of a model, when new data is introduced. If train and test data curves have high error rate and these curves diverge (figure-5.3[right]) then model over-fit on new data, if curves retain a low RMSE rate with varying size of train and test data then a model generalized well on the new data (figures 5.2 and figure 5.1).

Chapter 4

# Ontology Based Classification Method

We constructed ontology with job offers text files, which were already retrieved from the World Wide Web. We used ontology to extract the ontological concepts from our dataset and the output is concepts frequency, that is stored in MySQL database and with the help of this database we tagged ontology with certain ratio score. This tagged ontology is used to prepare classification model which is evaluated with the test dataset. This procedure work flow is shown in figure 4.1. The detail of all this procedure is elaborated as follows.

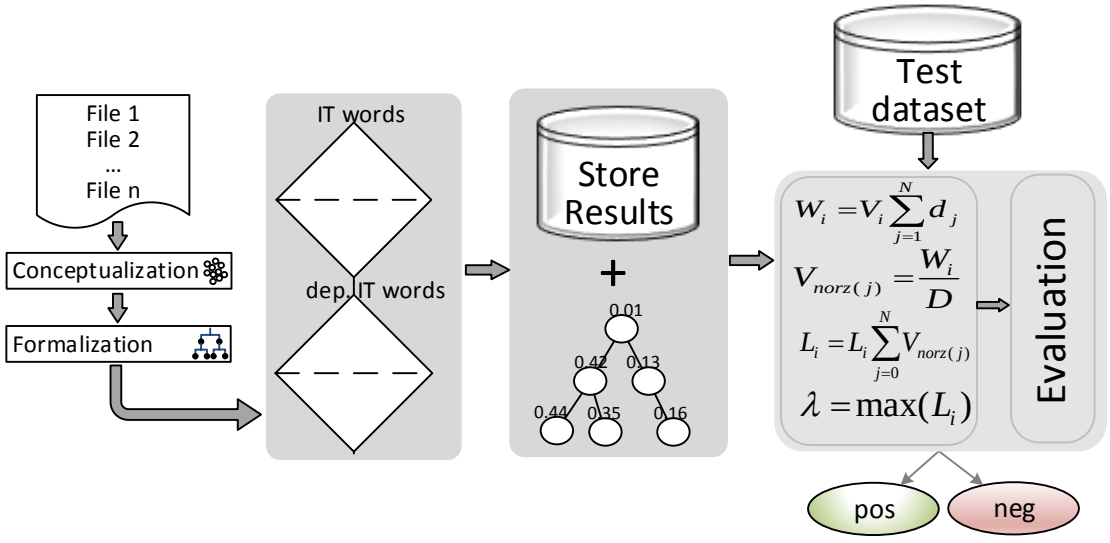


Figure 4.1: Architecture of system from ontology construction to classification

## 4.1 Data retrieval and preprocessing

We retrieved web documents from various job offer websites; like indeed[1], dice[2], careerbuilder[3] etc., and then preprocessed them with the help of boilerpipe framework(131). This framework extracted the main textual contents from these web pages. The boilerpipe removes all HTML tags and extract text from all these web pages and we saved this text into files. This dataset consists of two types of files, IT job offers related files, designated as *pos* class, and non-IT job offers (that also include different news and articles) as *neg* class. We simulate Machine Learning approach, i.e. first train and then test that classifier model. We used 80% data for training and 20% for testing. We used 4512 files, 2256 *pos*, and same amount of *neg* text files. For testing purpose we have 1000 text files dataset, 500 *pos* and 500 *neg*. We constructed ontology with the help of training dataset, this ontology is constructed into two part. One part is to classify general job offers (*pos* and *neg*) and other part is specialized for IT job offers (*pos*) classification. This ontology performs three main tasks,

- **Extraction:** Ontological concepts are used to extract the domain information from a file or batch of files
- **Minimum Classification Threshold:** Ontology helped to find a minimum threshold score, that is used to classify a generic job offer, or job offer as IT or non-IT
- **Classification** Ontology is used to develop a classification model. This procedure is equivalent to training an algorithm, in machine learning discipline

This developed classification model is used to classify an individual file or batch of files in a dataset. The classification model required to assigned a ratio score to each vocabulary of the ontology to classify IT job offers. This tagged ontology is used to calculate the minimum threshold score for classification of a job offer.

[1] www.indeed.com
[2] www.dice.com
[3] www.careerbuilder.com

## 4.2 Ontology

To construct an ontology we manually extracted some common keywords that are found in all job offers dataset. We selected random samples, between 100 to 200, to search for these common concepts - by manually reading the text files. We repeated this procedure for multiple times with random sampling and observed more of these common concepts. Next we searched these selected concepts to the whole dataset to find out how many documents contain these words and how significant is that specific vocabulary/concept, by observing their frequency. This procedure helped to explore more vocabulary. With the help of this iterative process we excavated the vocabulary for the ontology. Then we established a relationship between these concepts. For example, part one of this ontology we establish a relation such that 'job' *has-a* 'responsibilities', 'qualification', 'requirement', 'recruit', 'job_type', 'apply', 'benefits' and 'salary',all these nodes have further child nodes, as shown in figure 4.2. For IT job offers we established relationships between concepts like 'RestFul' is a 'web service', so we developed a relation such that web service *has-a* 'RestFull API', a partial ontological taxonomy structure is shown in figure-4.3. Root of both part of the ontology is named 'Job' that consists of different child nodes and these child nodes further have nodes. The 'keywords' node in IT ontology is a leaf node, this consists of words which are specific for IT job offers like 'Java', 'SQL', 'DBMS', 'data warehouse' etc. IT ontoloyg consists of total 177 concepts(including 'keywords'), the 'Keywords' concept further consists of 102 keywords. Ontoloy also deals with the IT specific synonym words, like 'TDD' synonym is 'Test Driven Development', 'OOP' synonym is 'object oriented programming' etc. So our ontology consists of 176 concepts, including synonym and then 102 keywords, so we have more than 279 IT vocabulary concepts.

## 4.3 Ontological Vocabulary extraction for Classification Model

To extract vocabulary from job offer text files, ontology concepts are mapped into its corresponding regular expressions. These regular expressions are used to

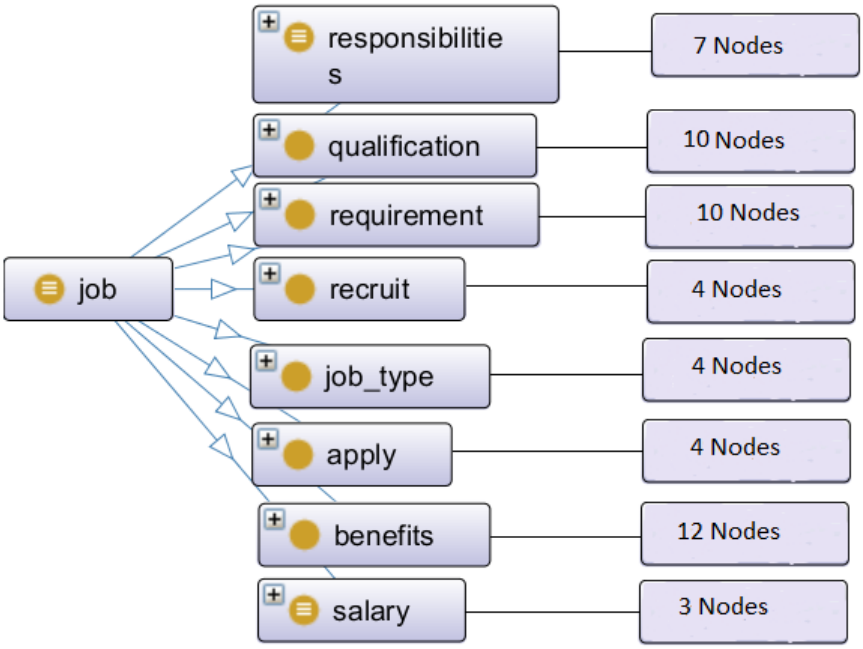


Figure 4.2: Ontology related to IT job offers with its hierarchy

extract the ontological concepts from the text files for general job offers and for IT and non-IT job offers. This extracted vocabulary count is saved into database. Each tuple of this database represents a job offers text file, and the columns of this tuple corresponds to vocabulary/concepts frequency that is found in that file. So we have total three tables in database one is for general job offers and second is for IT, and last for non-IT job offers. Every table is like a big sparse matrix of 4512 rows.

For second part of the ontology (for IT and non-IT job offers), we divided vocabulary into two part, 'Level-1' and 'Level-2'. Level-1 consists of 77 concepts that are specific to only IT job offers and has no presence in non-IT jobs. 'Level-2' concepts are common in both IT and non-IT job offers. Our extraction for IT job offers is conditional, such that 'Level-2' is extracted if and only if one or more of the 'Level-1' vocabulary has already been found. That way the common concepts are selected for only IT job offers and filtered out for the non-IT job offer.

Some concepts of vocabulary has significance presence in both IT and non-IT job offers but convey a very different meaning and purpose. For example

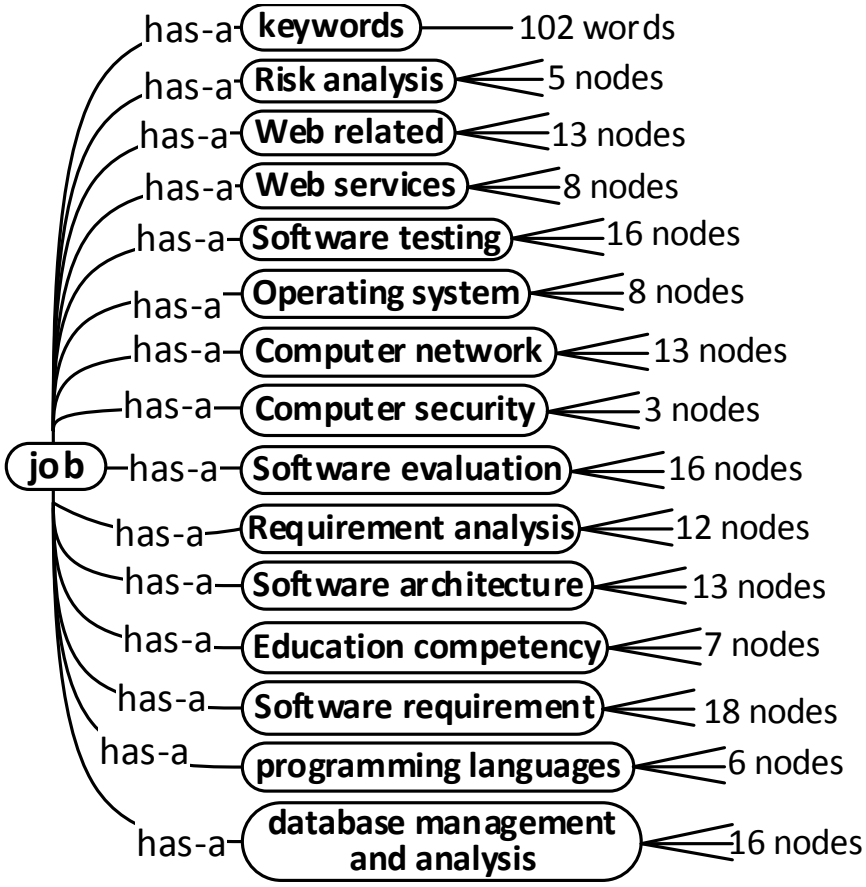


Figure 4.3: Ontology related to IT job offers with its hierarchy

'developer' or 'development' in IT job offer mean to develop an application into some specific language or framework but in non-IT job it showed some different purpose, e.g. a syntax in a non-IT job offer is "Develop electrical equipment specifications and electrical cost estimates", shows a very different perspective than an IT job offer does. But a careful analysis lead us some patterns which are found only IT job offer but did not found in non-IT job offers or its presence is so small that it has no significant effect on classification results. Three of these patterns are shown in figure 4.4, as regular expressions transition diagram. One of the Pattern in figure 4.4 shows the variation of 'develop' (development,developer), which is found in IT job offers. With the help of this pattern we successfully extracted this concept from IT job offers and excluded it for non-IT job offers.

Same is the case with 'cloud' and 'web' pattern, as shown in the figure 4.4.

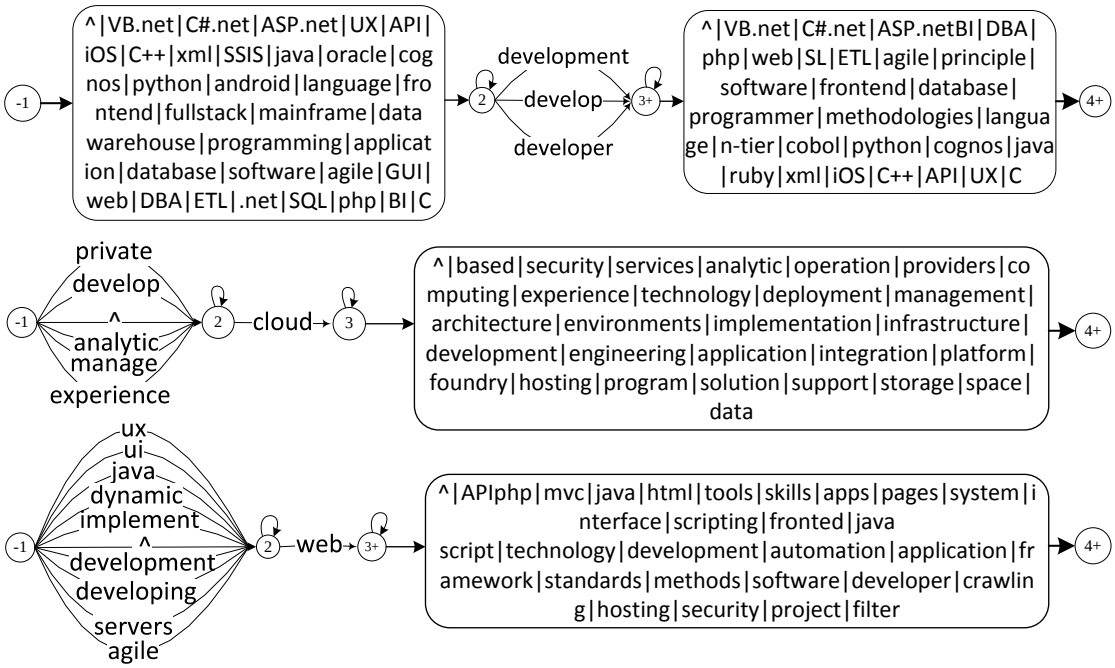


Figure 4.4: Regular Expressions to extract concepts 'development', 'cloud' and 'web' from IT job offers. Here '|' stands for or operator and '^' is for null transition

## 4.4 Architecture and implementation of an Ontology based Classification Model

To prepare an ontology based classification model we used both parts of the ontology, one for all types of job offers and other to classify IT and non-IT job offers - that part simulate the binary classification paradigm. This procedure is elaborated as follows,

1. **Reference Ontology:** To explain the construction of classification model we used an example ontology, as shown in figure 4.5(feature Vectorization) portion. This ontology consists of nine concepts or vocabulary $\{\mathbf{V}_0, \mathbf{V}_1, ..., \mathbf{V}_8\}$. $\mathbf{V}_0$ corresponds to 'job', $\mathbf{V}_1$ corresponds to 'experience' and vice versa.

2. **Feature Vectorization:** We extracted vocabulary from all available documents with the help of ontology (each concepts has corresponding regular expression) and converted it into numerical feature vectors. We started the process by counting the number of occurrences of a concept $\mathbf{V}_0$ in each document $\mathbf{D}$=$\{d_1, d_2, d_3, ..., d_m\}$, and then sum all these count for vocabulary $\mathbf{V}_0$ (For example if there are three documents in a dataset and $\mathbf{V}_0$ observed 5 time in $d_0$, 7 times in $d_1$ and 8 times in $d_2$, then total sum in all three document is 20). We repeated this process for all vocabulary from $\mathbf{V}_1$ to $\mathbf{V}_8$. This procedure can be defined with the following equation,

$$\mathbf{W}_{i(sum)} = \mathbf{V}_i \Big[ \sum_{j=1}^{N} (d_j) \Big] \tag{4.1}$$

Where:

$\mathbf{W}_{i(sum)}$= is the sum of Vocabulary $\mathbf{V}_i$

$D$ = total number of IT job offers documents

$i$ = {0,1,2,...,N}, here N=8

$j$ = {1,2,3,...,M}

This procedure is shown in figure- 4.5(Feature Vectorization) phase, along with the detail of vocabulary and its corresponding values in a table and ontology. The output shows numerical count of the features.

3. **Normalization:** In the next phase we normalized each vocabulary of the ontology by dividing it with total number of document **D** in the dataset. We denoted the normalized vocabulary by $\mathbf{V}_i$ and define its corresponding procedure as follows,

$$\mathbf{V}_i = \frac{\mathbf{W}_{i(sum)}}{size(dataset)} \tag{4.2}$$

This is shown in figure 4.5(Normalization) phase, along with the normalized features values as output.

4. **Minimum Threshold for classification:** In this phase we calculated the minimum threshold score to classify a job offer, as shown in figure 4.5(Minimum Threshold) part. In this procedure we split the ontology into sub taxonomies $\mathbf{T}_0, \mathbf{T}_1, ...\mathbf{T}_n$, by apply following simple rules.

   - We selected the root of the ontology as sub-taxonomy $\mathbf{T}_0$, if root has some synonyms or equivalent concepts, like 'job' may have synonym as 'work' or 'opportunity', then these concepts will also become part of this sub-taxonomy
   - Further if ontology root has some direct children then all these nodes and its descendant will become part of different sub-taxonomies, for example the root $\mathbf{V}_0$ of ontology, shown in figure 4.5, has three children, $\mathbf{V}_1, \mathbf{V}_4, \mathbf{V}_7$, all these three nodes along with their descendant become part of the sub-taxonomies named as $\mathbf{T}_1$, $\mathbf{T}_2$ and $\mathbf{T}_3$

   These sub-taxonomies are shown in the figure 4.5(Minimum Threshold) phase. Further we calculated the following values for these sub-taxonomies

$$\mathbf{T}_{0(sum)} = \sum_{i=0}^{0} \mathbf{V}_i \tag{4.3}$$

$$\mathbf{T}_{1(sum)} = \sum_{j=1}^{3} \mathbf{V}_j \tag{4.4}$$

$$\mathbf{T}_{2(sum)} = \sum_{k=4}^{6} \mathbf{V}_k \tag{4.5}$$

$$\mathbf{T}_{3(sum)} = \sum_{l=7}^{8} \mathbf{V}_l \tag{4.6}$$

$$\mathbf{T}_{sum} = \mathbf{T}_{0(sum)} + \mathbf{T}_{1(sum)} + \mathbf{T}_{2(sum)} + \mathbf{T}_{3(sum)} \tag{4.7}$$

$$\mathbf{T}_{i(norm)} = \mathbf{T}_{i(sum)} / \mathbf{T}_{sum} \tag{4.8}$$

Where

- $\mathbf{T}_{0(sum)}$ is equal to sum of $\mathbf{V}_0$ and its synonyms - if exists - for this sub-taxonomy
- $\mathbf{T}_{1(sum)}$ is equal to sum of $\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3$, values for this sub-taxonomy
- $\mathbf{T}_{2(sum)}$ is equal to sum of $\mathbf{V}_4, \mathbf{V}_5, \mathbf{V}_6$ values for this sub-taxonomy
- $\mathbf{T}_{3(sum)}$ is equal to sum of $\mathbf{V}_7, \mathbf{V}_8$ values for this sub-taxonomy
- $\mathbf{T}_{sum}$ is equal to sum of all the above calculated sub-taxonomies values
- $\mathbf{T}_{i(norm)}$ is normalized sub-taxonomy values, which is achieved by dividing all sub-taxonomy by $\mathbf{T}_{sum}$

With the help of above manipulation the minimum threshold $\lambda$ for classification can be defined as,

$$\lambda = \mathbf{max}\Big(\mathbf{T}_{i(norm)}\Big) \tag{4.9}$$

i.e. find maximum value of $\mathbf{T}_{i(norm)}$, that is our minimum threshold to classify individual text files that contain potential IT job offers (figure 4.5[Minimum Threshold] phase).

5. **Sub-taxonomy Vocabulary ratio score** We normalized vocabulary values for every sub-taxonomies and assigned that value to that vocabulary. Vocabulary normalization of a sub-taxonomy represents a ratio expressed as a fraction of normalized sub-taxonomy $\mathbf{T}_{i(norm)}$, as calculated in the previous step. Following are the expressions of vocabulary normalization of a sub-taxonomy.

$$\mathbf{V}_{i(norm)} = \frac{\sum_{i=0}^{0} \mathbf{V}_i}{\mathbf{T}_{0(norm)}} \tag{4.10}$$

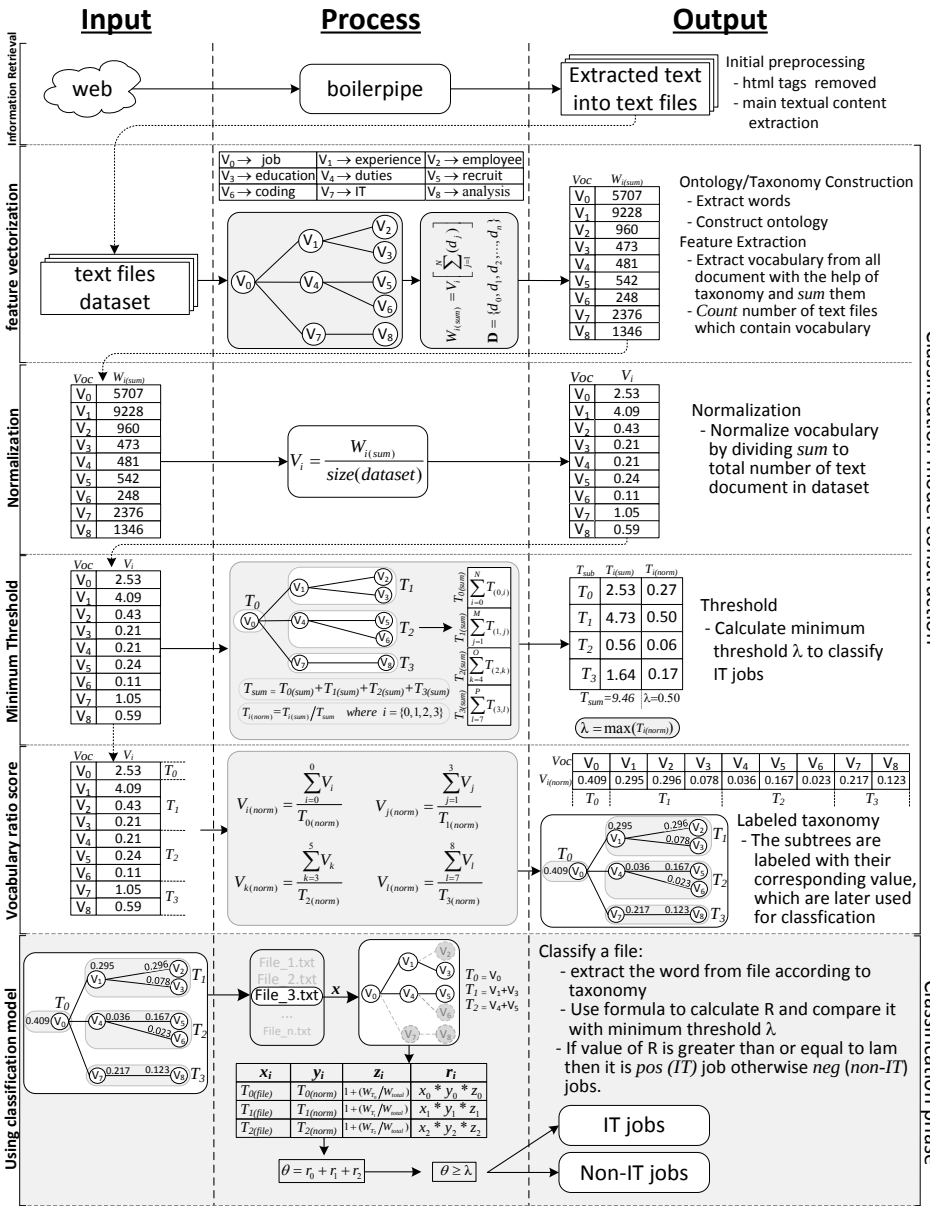


Figure 4.5: The architecture of the process for IT jobs classification with the help of ontology. Starting with the extraction of text from job offers websites with the help of boilerpipe framework. These text files dataset helped to construct an ontology. The vocabulary of this ontology is searched and counted in these text files to generate feature vectorization values. Then we normalize these vectorization. We Split ontology into sub-taxonomies to find the minimum threshold score for classification. We tagged the ontology by calculated vectorization features values. This labeled ontology and minimum threshold score made a classification model which helps to classify a potential job offer.

$$\mathbf{V}_{j(norm)} = \frac{\sum_{j=1}^{3} \mathbf{V}_j}{\mathbf{T}_{1(norm)}} \tag{4.11}$$

$$\mathbf{V}_{k(norm)} = \frac{\sum_{k=4}^{6} \mathbf{V}_k}{\mathbf{T}_{2(norm)}} \tag{4.12}$$

$$\mathbf{V}_{l(norm)} = \frac{\sum_{l=7}^{8} \mathbf{V}_l}{\mathbf{T}_{3(norm)}} \tag{4.13}$$

We have four sub-taxonomies and for each vocabulary of a sub-taxonomy we divided it by sub-taxonomy normalized sum $\mathbf{T}_{i(norm)}$(*where* $i$=0,1,2,3). For example every vocabulary of sub-taxonomy $\mathbf{T}_1$ is divided by normalized sum of $\mathbf{T}_{1(norm)}$. These values are tagged with all corresponding vocabulary of sub-taxonomy $\mathbf{T}_1$. This process is repeated for all other sub-taxonomies, as shown in figure 4.5(Vocabulary ration score) phase. This tagged ontology is used for classification of job offers.

## 4.5 Job Offers Classification

Previous section discussed in detail about the development of ontology based classification model. This section explains how to use this model to predict a job offer, or batch of job offers in a dataset - consists of job offers text files. For a job offer text file we extracted the corresponding vocabulary of labeled/tagged ontology, as shown in figure 4.5(Using classification model) input. It is not necessary that an individual file contains all the vocabulary of the ontology, as example is shown in figure 4.5(Using classification model) 'process' section, the grayed nodes are vocabulary not available in the job offer file 'File_3.txt'. Next we calculated the values of $\mathbf{T}_0$, $\mathbf{T}_1$, $\mathbf{T}_2$ and then get values of $x_i$, $y_i$ and $z_i$ (values are shown in table of figure 4.5[Using classification model] 'process' phase) and plugged these values

to the following equation-4.14, to get $\theta$.

$$\theta = \sum_{i=1}^{N} \left\{ \left(T_{i(file)}\right).\left(T_{i(norm)}\right).\left(1 + \frac{W_{T_i}\left(\sum_{j=0}^{M} V_j\right)}{W_{total}}\right) \right\} + CF_i \tag{4.14}$$

where

- $\mathbf{T}_{i(file)}$ is sub-taxonomy (figure 4.5 - Process) made out of input sub-taxonomy(figure 4.5 - Input), it consists of vocabulary that is found in input text file, while grayed nodes shows the vocabulary not available, as shown in figure 4.5(Using classification model) 'Process' part. There are three sub-taxonomy made out of this text file, $\mathbf{T}_0, \mathbf{T}_1, \mathbf{T}_2$ and these sub-taxonomies have vocabulary $\mathbf{V}_0$, $\mathbf{V}_1$, $\mathbf{V}_3$, $\mathbf{V}_4$ and $\mathbf{V}_5$

- $\mathbf{T}_{i(norm)}$ are the values that were calculated in 'Minimum Threshold' part of figure 4.5

- $\mathbf{W}_{T_i}$ is the sum of words count of all the vocabulary in sub-taxonomy $\mathbf{T}_i$, i.e. $\sum_{j=0}^{M} \mathbf{V}_j$, e.g. if $\mathbf{T}_1$ has vocabulary $\mathbf{V}_1$ and $\mathbf{V}_3$ and their frequencies are 3 and 5 respectively, then $\mathbf{W}_{T_i}$ is equal to their sum, 8

- $\mathbf{W}_{total}$ is total words found in a text file, these words are counted after removing stop words, non-ascii characters, and extra spaces

- $\mathbf{CF_i}$ is compensating factor for every file. which can be defined as $\mathbf{CF_i} = \frac{W_{C_i}}{\mathbf{W}_{total}} * \lambda$, where $W_{C_i}$ is the word count

Although ontological concepts are extracted from job offers dataset but it is not possible that one can get all the potential concepts. So the basic concept of this term $\mathbf{CF_i}$ is to compensate that potentially missed term(s). If a file contains detail description about a job offer then the division part reduced to almost zero value(because greater number of words count), on the contrary a file with small amount of words this factor produce bigger value. So a file with detail description of a job offer already surpass minimum threshold $\lambda$, and by adding this value does not improve classification procedure, but a file with small job description will take

advantage of this compensation and improves classification results (Section 5.2). The CF range is between 0 and 0.0398760638855

In equation 4.14, the 'File_3.txt' text file sub-taxonomy $\mathbf{T}_{i(file)}$ is the fraction of input sub-taxonomy $\mathbf{T}_{i(norm)}$, we calculated this by multiplying $\mathbf{T}_{i(file)}$ and $\mathbf{T}_{i(norm)}$. Further we multiply this fraction with normalized sum of words count $\mathbf{W}_{T_i}$ of the 'File_3.txt' text file sub-taxonomy (by dividing it with total words $\mathbf{W}_{total}$ in the text file) and then add this result into already calculated fraction. These values are calculated for sub-taxonomy $\mathbf{T}_0, \mathbf{T}_1$ and $\mathbf{T}_2$, and sum of all these three values gives us the value $\theta$. This $\theta$ is compared with minimum threshold score $\lambda$, (already calculated in section 4.4). We compare $\theta \geq \lambda$, in case this predicate is true then we have classified 'File_3.txt' as a job offer (*pos* class) otherwise negative (*neg* class).

Chapter 5

# Evaluation

## 5.1 Evaluation of Machine Learning Classification

All eight classifiers are trained on each of six dataset groups to find their optimal regularization values $\lambda$ (by sweet spot detection), table-5.1 shows these values, the last column shows the default values of $\lambda$. We visualized the different evaluation plots for generalization error with $\lambda$. There are only two dataset groups, Preprocessed-1 (Pre-1) and Preprocessed-5 (Pre-5), that achieve required evaluation criterion (Section 3.4), and both of these groups used *domain related preprocessing* method. *Learning curve* and *evaluation plots* helped to eliminate classifiers with clear evidence of generalization error. We selected Ridge Regression, SGD and SVC classifiers and these used Pre-1 and Pre-5 dataset groups. Visualization showed least RMSE and train and test dataset curves converged, as shown in figure 5.2 and figure 5.1 . Figure 5.4 shows the average RMSE of train and test datasets and their differences, this helped us to rank the classifiers. Figures 5.2 shows three *evaluation plots* for Ridge Regression, SGD and SVC with Pre-5 dataset group. *Evaluation plot* of Ridge Regression(top-left), with $\lambda = 100$, has average RMSE value 0.233 for training dataset and 0.256 for test dataset respectively, as shown in figure 5.4, and their difference is 0.02; figure 5.4 also showed SGD(top-right) and SVC(bottom-left) classifier evaluation values. These evaluation plots showed least RMSE values with train and test curves converged, which showed that these classifiers generalized well for new data (test data). Fig-

| Classifier | Raw | Prep-1 | Prep-2 | Prep-3 | Prep-4 | Prep-5 | Default |
|---|---|---|---|---|---|---|---|
| KNN | 6 | 25 | 15 | 15 | 20 | 30 | n_neighbors: 5 |
| Logistic Regression | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | C=1.0 |
| Naïve Bayes | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | alpha=1.0 |
| Perceptron | 1e-05 | 1e-05 | 1e-05 | 1e-05 | 1e-05 | 1e-05 | alpha=0.0001 |
| Random forest | 100 | 50 | 80 | 100 | 85 | 55 | n_estimators=10 |
| Ridge Classifier | 100 | 100 | 100 | 100 | 100 | 100 | alpha= 1.0 |
| SGD Classifier | 0.01 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 | alpha= 0.0001 |
| SVM | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | C =1.0 |

Table 5.1: Regularization parameter for different classifier to obtain optimum classification output and overcome over-fitting problem

ures 5.2(bottom-right) plot is the *learning curve* for SGD classifier and it is plotted for training and cross validation dataset to estimate [under and over]-fitting. *Learning Curve* for SGD classifier has Accuracy >0.9, both train and cross validation dataset converged, that shows that train dataset size is appropriate to eliminate high bias and low variance. Figure 5.1 shows the Ridge Regression and SGD classifiers with Pre-1 dataset group, their average RMSE values can be observed from figure 5.4.

We can observe from figure 5.4 that Ridge Regression,trained on Pre-1 dataset, has least RMSE and its train and test dataset RMSE difference is minimum, and next on the rank is SGD classifier with Prep-5 dataset.

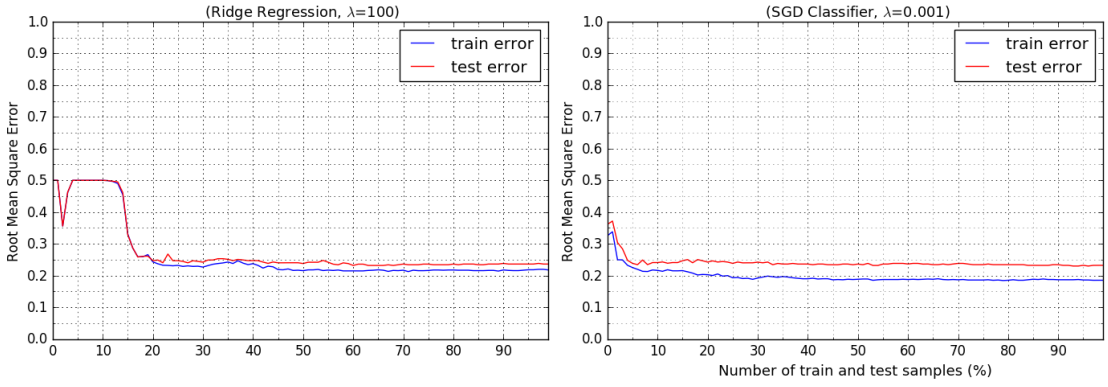


Figure 5.1: Preprocessed-1 (no Stemming or Lemmatization) dataset group evaluation graphs

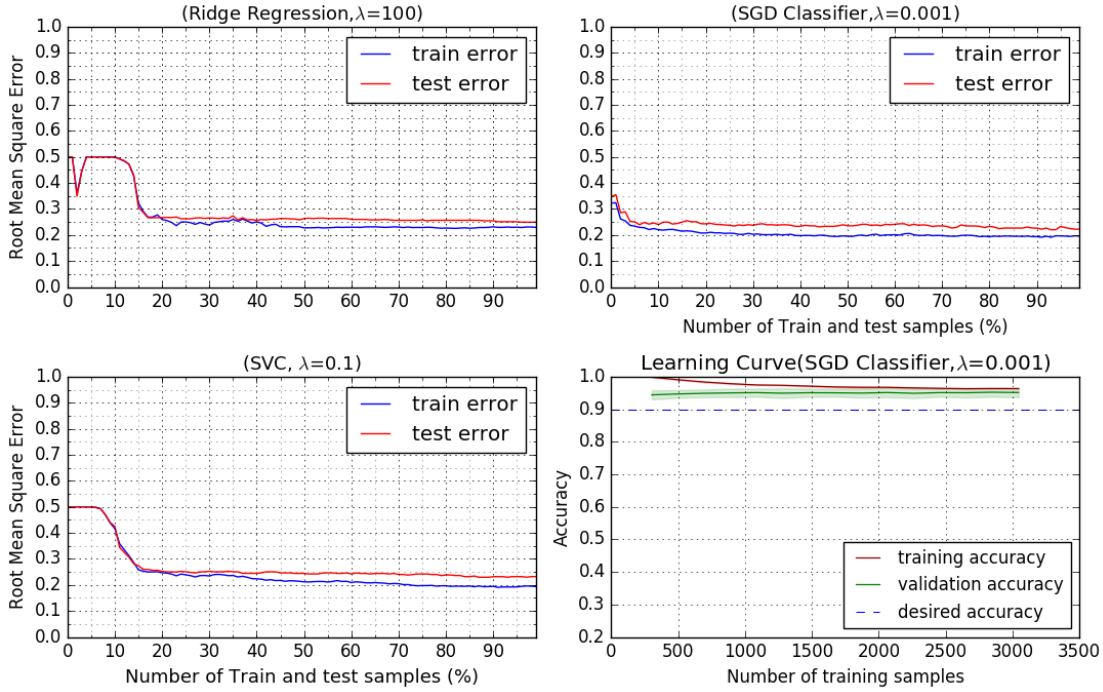


Figure 5.2: Preprocessed-5 (stemmed and Preprocessed-1) dataset group evaluation graphs and on bottom right a learning curve graph

Figure 5.3(right-column) shows two *evaluation plots* for Perceptron and Random Forest classifiers. Learning curve for both of these classifiers (left-column) shows Accuracy >90%, but there is a big gap between train and cross validation dataset, which shows the overfitting problem - while training. The train and test dataset curves (right-column) in both the plots diverging, that is clear sign of overfitting - while testing. These classifiers showed same behavior for all six dataset groups.

Table 5.2 shows the SGD, Ridge Regression and SVM classifiers and the other two (highlighted with red color - for overfitting) are Perceptron and Random Forest. The accuracy, precession, recall values are listed in the table, and F1 and AUC scores are calculated with precision and recall. AUC score is used to compare the performance of classifiers. Table 5.3 shows the summary of all the classifiers and dataset they used. Only Ridge Regression and SGD classifier generalized for all dataset but their RMSE vary, and Perceptron and Random Forest overfit for every

dataset, while other classifiers showed both behavior.

| | | | Training Data | | | | Test Data | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Classifier | Dataset | Accuarcy | Prec | Recall | F1 | AUC | Prec | Recall | F1 | AUC |
| SGD Classifier | Prep-1 | 96.0% | 96% | 96% | 96% | 98.5% | 95% | 94% | 94% | 98.3% |
| Ridge Reg | Prep-1 | 94.7% | 95% | 95% | 95% | 98.4% | 94% | 94% | 94% | 97.9% |
| SVM | Prep-5 | 95.1% | 96% | 96% | 96% | 98.7% | 95% | 94% | 94% | 98.3% |
| SGD Classifier | Prep-5 | 95.2% | 96% | 96% | 96% | 98.6% | 95% | 95% | 95% | 98.3% |
| Ridge Reg | Prep-5 | 94.1% | 95% | 95% | 95% | 98.4% | 94% | 94% | 93% | 98.0% |
| Perceptron | Prep-5 | 94.2% | 99% | 99% | 99% | 97.8% | 92% | 92% | 92% | 98.2% |
| Random Forest | Prep-5 | 93.9% | 99% | 99% | 99% | 98.5% | 95% | 95% | 95% | 96.9% |

Table 5.2: a comparison of different classifier evaluated on different datasets

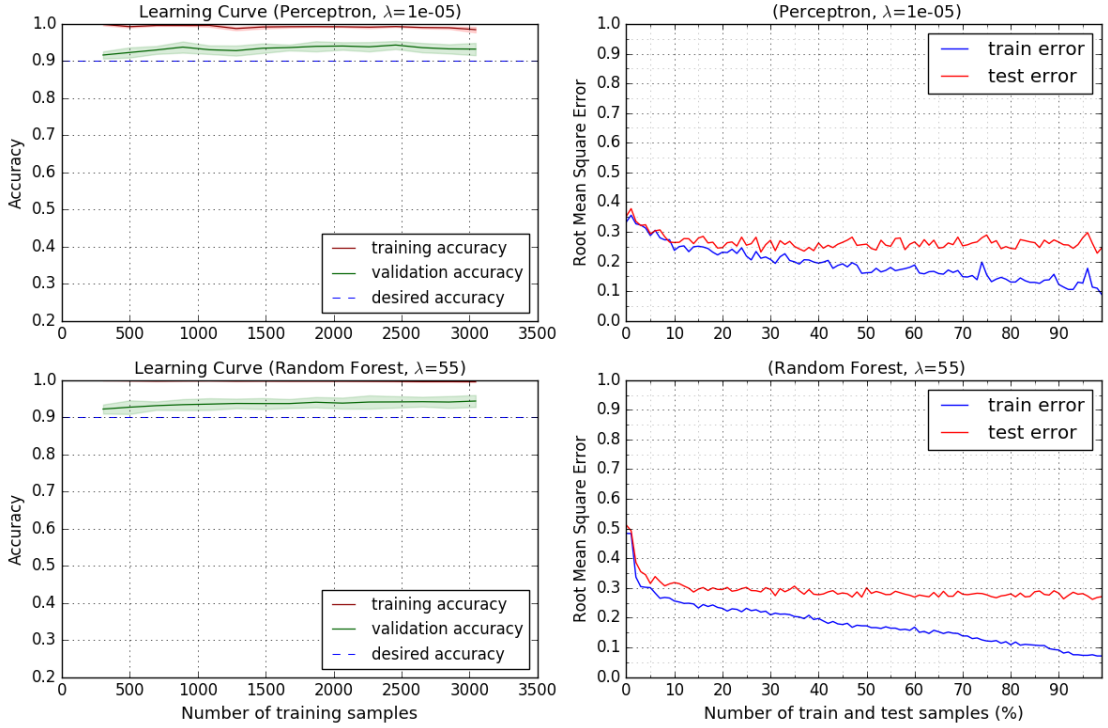


Figure 5.3: The Graphs to show the performance of Random Forest and Perceptron

### 5.1.1 Analysis

To classify IT job offers we investigated eight different classifiers to find out appropriate classifier(s) for IT job offers classification, and we also introduced a domain

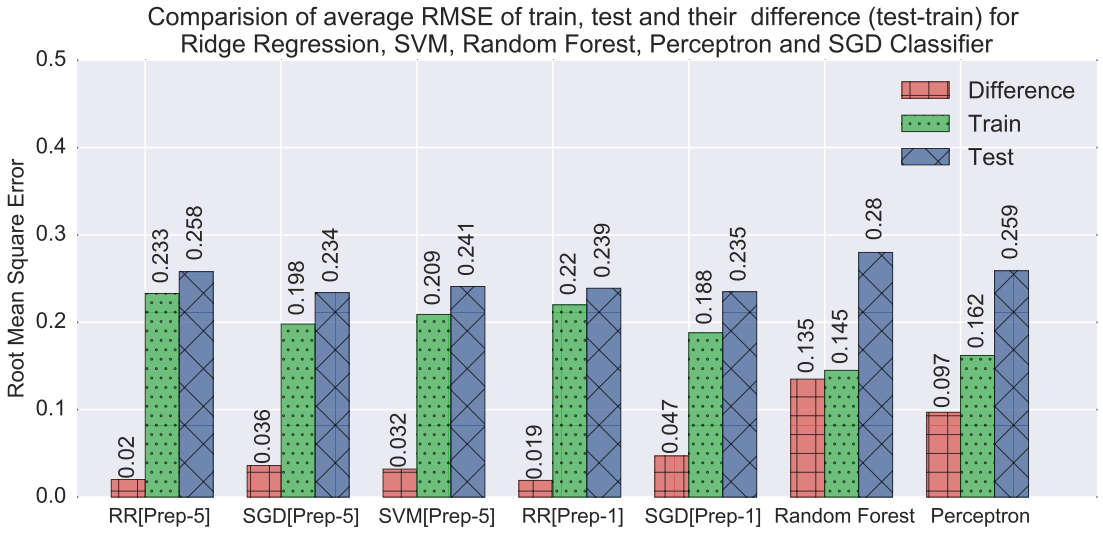


Figure 5.4: A comparison of RMSE

related preprocessing method, to get required performance of classifiers. The results of our study found that there are two classifiers, SGD and Ridge Regression, that are invulnerable for overfitting problem, for all dataset groups (Table 5.3), but the *domain related preprocessing* method reduced the RMSE values and improved the performance of these classifiers. Preprocessing of text is an important step for classifier outcome, e.g. the SGD and Ridge Regression classifier used raw data group without overfitting but RMSE values are higher, as compared to values after using *domain related preprocessing* (figure 5.4). Domain related preprocessing method improved the outcome of the classifier as compared to existing NLP preprocessing. It either used alone (figure 5.1) or with stemming (figure 5.2). *Domain related preprocessing* method is very specific to a certain domain - in our case IT job offers. We also found that Random Forest and Perceptron showed overfitting for every dataset group (Table 5.3) and not applicable for IT job offers classification.

Our suggested classifiers for IT job offers classification motivates to automate IT job offers retrieval from internet, by plugging our classification model into a crawler and upgrade it to a focused crawler(133) and harvest the required job offers from the World Wide Web.

| Dataset | kNN | LR | NB | Ptrn | RF | RR | SGD | SVM |
|---|---|---|---|---|---|---|---|---|
| `Raw` | ↱ | ↱ | ↱ | ↱ | ↱ | ∼ | ∼ | ↱ |
| `Prep-1` | ↱ | ↱ | ↱ | ↱ | ↱ | ∼ | ∼ | ∼ |
| `Prep-2` | ∼ | ∼ | ∼ | ↱ | ↱ | ∼ | ∼ | ↱ |
| `Prep-3` | ↱ | ∼ | ↱ | ↱ | ↱ | ∼ | ∼ | ↱ |
| `Prep-4` | ↱ | ∼ | ↱ | ↱ | ↱ | ∼ | ∼ | ∼ |
| `Prep-5` | ↱ | ∼ | ↱ | ↱ | ↱ | ∼ | ∼ | ∼ |

Table 5.3: A cross comparison of different classifiers to dataset groups. (Prep is equivalent to Preprocessing), ↱ symbol shows classifier over-fit while ∼ shows generalize well

The result of our study suggests that Ridge Regression and SGD classifier with *domain related preprocessing* method classify IT job offers accurately, as compare to any other classifier, and are applicable candidate to automate the retrieval of IT job offers from World Wide Web.

## 5.2 Evaluation of Ontology Based Classification

We used the ontology vocabulary to extract the concepts from all text files. One part of ontology, that deals with generic job offers, extracted general job offers with accuracy 0.8823(88.23%). The average concepts extracted from generic job offers files were 0.4302. For specific IT and not-IT job offers vocabulary, from the second part of ontology, the average rate of extraction for IT job offers is 0.156069 and for non-IT job offers this rate is 0.007178. Table 5.4 shows the seven maximum vocabulary frequency out of 177 concepts, extracted from IT and non-IT job offers. It is evident that for IT job offers ontology extracted concepts at higher rate then in non-IT job offers.

We observed that general job offers extraction rate is higher than more specific job offers because the concepts of ontology are common for all these job offers, that produced high extraction rate. On the contrary IT job offers ontology con-

| **Vocabulary** | **IT $W_{Count}$ Sum** | **non-IT $W_{Count}$ Sum** |
|---|---|---|
| keywords | 12502 | 239 |
| job | 6259 | 272 |
| development | 3881 | 35 |
| application | 3296 | 137 |
| support | 3169 | 317 |
| software | 2862 | 107 |
| projects | 2701 | 258 |

Table 5.4: Comparison of Concepts extraction from IT and non-IT job offers text files

sists of groups of concepts like programming, database management, networking, operating system maintenance etc., and these concepts are not common for all IT job offers. That is why the average rate of extraction of IT job offers are lower as compared to general job offer. Non-IT job offers extraction rate is much lower because in these job offers IT concepts are not available, due to which classifier failed to achieve minimum threshold and these jobs are classified as *neg*.

Figure 5.5 shows the graph of all the concepts frequency from IT and non-IT job offers text files (sorted in descending order), as a function of the vocabulary - numbered from 1 to 177. It can observed that the overall vocabulary frequency for IT job offers is better than non-IT job offers and words count for non-IT job offers never surpass the IT job offers and hence we can distinguish an IT job from non-IT job offer.

The compensating factor $CF_i$ (section 4.5) also showed some interesting results. Compensating factor improved the overall accuracy of the classifier by 3% (0.89 to 0.924) - for training dataset, and about 2% for test dataset.

For IT job offers we evaluated our ontology based classifier on training and test dataset and get the promising results as shown in table 5.5. We got more than 90% accuracy, precision, recall and F1 score for this ontology based classifier. For generalization error analysis, we plotted Root Mean Square Error (RMSE) of training and test dataset for varying number of training and test samples, as shown in the figure 5.6. This visualization depicts small RMSE with both train and test

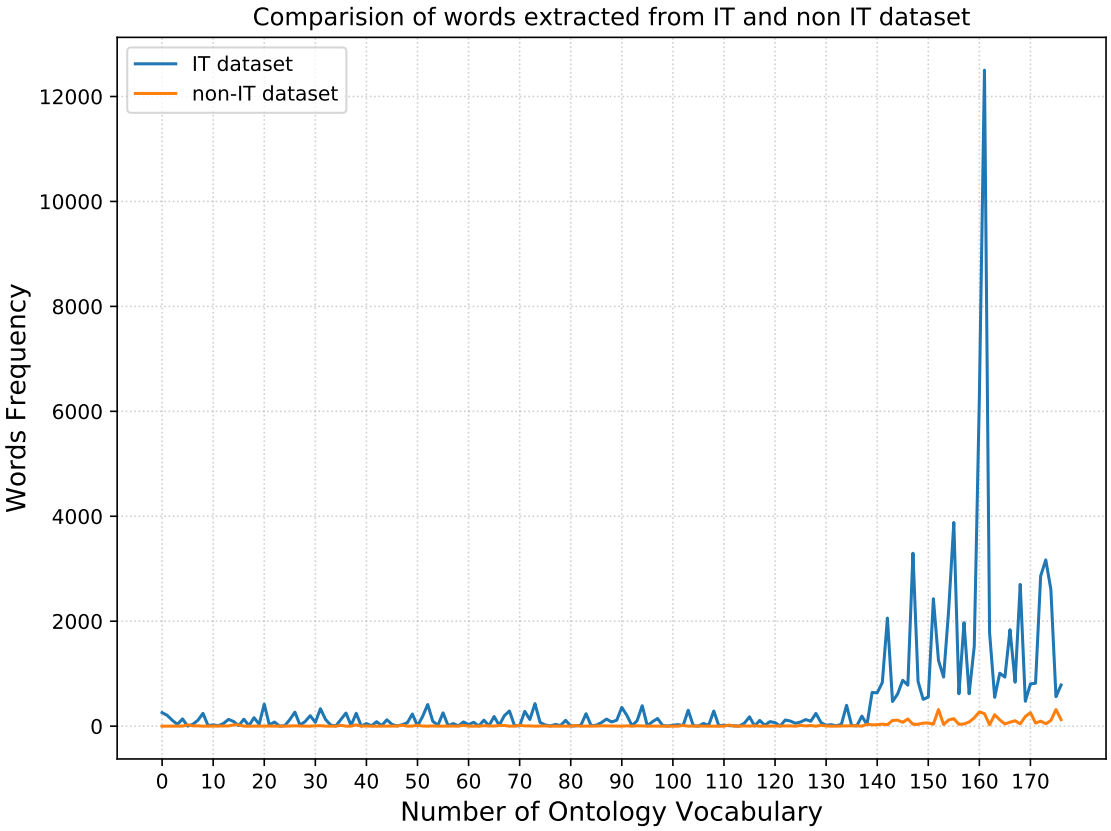


Figure 5.5: A comparison of concepts frequency extracted from IT and non-IT job offers text

dataset curves converged with small gap between two curves. The graph shows that for training and test dataset classifier started with varying RMSE and then curves become stable and converged. In this visualization train and test curves exhibited small RMSE, and both curves converged with small gap between two curves, that indicates model is not showing overfitting problem and generalized for new data.

### 5.2.1 Analysis

To classify job offers we developed an ontology which is capable of classifying generic as well as specific job offers; we also used this ontology to extract the ontological concept from the job offers text. The result of our study suggested

| Dataset | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Training | 0.924 | 0.930 | 0.917 | 0.923 |
| Testing | 0.915 | 0.919 | 0.91 | 0.915 |

Table 5.5: Ontology based classifier Accuracy, Precision, Recall and F1 score of IT job offers

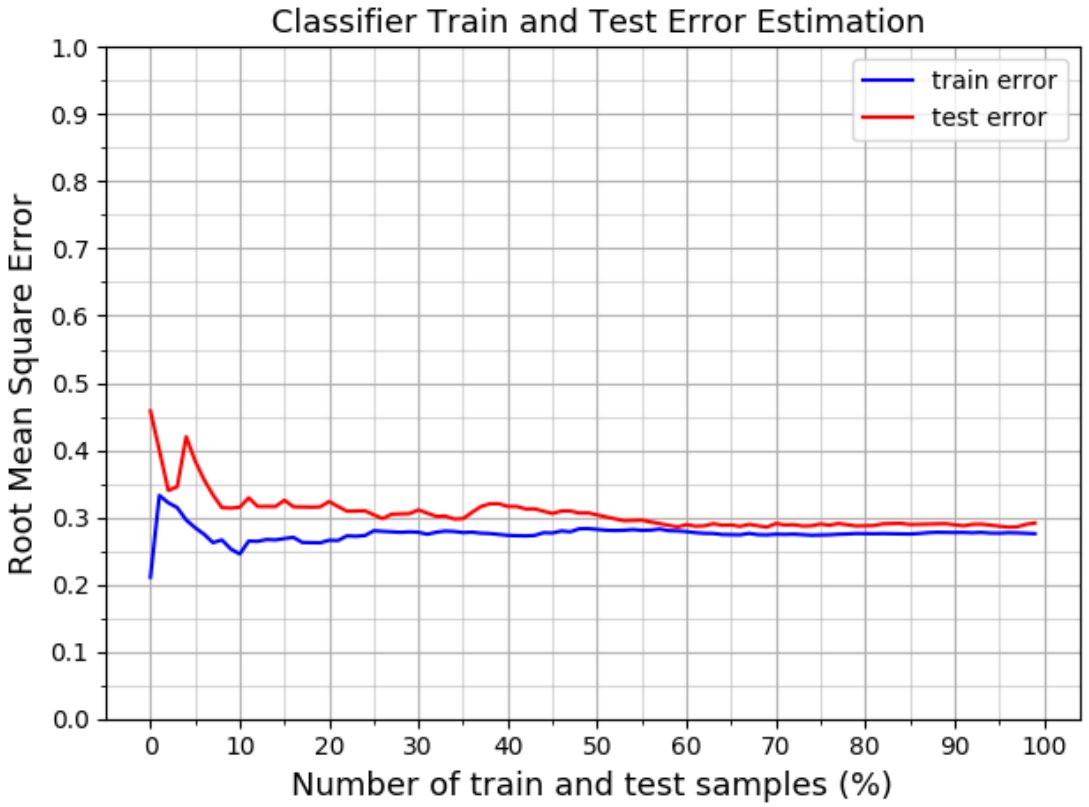


Figure 5.6: A comparative performance of classifier for training and testing dataset

that ontology based classifier can be used to classify both generic and specific job offers. The ontology we defined in two parts and both of these parts are capable to function independently, and we can classify generic job offers and IT specific job offers independently. That make our classification model more flexible, such that if we need to extend our ontology for more specific job offers like 'mechanical engineer', or 'chemical engineer', that only required to extend existing ontology and

update the existing classification model for that new defined ontology, that process simulate multiclass classification(134) in machine learning, but we don't need to repeat the whole training process in our ontology based classification system, rather update for only new defined ontology. Another advantage of this classifier is that it required only rudimentary preprocessing of text, as we treated job offer text by removing stop words only, and we don't need any *stemming* or *Lemmatization* or any other preprocessing method from Natrual Language Processing(NLP). If we need to update our existing classification model (i.e. generic and IT and non-IT ontology based model) we only need to add some additional concepts in the existing ontology and recalculate the minimum threshold only.

This ontology based classification also pave the way for automatic job offers classification and retrieval by plugging ontology based classifier into a crawler (focused crawler(133)), to classify and retrieve generic or specific job offers automatically.

Our results showed that with the help of ontology we successfully classified generic job offers, and further extend and applied model for binary classification and classified IT job offers from non-IT job offers with high accuracy, precision, and recall. With flexible classification model we can extend our model for further domain without disturbing the existing model.

Chapter 6

# Conclusion and Future Directions

For mining the information related to employment we used classification methods from two different discipline, named, Machine Learning and Semantic web technologies. From Machine Learning we used eight different algorithms for text classification to classify job offers in a binary classification paradigm. The other discipline we used for mining the job offers were ontology, for generic job offers, and further applied the same method for more specific job offers.

In the Machine learning classification procedure we used binary classification paradigm, and used dataset consists of IT and non-IT job offers. We found that Ridge Regression and SGD are the two most appropriate classifiers for job offers classification. The effectiveness and generalization of these classifiers are better then rest of the classifiers. We also found that two classifiers, Random forest and Perceptron, showed high bias and variance for all the data groups, and these two classifiers are unfavorable for job offers classification.

A very important phase for machine learning classification is preprocessing, the effectiveness of classification critically depends upon preprocessing of the input dataset. Preprocessing feature selection procedure reduces the computational cost and improves scalability, efficiency and effectiveness of a text classifier, by eliminating irrelevant text features(135)(20). Due to the high dimensionality of text features, feature selection is an important phase in text classification. A good feature selection method should consider domain characteristics to perform its task(136). We used multiple preprocessing methods on dataset and organized the dataset into five groups, plus, one dataset without any preprocessing. We

introduced one of our own method for preprocessing, called *domain related preprocessing*, that deals with preprocessing related to job domain. We found that *domain related preprocessing* method helped to improve the effectiveness of six classifiers, including those two which are on top of the ranking list, i.e. Ridge Regression and SGD classifiers.

Ontology represents the abstract model of concepts (also known as classes or terms) that may exist in some certain domain of interest. Share-ability and reuse-ability make ontology a powerful tool to represent domain knowledge (36)(137). These features of ontology encouraged us to use ontology for job offers documents classification. We defined our own algorithm based on ontology, for job offers classification. We develop multiple ontologies to classify not only generic job offers but also more specific job offers domain. We study the case of IT and non-IT job offers domain for more specific job offers. We defined a method to extract(138)(99) job offers concepts from job offer text with the help of ontology, and then classify the job offers without the use of Machine Learning algorithms. The classification phenomenon required a threshold score which decides to categorize a text into positive or negative category (in case of binary classification paradigm). Our ontology based algorithm first calculated minimum threshold ratio, and then based on this threshold score proceed for classification.

We found that with the help of ontology we can classify online job offers without using Machine learning algorithms. The domain ontology was used: to calculate and then use minimum threshold for the classification, extracted the ontology concepts from the job offer text, classify job offers with the help of simple mathematical equation (equation 4.14). We evaluated ontology based extraction process using a case study of IT and non-IT job offers. We used Machine learning method for evaluation of ontology based classification model and its generalization behavior. Preprocessing is an important and time consuming phase of classification in machine learning. In our approach, we used only stop words removal preprocessing because these stop words can bias the calculation, which we have defined. There is no other preprocessing or conversion is required, and hence the classification is more efficient as compared to machine learning. Another attribute of our system is its extensibility. If we need to extend our classification for more specific job offers then all we need to do is to construct it corresponding ontology, and update model

for the newly defined job type, we do not need to change existing model. We found promising evaluation results, which can be used in future in any application, to automate the process of job offers classification from the World Wide Web. As a future work, we need to investigate some automatic or semi-automatic mechanism to extract common vocabulary from job offers dataset.

## 6.1 Future Directions

Change is a continuous phenomenon, the only thing in the universe which is 'constant' is 'change'. We are living in the world where rate of change is much higher as compared to one hundred year ago, because of the invention of very high performance computing devices. Information technology is a discipline which is also changing with the passage of time, new fields emerge, old improve and older become obsolete, due to which new trends for IT jobs emerge. That phenomenon required to update the machine learning based classification model, to get the same performance of an existing IT job offer classifier. An effective mechanism needs to study in future to update the learning cycle of IT job offers classifiers models, in order to continuously achieve the required performance.

Currently we are manually selecting ontology vocabulary from job offers dataset. As a future work, we need to investigate some automatic or semi-automatic mechanism to extract common vocabulary from job offers. Feature selection can be one of the methods that can help to extract common vocabulary. The feature section is a procedure to select subset of features from the original documents by using some statistical manipulation, such that the words with highest score are selected. Some of available methods for measurements are *Gini Index*, *Information Gain*, *Mutual Information*, *Chi-Square* or $x_2$ *Statistic* or *TF-IDF*(28)(139)(140) etc. This feature selection procedure can help us to construct domain ontology by semi-automatic way, and hence we can reduce time for manual selection of features - for ontology construction. Due to this procedure the probability to find more vocabulary is higher as compare to manual selection.

**Web Crawler** Web crawler is one of the key components of the search engine. Web crawler is a multi threaded module which browses the world-wide-web in

automated manner by following the hyperlinks. Every search engine crawler visits the whole web periodically to retrieve the web documents and if there is any modification in the existing documents, or there are new web documents available, then this change is updated in search engines documents repository. Retrieved web documents are indexed and stored in the document repository for efficient execution of user queries(141). A well-behaved crawler follows an Exclusion protocol, defined in a robots.txt file, to visit only pages which are authorized by web site owner and follow a defined number of requests within a given period of time to avoid traffic overflow.

**Focused Web Crawler** Current search engines based on keyword search have some limitations, such as mostly retrieved web contents are irrelevant, results are highly dependent on vocabulary and semantically synonymous keywords are meaningless for the search engines, and information which spread over multiple web pages is hard to retrieve. Even if the query respond is successful, user time and effort are required to browse these web contents to retrieve the relevant information(142). This motivation leads to an enhanced version of the web crawler called Topical crawler (143)(144) or focused crawler (133), which traverses only those web pages which comply with the predefined domain or topic and discard irrelevant web documents, and save computation and communication resources(145).

In our research, we found multiple methods using two different discipline for job offer classification. One method is based on machine learning procedure and other is ontology based method for text classification. We observed promising results from both of these methods. We can utilize both of these method to develop a focused crawler. With our methods for job offer classification we can upgrade a general web crawler to a focused web crawler, to harvest the online job offers from the World Wide Web, and can automate the job offers classification on the internet or World Wide Web.

Appendix A

# Job Regular Expressions used in Machine Learning Classification

1. `removeNonAsciiChar = [^\\x20-\\x7E]`
2. `dotNetRegx = (\.(net|Net|NET))|(dot(\s+(Net|NET|net)))`
3. `sdlc = (system|software)\s+development\s+life\s+cycle(\s*\(sdlc\))*`
4. `oop = object(-*|\s+)oriented(-*|\s+)programming(\s*\(oop\))*`
5. `ood = object(-*|\s+)oriented(-*|\s+)design(\s*\(ood\))*`
6. `ooNotPOrD = object(-|\s+)oriented(?!(-|\s+programming|-|\s+design))`
7. `salary = (\$|Âč|âĆň|eur)*\s*(?<!(db|\bneo|\blog))\d+(?!ee)\s*(\.*,*\d*)k*`
8. `htmlTag = (\<\s*\w*\s*\>)`

9. `replaceIT = \bIT\b|\b(?i)information\s+technology\b`

10. `salary = (?i)((((\bpension\b|\bbenefit\b|\bcompensation\b|\bcompetitive\b|\bannual\b|\bpays*\b|\bchild\s*care\b|\binsurance\b|\bmedical\b).*(salary|pay))|((salary|pay).*(\bregion\b|\bbetween\b|\bpay\b|\bcompetitive\b|\bbonus(es)*\b|\bcommensurate\b|\bannually\b|\bbenefits*\b|\bmedical\b|\bdental\b|\binsurance\b|\bvacation\b|\bholidays*\b|\btime\soff\b|\bvision\b|\bdisability\b|\bexp(erience)*\b|\bpension\b|\bper\sannum\b|\bhealth\s*care\b|((Âč|$|âĆň|euro)*\dk*(.*(âĂŞ|-|to).*\d(k*))*)|\ballowances*\b|\btraining\b|\bchild\s*care\b|\bprofit\b|\bretirement\b))|\bsalary\b)`

11. `apply = (?i)(((\bopportunity\b)|(\bthank\syou\b)|(\bview\b)|(\bencouraged\b)|(\bcitizens*\b)|(\bplease\b)|(\bclick\b)).*\bapply(ing)*\b)|(\bapply(ing)*\b.*(\bvacancy\b|\bclick\b|\bnow\b|\bhere\b|\bonline\b|\bjob\b|\bposition\b|\brole\b|\bhimself\b|\bherself\b|\be\s*-*mail\b|\bbutton\b|\btoday\b|\bopportunity\b|\b(web)*site\b|\bcv\b|\bresume\b|\bsend\b|\bvisa\b))`

12. `splitWithFullStop = (?i)((?<!(\d+|etc|\bu\b|\bs\b|\binc\b|\bltd\b|\bno\b))(\.|;)(?!(\d+|\bnet\b|\betc\b|\bjs\b)))`

13. `atTheRateOf = (?i)(twitter\s*)*@`

14. `shareJob = (?i)(\bshare\b).*(\bjob\b)`

15. `pleaseNote = (?i)please(\s+(note\:|note))`

16. `subscribeTo = (?i)(to\s+)*\b(un)*subscribe\b(\s+to)*`

17. `email = (?i)((call|please|job|check|position)*.*(\be-*\s*mail\b|\bsend\b).*(resume|cv|job|positions*|address))|(call|please|job|check|position).*(\be-*\s*mail\b)`

18. `srcrg = (?i)\bsex\b|\brace\b|\bcolor\b|\breligion\b|\bgender\b`

19. `eoe = (?i)@|\bequal\s+employment\s+opportunity\b|\beeo\b|\bequal\s+opportunity\s+employer\b|\beoe\b //|\b(to)*\s+apply\b|\bapply(\s+now)*\b`

Appendix B

# Jobs Ontology Mapped Regular Expressions

1. `(?i)\bcareers*\b`
2. `(?i)\bjobs*\b`
3. `(?i)\bemployment\b`
4. `(?i)\bresponsibilities\b|\bresponsibility\b|\bresponsible\b`
5. `(?i)\bduties|duty\b`
6. `(?i)\bsolutions*\b`
7. `(?i)\bimplements*(ations*|ing|ed)*\b`
8. `(?i)designs*(ing|ers*)*`

9. `(?i)\btask(s|ed|ing)\b`

10. `(?i)requirement|\brequired*\b`

11. `(?i)\bskills*\b`

12. `(?i)projects*\s*(manage(ment|r))*`

13. `(?i)\bcommunications*\b`

14. `(?i)\bwritten\b|\bwriting\b`

15. `(?i)\boral(ly)*\b`

16. `(?i)\btechnical\b`

17. `(?i)\bexperienced*\b`

18. `(?i)\byears*\b|\byrs\b`

19. `(?i)\bprefer\w{2,6}\b`

20. `(?i)\bqualifications*\b|\bqualify\b|\bcredentials*\b`

21. `(?i)\bcompetenc(y|ies)\b`

22. `(?i)\bdiplomas*\b`

23. `(?i)\bhigh\s+school\b|\bHS\b`

24. `(?i)equivalent`

25. `(?i)\b(?<!\d{1,3}.{0,3})degrees*\b`

26. `(?i)\beducation\b`

27. `(?i)\bknowledge\w{0,5}\b`

28. `(?i)\bundergraduate\b|\bgraduate(d|s)*\b`

29. `(?i)\bmasters*\b`

30. `(?i)bachelor`

31. `(?i)\bdescription\b`

32. `(?i)\binformation\s+technology\b|\b(?-i)IT\b`

33. `(?i)\bdevelopments*\b`

34. `(?i)\btest(ing)*\b`

35. `(?i)evaluat(es*|ion)`

36. `(?i)\bcoding\b`

37. `(?i)\banalysis\b|\banalyze(s|d)*\b`

38. `(?i)\bbenefits*\b`

```
39.     (?i)\bretirements*\b

40.     (?i)\bmedical\b

41.     (?i)\ballowances*\b

42.     (?i)\bdental\b

43.     (?i)\btime\s+off\b|\bvacations*\b|\bholidays*\b

44.     (?i)\bsalary\b|\bpay\b

45.     (?i)\bapplications*\b

46.     (?i)\bapply(ing)*\b

47.     (?i)\bpensions*\b

48.     (?i)\binsurances*\b

49.     (?i)\bhealth\s+care\b

50.     (?i)\bannum\b|\bannual(ly)*\b

51.     (?i)\bbonus(es)*\b

52.     (?i)child\s+care

53.     (?i)\brecruit(s|ing|ment|ers*)*\b|\bhiring\b|\blooking\s+to\b
            |\blook(ing)*\s+for\b
```

54. `(?i)\bcontribut(es*|ion|ing)\b`

55. `(?i)\btravel\b`

56. `(?i)\bpositions*\b|\broles*\b`

57. `(?i)\bopportunit(y|ies)\b`

58. `(?i)\bresumes*\b|\b(?i)cv\b|\bcurriculum\s+vitae\b|\bcover\s+letter\b`

59. `(?i)\be*-*mail(ing)*\b`

60. `(?i)\bwork(ing)*\b`

61. `(?i)\bapplicants*\b`

62. `(?i)\bemploy(ee|er)s*\b`

63. `(?i)\blocations*\b`

64. `(?i)\bpart-*\s*time\b|\bfull-*\s*time\b|\bpermanent\b|\bcontract\b|\btemporary\b|\b(job|employment)*\s*type\b`

65. `(?i)\be\s?-?commerce\b`

Appendix C

# IT Jobs Ontology Mapped Regular Expressions

1. `(?i)job`
2. `(?i)software\s+solution`
3. `(?i)debugs?(ing)?`
4. `\b(?-i)TDD\b|(?i)\btest\s+driven\s+development\b`
5. `(?i)((data|software|test(ing)?)\s+\w{0,15}\s*verification)|(validation\s+\w{1,15}\s+verification)|(verification\s+\w{1,15}\s+validation)`
6. `(?i)quality\s+metrics`
7. `(?i)business\s+modeling`

8. `(?i)\bautomated\s+test(ing)?\b`

9. `(?i)(?<!\bbig\b.{0,5})\bdata\b(?!.{0,5}(\barchitecture\b|warehous(es|e|ing)?\b|\bmining\b|\bdesign\b|\bstructure\b|\bverification\b))`

10. `(?i)\bbig\s?data\b`

11. `(?i)data\s+design`

12. `(?i)normalization`

13. `(?i)(?!business.{1,5})model(ing)?`

14. `(?i)reporting\s+tool`

15. `(?i)masking`

16. `(?i)quality`

17. `(?i)metadata`

18. `(?i)\bflow\s+diagrams?\b`

19. `(?i)(?<!\b[information\s+technology].{1,5})management(?!.{1,5}(information\s+systems?\b))`

20. `(?i)\breport\s+writing\b`

21. `(?i)dictionaries`

22. `(?i)\banaly(tical|tics?)\b`
23. `(?i)repository`
24. `(?i)processing`
25. `(?i)\bmodeler\b`
26. `(?i)\bdata\s+architecture\b`
27. `(?i)\bevaluat(e|ing).{1,11}(?!\.\:\;\,)(\bprecision\b|\baccuracy\b)`
28. `(?i)(?<!test.{1,5})cases?`
29. `(?i)(?<!\bmarket.{1,5})\b(IT\s)?risks?\b(?!.{1,5}(reporting|manage(ment|r)|analy(sis|tic)))`
30. `(?i)\brisk\s+architecture\b`
31. `(?i)\brisk\s+reporting\b`
32. `(?i)risk\s+manage(ment|r)`
33. `(?i)\brisk\s+analy(sis|tic)\b`
34. `(?i)\bcode\s+quality\b`
35. `(?i)\bQA\b|\bquality\s+assurance(\s+engineer)?\b`
36. `(?i)automation`

37. `(?i)plans`

38. `(?i)troubleshoot(ing)?`

39. `(?i)\bmarket\s+risk\b`

40. `(?i)maintenance`

41. `\b(?<=(\bux\b|\bui\b|dynamic|java|develop(ing|ment)?|agile|servers?|implement).{1,11})\bweb\b|\bweb\b(?=.{1,15}(filter|framework|project|skills|applications?|develop(ment|er)?|interface|technolog(y|ies)|systems?|apps?|frontend|\b(java)?script(ing)?\b|html\d?|tools?|php|hosting|methods?|apis?|crawling|automation|software|security|pages|mvc|standards?))`

42. `\b(windows)?\s*?(O/S|O\s?S)\b|\b(windows)?\s+operating\s+system\b`

43. `(?i)\biOS\b`

44. `(?i)\bWindows\s+based(\s+environment)*\b`

45. `(?i)\bUnix\b`

46. `(?i)\bLinux\b`

47. `(?i)\bFreeBSD\b`

48. `(?i)\bAndroid\b`

49. `(?i)deployment`

50. `(?i)(?<!(desktop|windows|enterprise|internet).{1,5})application (?!.{1,5}(integration|development|server))`

51. `(?i)\bimplement(s?|ing|ations?)?\b`

52. `(?i)(?<=(oracle|ETL|mainframe|DB(A|\d)|agile|\bGUI\b|(\B\.\s?net\s?\d?\b)|(c#\s?|c\s?sharp\s?|\bvb\s?\d{0,2}\b|\b(?<!\.)asp\b)(\.\s?net)?|applications?|\bweb\b|database|programming|software|front-?\s?end|php|\bBI\b|business\s?intelligence|data\s?warehouse|SQL|full\s?stack|android|\bios\b|xml|ux|java|python|cognos|c\+\+|\bc\b|\bapi\b|\bssis\b|language).{1,25})(\bdevelopment\b|\bdevelopers?\b|\bdevelops?\b)|(\bdevelopment\b|\bdevelopers?\b|\bdevelops?\b)(?=.{1,23}(DB(A|\d)|ETL|code|agile|(\B\.\s?net\s?\d?\b)|(c#\s?|c\s?sharp\s?|\bvb\s?\d{0,2}\b|\b(?<!\.)asp\b)(\.\s?net)?|principles?|\bweb\b|database|programmer|software|front-?\s?end|php|\bBI\b|business\s?intelligence|SQL|ios|xml|ux|java|python|ruby|cognos|cobol|c\+\+|\bc\b|\bapi\b|n-?tier|methodologies))`

53. `(?i)(?<!(database|schema|oriented|\boo\b|data|ux).{1,5})\bdesign\b (?!.{1,5}patterns?)`

54. `(?i)\bcomputer\s+systems?\b`

55. `(?i)(?<!relational\s{0,5})(databases?)(?!.*?management)|content\s+?(management\s+?systems?)|\bCMS\b`

56. `(?i)\bRDBMS\b|\brelational\s+?databases?\s+?management\s+?systems?\b`
57. `(?i)database\s*?design(er|ing)?`
58. `(?i)(?<!(sql|web|application).{0,5})servers?(?!.{1,5}(process|side))`
59. `(?i)administration`
60. `(?i)\bquery(ing)?\b`
61. `(?i)\bstored?\s+?procedures?\b`
62. `(?i)(platform)*\s+performance`
63. `(?i)SQL\s+Server|SQL(?!\s+Server)`
64. `(?i)schema\s+design`
65. `(?i)\bDBMS\b|\bdatabase\s*?management\s*?systems\b`
66. `(?i)architecture`
67. `(?i)\brelational\s+data\s*?bases?\b`
68. `(?i)\bSQL\s+reporting\b`
69. `(?i)system\s+(\badmin\b|\badministration\b|\badministrator\b)+`
70. `(?i)\bDBA\b`

71. `(?i)analyst(?!.{1,5}(requirements?|procedure))`

72. `(?i)requirement\s+specifications?`

73. `(?i)(?<=linux.{1,5})embedded|embedded(?=.{1,21}(c\+\+|\bc\b|real-?time|system|develop(er|ment|ed)|linux|platform|firmware))`

74. `(?i)analy(zing|sis|st|ze|ics)\s+requirements?`

75. `(?i)analy(zing|sis|st|ze|ics)\s+procedure`

76. `(?i)programming`

77. `(?i)\btest\s+cases?\b`

78. `(?i)\bdata\s+warehous(es|e|ing)*\b`

79. `(?i)technical\s+specification`

80. `(?i)documentation`

81. `(?i)(?<=(develop(ment|er)|programm(ing|er)).{1,41})(\bGUI\b|\bgraphical\s?user\s?interface)|(\bGUI\b|\bgraphical\s?user\s?interface)(?=.{1,30}(develop(ment|er)|programm(ing|er)|framework|testing|web|tool|validation|design|software|front\s?end|implementation|interfaces?|applications?))`

82. `(?i)\b(systems?|project|software|app(lications?)?)\s+\w{0,25}\s*(development|test(ing)?)?\s+life\s?cycle\b`

83. `(?i)system\s+requirement`

84. `(?i)\bcoding\b`

85. `(?i)programming\s+language`

86. `(?i)development\s+language`

87. `(?i)coding\s+language`

88. `(?i)program`

89. `(?i)software(?!.{1,5}(solution|application|system|project))`

90. `(?i)software\s+application`

91. `(?i)software\s+system`

92. `(?i)software\s+project`

93. `(?i)(?<!automated.{1,5})test(er|ing)?(?!.{1,5}cases?)`

94. `(?i)upgrade`

95. `(?i)web\s+related`

96. `(?i)web\s+based`

97. `(?i)data\s+structure`

98. `(?i)\binstall(ation)*\b`

99. `(?i)packages`

100. `(?i)(\bclouds*\b\s*\w{0,100}\s*(-*based|centric|computing|analytics*|services|security|experience|deployment|management|technolog(y|ies)|providers*|implementation|environments*|operation|platform|engineer(ing)*|application|data|architecture|infrastructure|foundry|hosting|program|support|integration|development|storage|space|solution))|((?:)(private|develop|analytic|experience|manage)\s+\w{0,100}clouds*)`

101. `(?i)\b(inversion\s+of\s+control)|\bIoC\b`

102. `(?i)internet\s+application`

103. `(?i)network\s+service`

104. `(?i)server\s+process`

105. `(?i)application\s+development`

106. `(?i)\bmodel\s+view\s+controller\b|\bMVC\b`

107. `(?i)\bcascading\s+style\s+sheets*\b|\bCSS\d?\b`

108. `(?i)application\s+servers?`

109. `(?i)server.{1,2}side`

110. `(?i)\bwebserver\b|\bweb\s+\w?\s?server\b`

111. `(?i)\bsemantic\s+web\b`

112. `(?i)\bweb\s?services?\b`

113. `(?i)\bservice\s+oriented\s+environment\b`

114. `(?i)\bservice\s+oriented\s+architecture\b|\bSOA\b`

115. `(?i)\bWSDL\b`

116. `(?i)\bSOAP\b`

117. `(?i)\bRESTful\b`

118. `(?-i)\bREST\b`

119. `(?i)\b(?<!http\:.{0,100}|www\..{0,100})x?HTML\d?\b`

120. `(?i)(?<!computer.{1,5})\bengineer(s|ing)?\b`

121. `(?i)validate`

122. `(?i)projects?\s?(manage(ment|r))?`

123. `(?i)\bSDLC\b|\bAgile\b`

124. `(?i)programmer`

125. `(?i)\balgorithms?\b`

126. `(?i)configuration\s+manager`

127. `(?i)(object(-*|\s+)oriented|OO)\s+(programm(er|ing)|code)|\bOOP\b|(?-i)\bOO(?!\s+(design|analysis))\b`

128. `(?i)desktop\s+application`

129. `\bUML\b`

130. `(?-i)\bOOD\b`

131. `(?i)application\s+integration`

132. `(?i)\bobject(-?|\s+)oriented\s+analysis\b`

133. `(?i)windows\s+application`

134. `(?i)build\s+tool`

135. `(?i)\bapps?\b`

136. `(?i)\bobject(-?|\s+)oriented\s+design\b`

137. `(?i)\bOOAD\b`

138. `\b(?i)OO\s+design\b`

139. `(?i)\bAPIs?(?!\s?\d{1,3})\b`

140. `(?i)\bcontinuous\s+integration\b`

141. `(?i)development\s+tool`

142. `(?i)\benterprise\s+application\b`

143. `(?i)\bspecification\s+development\b`

144. `(?i)\bdesign\s+patterns\b`

145. `(?i)program\s+logic`

146. `(?i)\bdata\s+mining\b`

147. `(?i)performance\s+(tune?n?(ing)?|computing)`

148. `(?i)\bnetwork\b(?!\s+(manager|server|services?|engineer(s|ing)?))`

149. `(?i)network\s+manager`

150. `(?i)network\s+servers?`

151. `(?i)computer\s+security`

152. `(?i)firewall`

153. `(?i)\bHTTPS?\b(?!\:\s?//)`

154. `(?i)(?<!risk.{1,5}) analysis(?!.{1,5}(requirements?|procedure))`

155. `(?i)workstation`

156. `(?i)\bDNS\b|\bDHCP\b`

157. `(?i)\bmainframe\b`

158. `(?i)networking`

159. `(?i)\bIPv\d+\b|\bTCP.{1,7}?IP\b|(?<=(routers?|switches?|network(ing)?|voice\s?over).{1,15})\bIP\b|\bIP\b(?=.{1,31}?(subnets?|conflicts?|DHCP|rout(ing|e)|network(ing|s)?|phone|address(ing)?|load\s?balance|configuration|protocols?|tcp))|\binternet\s+protocol\b`

160. `\bSMTP\b`

161. `(?i)evaluation`

162. `(?i)security\s+specialist`

163. `(?i)support`

164. `(?i)security\s+threat`

165. `(?i)security\s+system`

166. `(?i)\bnetwork(ing)?\s+engineer(s|ing)?\b`

167. `(?i)degree`

168. `(?i)qualification`

169. `(?-i)\bMIS\b|(?i)\bmanagement\s+information\s+systems?\b`

170. `(?i)\binformation\s+technology\s+(management)?\b`

171. `(?i)\binformation\s+system\b`

172. `(?i)\bcomputer\s+science\b`

173. `(?i)\bcomputer\s+engineering\b`

174. `(?i)framework`

175. `(?i)\b(?<!(programming|development|coding).{1,5})languages?\b`

176. `(?i)\bscripting\b`

177. `(?i)c(\s+sharp|\s*#)|\betl\b|\bdb2\b|(c\+\+)|\bjdk\b|\b(?<!(www|http).*)asp\b|\bwan\b|\b(?<!(www|http).*)php\b|\bqtp\b|\bpig\b|\bxml\b|\bcism\b|\bjira\b|\bsolr\b|\bajax\b|\bpj2d\b|\b(?<!(http|www).*)java\b|\bruby\b|\bidms\b|\bj\d*ee\b|\bajax\b|\bjdbc\b|\bhive\b|\bjson\b|\bcissp\b|\bagile\b|\bscrum\b|((?<!@.{0,100}|\w+)(\.(net|Net|NET))|(dot(\s+(Net|NET|net))))|\bazure\b|\bmysql\b|\bcissp\b|\bneo4j\b|\bhbase\b|\bnosql\b|\bsqoop\b|\bnosql\b|\bscala\b|\bdjango\b|\bn-Tier\b|\bvmware\b|\bhadoop\b|\b(perl)\b|\bjquery\b|\boracle\b|\bhadoop\b|\bmahout\b|\bgroovy\b|\blucene\b|\blabview\b|\bandroid\b|\bdba(2*)\b|\b(scala)\b|\bBizTalk\b|\btcpdump\b|\bmongodb\b|\bwinforms\b|\bms\s+sql\b|\bmapreduce\b|\`

```
bcassandra\b|\b(rabbitmq)\b|\b(teradata)\b|\bjavascript\b|\b(
angularjs)\b|\bcryptography\b|((?-i)\bERP\b)|\bvirtualization\b|\
b(z/VM|z\s+VM)\b|\bscripting\b|\bvisual\s+studio\b|\bux\s+design(
er)*?\b|\b(objective\s*-*c)\b|\bactive\s+directory\b|\bembedded\s
+software\b|\bjunit\b|\bunit\s+testing|\b(crystal\s+report(s*))\b
|\bhttpd\s+server\b|\baggressive\s+refactoring\b|\bmulti-*\s*
thread(ed|ing)*\b|\b(python(\s+developers*)*)\b|\bmicrosoft\s+
server\s+\d+\b|\bfull\s+stack\b|(?<=develop(ers?|ment).{1,27}|
programming.{1,10}|web(site)?.{1,15}|UI.{1,15}|test\s?automation
.{1,15}|HTML\d?.{1,55}|CSS\d?.{1,51}|application.{1,19})front\s?
ends?|front\s*ends?(?=.{1,15}develop(ers?|ment)|.{1,10}
programming|.{1,15}web(site)?|.{1,15}UI|.{1,15}test\s?automation
|.{1,36}CSS\d?|.{1,37}skills|.{1,10}application|.{1,27}interfac(
es?|ing)|.{1,15}testing|\s+tools?|\s+design)|\bback\s+end\b|\bRPG
\b|\bAS/?400\b|\bFPGAs?\b|\bCCIE\b|ITIL|\bIT\s?risk\b|CCNA|CCNP|\
bpowershell\b|websphere|\bDynamics\s?AX\b|\bdata\s*?scientists?\b
|cyber\s?security|cognos|\bjboss\b
```

# References


[1] International Labour Organization. Global unemployment expected to rise by 3.4 million in 2017. publised on Web of ILO, January 2017. 1

[2] Eleanna Galanaki. The decision to recruit online: a descriptive study. *Career Development International*, 7(4):243–251, July 2002. 1

[3] LinkedIn. Leading sources of quality hires according to hiring decision makers worldwide in 2016, 2016. 2

[4] Charmine E. J Härtel and (author.) Fujimoto, Yuka. *Human resource management*. Frenchs Forest, NSW Pearson Australia, 3rd edition edition, 2015. Previous edition: 2010. 2

[5] W. Finn. Screen test. *People Management*, pages 38–41, 2000. 2

[6] Lakshmi S.L. E-recruitment: A boom to the organizations in the competitive world. *IOSR Journal of Business and Management (IOSR-JBM)*, Volume 1. 3

[7] J. Rudich. Job hunting on the web. *Link-Up*, Vol.17, No 2:pp. 21–24, 2000. 3

[8] C. Taylor. Windows of opportunityŤ. eople Management, Vol. 7, No. 5:pp.32–36, 2001. 3

[9] David H. Autor. Wiring the labor market. *Journal of Economic Perspectives*, 15(1):25–40, March 2001. 3

[10] Peter Kuhn and Mikal Skuterud Skuterud. Internet job search and unemployment durations. 11 2002. 3

[11] Hazrul Shahiri and Zulkifly Osman. Internet Job Search and Labor Market Outcome. *International Economic Journal*, 29(1):161–173, January 2015. 3

[12] Bernard J. Jansen, Karen J. Jansen, and Amanda Spink. Using the web to look for work: Implications for online job seeking and recruiting. *Internet Research*, 15(1):49–66, February 2005. 3

[13] Tabbasum Naz. Configurable meta-search in the human resource domain: A hybrid approach to schema and data integration for meta-search engines. 2009. 4

[14] Yiu-Kai Ng, J. Tang, and M. Goodrich. A binary-categorization approach for classifying multiple-record Web documents using application ontologies and a probabilistic model. pages 58–65. IEEE, 2001. 4

[15] Mu-Hee Song, Soo-Yeon Lim, Dong-Jin Kang, and Sang-Jo Lee. Automatic classification of web pages based on the concept of domain ontology. In *Software Engineering Conference, 2005. APSEC'05. 12th Asia-Pacific*, pages 7–pp. IEEE, 2005. 4, 9, 30

[16] Rudy Prabowo, Mike Jackson, Peter Burden, and H.-D. Knoell. Ontology-based automatic classification for web pages: design, implementation and evaluation. In *Web Information Systems Engineering, 2002. WISE 2002. Proceedings of the Third International Conference on*, pages 182–191. IEEE, 2002. 4, 9, 31

[17] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996. 5

[18] Alper Kursat Uysal and Serkan Gunal. The impact of preprocessing on text classification. *Information Processing & Management*, 50(1):104 – 112, 2014. 5

[19] P. Chapman. *CRISP-DM 1.0: Step-by-step Data Mining Guide*. SPSS, 2000. 5

[20] Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. A Review of Machine Learning Algorithms for Text-Documents Classification. *Journal of Advances in Information Technology*, 1(1), February 2010. 5, 7, 18, 25, 73

[21] Cui Zifeng, Xu Baowen, Zhang Weifeng, and Xu Junling. A new approach of feature selection for text categorization. *Wuhan University Journal of Natural Sciences*, 11(5):1335–1339, September 2006. 6

[22] Li-Wei Lee and Shyi-Ming Chen. New methods for text categorization based on a new feature selection method and a new similarity measure between documents. *Advances in Applied Artificial Intelligence*, pages 1280–1289, 2006. 6

[23] Elena MontaÃśÃľs, Javier FernÃąndez, Irene DÃŋaz, ElÃŋas F. Combarro, and JosÃľ Ranilla. Measures of Rule Quality for Feature Selection in Text Categorization. In Michael R. Berthold, Hans-Joachim Lenz, Elizabeth Bradley, Rudolf Kruse, and Christian Borgelt, editors, *Advances in Intelligent Data Analysis V: 5th International Symposium on Intelligent Data Analysis, IDA 2003, Berlin, Germany, August 28-30, 2003. Proceedings*, pages 589–598. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. DOI: 10.1007/978-3-540-45231-7_54. 6

[24] Fadi Thabtah, M. Eljinini, Mannam Zamzeer, and W. Hadi. NaÃŕve Bayesian based on chi square to categorize arabic data. In *proceedings of The 11th International Business Information Management Association Conference (IBIMA) Conference on Innovation and Knowledge Management in Twin Track Economies, Cairo, Egypt*, pages 4–6, 2009. 6, 7

[25] Jukka Iivarinen, Kimmo Valkealahti, Ari Visa, and Olli Simula. Feature Selection with Self-Organizing Feature Map. In Maria Marinaro and Pietro G. Morasso, editors, *ICANN âĂŹ94: Proceedings of the International Conference on Artificial Neural Networks Sorrento, Italy, 26âĂŞ29 May 1994 Volume 1, Parts 1 and 2*, pages 334–337. Springer London, London, 1994. DOI: 10.1007/978-1-4471-2097-1_78. 6

[26] Jun Yan, Ning Liu, Benyu Zhang, Shuicheng Yan, Zheng Chen, Qiansheng Cheng, Weiguo Fan, and Wei-Ying Ma. OCFS: optimal orthogonal centroid feature selection for text categorization. page 122. ACM Press, 2005. 6

[27] M. Ikonomakis, S. Kotsiantis, and V. Tampakas. Text classification using machine learning techniques. *WSEAS Transactions on Computers*, 4(8):966–974, 2005. 6, 8, 9, 14

[28] Charu C. Aggarwal and ChengXiang Zhai. A Survey of Text Classification Algorithms. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 163–222. Springer US, Boston, MA, 2012. 7, 8, 9, 17, 25, 75

[29] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988. 7, 15

[30] Juan Ramos. Using tf-idf to determine word relevance in document queries. 2003. 7

[31] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002. 8, 9, 14, 17, 20, 25

[32] Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, Khairullah Khan, and Universiti Teknologi Petronas Tronoh. A review of machine learning algorithms for text-documents classification. In *Journal of Advances In Information Technology, VOL*, 2010. 8, 17

[33] F. Kabir, S. Siddique, M. R. A. Kotwal, and M. N. Huda. Bangla text document categorization using stochastic gradient descent (sgd) classifier. In *2015 International Conference on Cognitive Computing and Information Processing(CCIP)*, pages 1–4, March 2015. 8

[34] Armando Segnini and Juanita Joyce Tayou Motchoffo. Random Forests and Text Mining. 8, 21

[35] G Stumme, A Hotho, and B Berendt. Semantic Web MiningState of the art and future directions. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(2):124–143, June 2006. 9

[36] Francisco Garcìa-Sànchez, Rodrigo Martìnez-Béjar, Leonardo Contreras, Jesualdo Tomàs Fernàndez-Breis, and Dagoberto Castellanos-Nieves. An ontology-based intelligent system for recruitment. *Expert Systems with Applications*, 31(2):248–263, August 2006. 9, 10, 33, 74

[37] Malgorzata Mochol, Radoslaw Oldakowski, and Ralf Heese. Ontology based Recruitment Process. In *GI Jahrestagung (2)*, pages 198–202, 2004. 9, 34

[38] Maryam Fazel-zar and Mark S. Fox. Semantic matchmaking for job recruitment: An ontology-based hybrid approach. 9

[39] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993. 10, 22, 23

[40] Michael Grüninger and Mark S. Fox. Methodology for the design and evaluation of ontologies. 1995. 10

[41] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer-Verlag, Berlin, Heidelberg, 2009. 14, 15, 18

[42] Ismail I Hmeidi, Mohammed I Khaleel, and Hassan M Najadat. Automatic text classification: A comparative study. 14, 17

[43] L. Enrique Sucar, Concha Bielza, Eduardo F. Morales, Pablo Hernandez-Leal, Julio H. Zaragoza, and Pedro Larrañaga. Multi-label classification with

bayesian network-based chain classifiers. *Pattern Recogn. Lett.*, 41(C):14–22, May 2014. 14

[44] K. Nithya, P. C. D. Kalaivaani, and R. Thangarajan. An enhanced data mining model for text classification. In *2012 International Conference on Computing, Communication and Applications*, pages 1–4, Feb 2012. 14

[45] Charu C. Aggarwal and ChengXiang Zhai, editors. *Mining Text Data*. Springer US, Boston, MA, 2012. DOI: 10.1007/978-1-4614-3223-4. 15

[46] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011. 15, 26

[47] H. Brücher, G. Knolmayer, and M.-A. Mittermayer. Document classification methods for organizing explicit knowledge. In *Proceedings of the Third European Conference on Organizational Knowledge, Learning, and Capabilities*, Athens, Greece, 2002. 15, 16, 17, 19

[48] Claude Sammut and Geoffrey I. Webb. *Encyclopedia of Machine Learning*. Springer Publishing Company, Incorporated, 1st edition, 2011. 15, 21

[49] Davy Cielen, Arno Meysman, and Mohamed Ali. *Introducing data science: big data, machine learning, and more, using Python tools*. Manning Publications, Shelter Island, NY, 2016. OCLC: ocn910773263. 15, 16

[50] Diana Maynard, S. Dasiopoulou, S. Costache, K. Eckert, H. Stuckenschmidt, M. Dzbor, and S. Handschuh. D1. 2.2. 1.3 Benchmarking of annotation tools. *Knowledge Web Project, University of Sheffield, Technical Report*, 2007. 17

[51] P. Soucy and G. W. Mineau. A simple knn algorithm for text categorization. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 647–648, 2001. 17

[52] William W. Cohen and Haym Hirsh. Joins that Generalize: Text Classification Using WHIRL. In *KDD*, pages 169–173, 1998. 17

[53] David Ferrucci and Adam Lally. Uima: An architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, September 2004. 17

[54] Yiming Yang and Christopher G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Trans. Inf. Syst.*, 12(3):252–277, July 1994. 17, 20

[55] Pascal Soucy and Guy W. Mineau. A simple knn algorithm for text categorization. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 647–648, Washington, DC, USA, 2001. IEEE Computer Society. 17

[56] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000. 17, 18

[57] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. 17

[58] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Madison, WI, 1998. 18

[59] Irina Rish. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM, 2001. 18

[60] Irina Rish, Joseph Hellerstein, and Jayram Thathachar. An analysis of data characteristics that affect naive Bayes performance. *IBM TJ Watson Research Center*, 30, 2001. 18

[61] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2):103–130, Nov 1997. 18

[62] Sang-Bum Kim, Hae-Chang Rim, DongSuk Yook, and Heui-Seok Lim. Effective Methods for Improving Naive Bayes Text Classifiers. In Mitsuru Ishizuka and Abdul Sattar, editors, *PRICAI 2002: Trends in Artificial Intelligence: 7th Pacific Rim International Conference on Artificial Intelligence Tokyo, Japan, August 18Ũ22, 2002 Proceedings*, pages 414–423. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002. DOI: 10.1007/3-540-45683-X_45. 18

[63] Xu LinBin, Liu Jun, Zhou WenLi, and Yan Qing. Adaptive naive bayesian classifier for automatic classification of webpage from massive network data. In *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference on*, volume 1, pages 127–130. IEEE, 2014. 18

[64] Peter Harrington. *Machine Learning in Action*. Manning Publications Co., Greenwich, CT, USA, 2012. 19, 21

[65] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 42–49, New York, NY, USA, 1999. ACM. 19, 20

[66] Thorsten Joachims. Text categorization with Support Vector Machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98: 10th European Conference on Machine Learning Chemnitz, Germany, April 21Ũ23, 1998 Proceedings*, pages 137–142. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. DOI: 10.1007/BFb0026683. 19

[67] Saurav Sahay. Support vector machines and document classification. *URL: http://www-static. cc. gatech. edu/ssahay/sauravsahay7001-2. pdf*, 2011. 19

[68] I. H. Witten and Eibe Frank. *Fast and accurate text classification via multiple linear discriminant projections*. Morgan Kaufmann, San Francisco, Calif, 2000. 19

[69] Yi Lin. Support vector machines and the bayes rule in classification. *Data Mining Knowledge Disc*, pages 259–275, 2002. 19

[70] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012. 19

[71] Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics, 2009. 19

[72] Saswata Chakravarty. Stochastic gradient descent methods for large scale pattern classification. 2011. 20

[73] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 116–, New York, NY, USA, 2004. ACM. 20

[74] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 20, 27, 47

[75] Hinrich Schütze David A. Hully Jan and O. Pedersen. A Comparison of Classifiers and Document Representations for the Routing Problem. *representations*, 15:16. 20

[76] Erik Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th annual symposium on document analysis and information retrieval*, volume 317, page 332. Las Vegas, NV, 1995. 20

[77] Ido Dagan, Yael Karov, and Dan Roth. Mistake-driven learning in text categorization. *arXiv preprint cmp-lg/9706006*, 1997. 20

[78] Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '97, pages 67–73, New York, NY, USA, 1997. ACM. 20

[79] Norbert Fuhr and Ulrich Pfeifer. Probabilistic information retrieval as a combination of abstraction, inductive learning, and probabilistic assumptions. *ACM Trans. Inf. Syst.*, 12(1):92–115, January 1994. 20

[80] David J. Ittner, David D. Lewis Y, and David D. Ahn Z. Text categorization of low quality images. In *In Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 301–315, 1995. 20

[81] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc. 20

[82] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 841–848. MIT Press, 2002. 21

[83] Peter Flach. *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. Cambridge University Press, New York, NY, USA, 2012. 21

[84] Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55, February 1970. 21

[85] Ildiko E. Frank and Jerome H. Friedman. A Statistical View of Some Chemometrics Regression Tools. *Technometrics*, 35(2):109–135, 1993. 21

[86] Erika Cule and Maria De Iorio. Ridge Regression in Prediction Problems: Automatic Choice of the Ridge Parameter: Ridge Regression in Prediction Problems. *Genetic Epidemiology*, 37(7):704–714, November 2013. 21

[87] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. 21

[88] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996. 21

[89] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998. 21

[90] Marina Skurichina and Robert P. W. Duin. Bagging, Boosting and the Random Subspace Method for Linear Classifiers. *Pattern Analysis & Applications*, 5(2):121–135, June 2002. 21

[91] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, Aug 1998. 21

[92] Gérard Biau. Analysis of a random forests model. *J. Mach. Learn. Res.*, 13(1):1063–1095, April 2012. 21

[93] Andreas C. Müller and Sarah Guido. *Introduction to machine learning with Python a guide for data scientists*. O'Reilly, October 2017. 22

[94] Stephen Marsland. *Machine Learning: An Algorithmic Perspective, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014. 22

[95] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. (Advanced Information and Knowledge Processing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. 24

[96] Riichiro Mizoguchi and Mitsuru Ikeda. Towards ontology engineering. *Journal-Japanese Society for Artificial Intelligence*, 13:9–10, 1998. 24

[97] Ellen Riloff. Understanding language understanding. chapter Information Extraction As a Stepping Stone Toward Story Understanding, pages 435–460. MIT Press, Cambridge, MA, USA, 1999. 24

[98] Martin Labskážş, Vojtech SvÃątek, and Marek Nekvasil. Information Extraction Based on Extraction Ontologies: Design, Deployment and Evaluation. *ONTOLOGY-BASED INFORMATION EXTRACTION SYSTEMS (OBIES 2008)*, page 9, 2008. 24

[99] Daya C. Wimalasuriya and Dejing Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 36(3):306–323, 2010. 24, 74

[100] Shi Yong-feng and Zhao Yan-ping. Comparison of text categorization algorithms. *Wuhan university Journal of natural sciences*, 9(5):798–804, 2004. 25

[101] Faizan Javed, Matt McNair, Ferosh Jacob, and Meng Zhao. Towards a job title classification system. In *WSCBD 2014: Webscale Classification: Classifying Big Data from the Web, WSDM Workshop*, 2014. 25, 29

[102] Faizan Javed, Qinlong Luo, Matt McNair, Ferosh Jacob, Meng Zhao, and Tae Seung Kang. Carotene: A Job Title Classification System for the Online Recruitment Domain. pages 286–293. IEEE, March 2015. 25, 29

[103] Stanislaw Osinski and Dawid Weiss. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54, 2005. 26

[104] Flora Amato, Roberto Boselli, Mirko Cesarini, Fabio Mercorio, Mario Mezzanzanica, Vincenzo Moscato, Fabio Persia, and Antonio Picariello. Challenge: Processing web texts for classifying job offers. In *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pages 460–463. IEEE, 2015. 26

[105] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003. 26

[106] Francesco Colace, Massimo De Santo, Luca Greco, and Paolo Napoletano. Improving Text Retrieval Accuracy by Using a Minimal Relevance Feedback. In Ana Fred, Jan L. G. Dietz, Kecheng Liu, and Joaquim Filipe, editors, *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 348, pages 126–140. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. DOI: 10.1007/978-3-642-37186-8_8. 26

[107] Francesco Colace, Massimo De Santo, Luca Greco, and Paolo Napoletano. Text classification using a few labeled examples. *Computers in Human Behavior*, 30:689–697, January 2014. 26

[108] Shilin Zhang, Heping Li, and Shuwu Zhang. Job Opportunity Finding by Text Classification. *Procedia Engineering*, 29:1528–1532, 2012. 27, 29

[109] Noah A. Smith. Log-linear models. *Cited by*, 3, 2004. 27

[110] Emmanuel Malherbe, Mamadou Diaby, Mario Cataldi, Emmanuel Viennet, and Marie-Aude Aufaure. Field selection for job categorization and recommendation to social network users. pages 588–595. IEEE, August 2014. 28

[111] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. 29

[112] David A. Easley and Jon M. Kleinberg. *Chapter 13 The Structure of the Web*. Cambridge University Press, 2010. 30

[113] Andreas Hotho, Alexander Maedche, and Steffen Staab. Ontology-based text document clustering. *KI*, 16(4):48–54, 2002. 30

[114] Jihoon Yang, Prashant Pai, Vasant Honavar, and Les Miller. Mobile intelligent agents for document classification and retrieval: A machine learning approach. In *14th European Meeting on Cybernetics and Systems Research.*

*Symposium on Agent Theory to Agent Implementation, Vienna, Austria*, pages 707–712, 1998. 31

[115] J. P. Burden and M. S. Jackson. WW-Lib-TNG-New direction in search engine technology. In *COLLOQUIUM DIGEST-IEE*, pages 10–10. IEE; 1999, 1999. 31

[116] Charlotte Jenkins, Mike Jackson, Peter Burden, and Jon Wallis. Automatic rdf metadata generation for resource discovery. *Computer Networks*, 31(11):1305 – 1320, 1999. 31, 32

[117] Melvil Dewey. *Dewey decimal classification and relative index*. OCLC Online Computer Library Center, Dublin, Ohio, 22nd edition, 2003. 31

[118] Library of Congress Bibliographies. Library of congress. 31

[119] Dieter Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2 edition, 2003. 31

[120] Ian Horrocks, Dieter Fensel, Jeen Broekstra, Stefan Decker, Michael Erdmann, Carole Goble, Frank van Harmelen, Michel Klein, Steffen Staab, Rudi Studer, et al. The ontology inference layer oil, 2000. 32

[121] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3c recommendation, W3C, February 1999. 32

[122] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003. 32

[123] J. Fagan. Automatic phrase indexing for document retrieval. In *Proceedings of the 10th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '87, pages 91–101, New York, NY, USA, 1987. ACM. 32

[124] Michael W. Berry and Murray Browne. *Understanding Search Engines: Mathematical Modeling and Text Retrieval (Software, Environments, Tools), Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005. 32

[125] Kimbro Staken. Introduction to native xml databases, October 2001. 32

[126] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure. Semantic community web portals. *Computer Networks*, 33(1-6):473–491, jun 2000. 33

[127] Sheng-Yuan Yang. OntoPortal: An ontology-supported portal architecture with linguistically enhanced and focused crawler technologies. *Expert Systems with Applications*, 36(6):10148–10157, aug 2009. 33

[128] G. van Heijst, A.Th. Schreiber, and B.J. Wielinga. Using explicit ontologies in KBS development. *International Journal of Human-Computer Studies*, 46(2-3):183–292, feb 1997. 33

[129] W3C Recommendation 10 February 2004. Resource description framework (rdf): Concepts and abstract syntax. February. 35

[130] Jim Hendler Ian Horrocks Deborah L. McGuinness Peter F. Patel-Schneider Lynn Andrea Stein Sean Bechhofer, Frank van Harmelen. Owl web ontology language - reference. November 2009. 35

[131] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 441–450, New York, NY, USA, 2010. ACM. 38, 51

[132] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent, 2009. 48

[133] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. 1999. 67, 72, 76

[134] Neha Mehra and Surendra Gupta. Survey on multiclass classification methods. 72

[135] Jingnian Chen, Houkuan Huang, Shengfeng Tian, and Youli Qu. Feature selection for text classification with naïve bayes. *Expert Systems with Applications*, 36(3, Part 1):5432 – 5435, 2009. 73

[136] Zi qiang Wang, Xia Sun, De xian Zhang, and Xin Li. An optimal SVM-based text classification algorithm. In *2006 International Conference on Machine Learning and Cybernetics*. IEEE, 2006. 73

[137] Cristina Niculescu and Stefan Trausan-Matu. An ontology-centered approach for designing an interactive competence management system for IT companies. *Informatica Economica*, 13(4):159, 2009. 74

[138] Raghu Anantharangachar, Srinivasan Ramani, and Rajagopalan S. Ontology Guided Information Extraction from Unstructured Text. *International journal of Web & Semantic Technology*, 4(1):19–36, January 2013. 74

[139] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420, 1997. 75

[140] Elena Montañés, Javier Fernández, Irene Díaz, Elías F. Combarro, and José Ranilla. *Measures of Rule Quality for Feature Selection in Text Categorization*, pages 589–598. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. 75

[141] S. Ganesh, M. Jayaraj, V. Kalyan, SrinivasaMurthy, and G. Aghila. Ontology-based web crawler. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, volume 2, pages 337–341 Vol.2, April 2004. 76

[142] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. The MIT Press Cambridge, Massachusetts London, England, 2004. 76

[143] Filippo Menczer. Arachnid: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery, 1997. 76

[144] Filippo Menczer and Richard K. Belew. Adaptive information agents in distributed textual environments, 1998. 76

[145] Mukesh Kumar and Renu Vig. Design of core: context ontology rule enhanced focused web crawler. In *ICAC3 '09: Proceedings of the International Conference on Advances in Computing, Communication and Control*, pages 494–497, New York, NY, USA, 2009. ACM. 76