2017-9

An Analysis of Predicting Job Titles Using Job Descriptions

John Lynch
Technological University Dublin

An Analysis of Predicting Job Titles Using Job Descriptions

John Lynch

A dissertation submitted in partial fulfilment of the requirements of
Dublin Institute of Technology for the degree of
M.Sc. in Computing (Data Analytics)

30 June 2017

Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institutes guidelines for ethics in research.

Signed:

Date:

Abstract

A job title is an all-encompassing very short form description that conveys all of the pertinent information relating to a job. The job title typically encapsulates - and should encapsulate - the domain, role and level of responsibility of any given job.

Significant value is attached to job titles both internally within organisational structures and to individual job holders. Organisations map out all employees in an organogram on the basis of job titles. This has a bearing on issues like salary, level and scale of responsibility, employee selection and so on. Employees draw value from their own job titles as a means of self-identity and this can have a significant impact on their engagement and motivation.

Classification of job titles based upon the details of the job is a subjective human resources exercise, however, which risks bias and inconsistency. I am instead proposing that the job title classification process can be performed in a systematic, algorithmic-based process with the application of standard Natural Language Processing (NLP) together with supervised machine learning.

In this paper, data (job descriptions) labelled with Job Titles was collected from a popular national job postings website (www.irishjobs.ie). The data went through several standard text-pre-processing transformations which are detailed below, in order to reduce dimensionality of the corpus of data. Feature engineering was used to create a Data Model(s) of selected keyword sets characteristic to each Job Title generated on the basis of term frequency. The models developed with the Random Forest and Support Vector Machines supervised learning algorithms were used to generate prediction models to make predictions based on the Top 30 most frequently occurring Job Titles.

The most successful model was the SVM linear kernel based model, which had an Accuracy rate of 71%, Macro Average Precision of 70%, Macro Averaged Recall of 67% and a Macro Average F-Score of 66%. The Random Forest Model performed less well; with a Accuracy rate of 58%, Macro Average Precision of 56%, Macro Average Recall of 55% and Macro Average FScore of 56%.

The data model described here and the prediction performance obtained indicate that several particularities of the problem its high dimensionality and the complexity of feature engineering required to generate a data model with the correct keywords for each job lead to data models that cannot provide an optimal performance even when using powerful Machine Learning (ML) algorithms.

The data model design can be improved using a wider data set (completed from job descriptions collected from a variety of websites) thus optimising the set of keywords describing each job title. More complex and computationally expensive algorithms - based on deep learning - may also provide more refined and more accurate predictive models.

No research was found during this study which examined the subject matter of classification of job titles using machine learning specifically. However, other relevant literature was reviewed on text classification via supervised learning which was useful in designing the models and applied to this domain.

While supervised ML techniques are commonly applied to text classification including sentiment analysis, there was no similar study described in the literature approaching the link between job titles and the corresponding required skills. Nevertheless, the work presented here describes a valid and practical approach to answering the proposed research question within the constraints of a limited data model and basic ML algorithms. Such an approach may prove a working base for designing future models for artificial intelligence applications.

Keywords: Job titles, random forest, support vector machines, supervised learning

Acknowledgements

This paper would not be possible without the help of a few people.

Firstly, thanks to my supervisor, Tamara Matthews, whose patience was tested with this student's headstrong nature, and who imparted valuable advice and guidance throughout the process.

Secondly, to my partner, who gave birth to our wonderful little girl in the latter stages of the Msc programme. She carried more than her share of the burden that comes with parenthood so that this new father could devote the time needed to this paper and the subjects studied. Her contribution did not end there; she was also an invaluable proof-reader and source of advice and encouragement throughout.

Contents

List of Figures

List of Tables

List of Acronyms

ABT	Analytics Base Table
AI	Artificial Intelligence
BI	Business Intelligence
DDL	Data Definition Language
IT	Information Technology
ML	Machine Learning
NLP	Natural Language Processing
RF	Random Forest
SQL	Structured Query Language
SVM	Support Vector Machines

Chapter 1

Introduction

1.1 Background

Job titles have a very important role in organisations. The small number of words that make up a job title are supposed to synopsise all that there is about a certain role and convey its full import. Job titles typically encapsulate the specific field, skills, and level of responsibility of that particular function in the organisation (Cable, Grant, & Berg, 2013). By labelling people with job titles, an organisation now has the means to cluster people into departments of related activities.

Job titles are also the source of individual workers' self-identity. Within the organisation, we use business cards and/or email signatures with no more than our name and job title and this information imparts to the reader a significant amount of information. Outside of the organisation, the typical answer to "what do you do?" is often relaying one's job title; and not other descriptors such as "married father of two, coach at an under-age football club, like to read and travel...". We do not have an equivalent very short form synopsis for our personal life and therefore our professional one carries great significance.

In more recent times, the phenomenon of job title inflation has occurred. One study found that 46% of executives received a promotion that came with a new job title although the responsibilities largely remained unchanged. It was used as a tool during recessionary times to motivate and retain employees when the financial means

were not there to do so (Giancola, 2014). Related to job title inflation is the concept of self-titling. Self-titling is where an employee devises their own job title in order to convey the value that they believe they bring to the organisation. This is another low cost tool to increase employee motivation and engagement which feeds into the social currency we attach to job titles (Grant, Berg, & Cable, 2014).

The irony of self-titling/job title inflation to add more value to self-identity makes classification a more difficult exercise. For example, when a job candidate seeks a new role, the normal action when perusing a job postings website is to type in the candidate's own job title or desired job title into the search feature, get a list of results and progress from there. With embellished or alternate job titles, this process is less effective as a keyword search will not yield all of the job titles.

Traditionally, organisational structure had a rigid wording structure. For example, specific words denote order of seniority such as: junior to associate to senior to vice president to president. The next aspect is the department; for example, finance, IT or marketing. Modern titles belie this structured approach; where would a job title such as "Brand Defender" fit in for example?

Even outside of the modern trend towards non-traditional job titles, there is also the issue of lack of consistency in job titles across multiple organisations. An organisation may have rules in place for the naming convention of job titles but these rules will differ from organisation to organisation. For example, an organisation may use a generic term of "Database Administrator" as a catch-all title for individuals who maintain databases. Whereas another organisation may have distinct titles that are application specific; such as "Oracle Developer" or another separate title such as "SQL Server Developer". All three could be highly correlated in terms of their inherent roles and responsibilities, with only the platforms differing. But in the case of the generic title, the platform is not specified or relevant.

1.2 Research Project/Problem

Generally, the classification of an employee or potential employee's job would typically be performed as a subjective exercise based on a visual read of the job details by someone such as a HR Manager or Departmental Manager. This approach can lead to personal interpretation, bias and inconsistencies.

The problem that this study is trying to address is how best to classify job details to a job title through the application of supervised machine learning techniques, thus bringing a systematic and algorithmic-based approach to what can often be a subjective and inaccurate process.

The job title correlates to the suitable placement of an individual within an organisation's hierarchy. It is important to the organisation that this exercise is accurate, to ensure allocation of correct salary levels and avoid underutilisation of employees in inappropriate roles. For the individual, it is imperative that they are being remunerated at the right level and that they are recognised appropriately for their individual skill-sets which otherwise can be demotivating and stressful.

Supporting processes and procedures which provide for greater accuracy of job titling can also present an opportunity for the individual to upskill and enhance their earnings potential. It can assist the organisation's selection process by screening out unsuitable candidates.

The research question being posed is therefore as follows:

To what extent is the prediction of a job title from a job description possible through the use of supervised learning techniques?

The research question will be investigated across this work observing the implications towards building a predictive model.

1.3 Research Objectives

The main objectives of this dissertation are as follows:

- To study the link between job titles and job descriptions, and the extent to which

predictions can be made of a job title using job details.

- To build a model and use supervised learning algorithms to make these predictions.
- To assess the outcomes of parameter tuning of selected algorithms and alternate feature selection.

1.4 Research Methodologies

This paper will be based on both primary and secondary research.
The primary research will be in the form of the experiments conducted to produce supervised learning-based predictive models, with a quantitative analysis of the outcomes of these models.
The secondary research will be in the form of the literature review which was undertaken in order to observe and present any relevant background research in this field.

1.5 Scope and Limitations

The data used will be secondary data in nature which has been collected from a jobs posting website. The data in question was collected on multiple occasions between December 2016 and May 2017 from a popular jobs posting website. The website is not industry or sector specific and operates almost entirely for the Irish job market.
During the experiment phase, the data was restricted to the top 30 most frequently occurring job titles.

1.6 Document Outline

The dissertation is organised as follows:

Chapter Two covers the secondary research conducted based on a review of relevant literature in the areas of data analytics, collection of web-based unstructured data, text processing and categorisation and text analytics.

Chapter Three focuses on the experiment design and planned implementation. Covering the data collection process, an explanation of the text pre-processing and machine learning algorithms is presented.

Chapter Four examines the implementation process in depth. Some of the challenges faced during the implementation phase are addressed.

Chapter Five presents the results, with an evaluation on the findings.

Chapter Six concludes the dissertation and provides an overview of the whole experiment. This chapter also presents possible areas for further research.

Chapter 2

Review of existing literature

2.1 Introduction

A number of studies were reviewed in order to provide a theoretical basis for the experiment and to learn which processes and tools would be most suitable for the research problem in question. Text analytics can provide insights into a variety of data sources over the internet and a number of studies reviewed highlight how the internet is a source of genuine and time-sensitive data for BI. Given that job titles and job descriptions have particular characteristics (neutral and factual language), studies which analysed similar types of data were found to have most relevant findings in terms of appropriate algorithms which could be replicated or employed for the purposes of this study. The most relevant research which informed my own findings is summarised below, and is organised according to the chronology of the methodology followed.

2.2 State of the Art

In a survey in the field of web mining by Kosala and Blockeel, the paper acknowledges the fact that with conventional search tools a list of results with low precision and recall are returned. The paper proposes a potential framework for web mining including the use of NLP and machine learning for the purposes of mining of web content for better search outcomes (Kosala & Blockeel, 2000).

Guo et al conducted research building an engine that matches resums to job postings (Guo, Alamudun, & Hammond, 2016). From on-line job postings, a job model is constructed based on the job title, area of study, level of education and competences. A similar process is employed with candidate resums. Similarity distances are calculated on each of the four components of the job model. This research proved useful in terms of relevance to the research problem in my study as it noted how the process of matching can be done on the basis of specially-designed models of job characteristics.

In their study, Nigam Lafferty & McCallum examine the application of maximum entropy (ME) on three differing datasets in order to to make multi-class predictions (Nigam, Lafferty, & McCallum, 1999).One dataset was the WebKB; this is a collection of data from web pages taken from various universities classified into seven types of web page. The data consisted of 4,199 pages with 23,830 words. The second dataset was the Industry Sector Hierarchy; this is a collection of 6,440 web pages classified by industry sector, 71 classes. This corpus contained 29,964 words. The third dataset was the Newsgroups dataset, a collection of 20,000 articles categorised into 20 discussion groups. This corpus contained 57,040 words. The classification error rate varied from 7.9% for the WekKB, to 15.8% for the Newsgroups dataset and 21.1% for the Industry Sector dataset. Given the wide application and ease of use of this algorithm, I attempted to apply similar maximum entropy on the initial dataset for the experiment set out in this paper. However, the attempt was not successful, as set out in the Method and Implementation section below.

In research conducted by Cavnar and Trenkle, the authors employed the use of n-grams for subject classification (Cavnar & Trenkle, 1994). The dataset was 778 articles from five Usenet newsgroups, with the textual data contained within seven categories of FAQs used as labels. The classification results varied from 30% to 80%. The lowest classification rate of 30% was for subject matter AI (Artificial Intelligence). The researchers reasoned that given that the textual data in the FAQ for AI was by far the largest subset of the FAQs - making up 33% of the whole FAQ dataset - this subset was too large and affecting the results, as they are grounded on the frequency of n-grams.

A comparative study was conducted of different n-gram types on a dataset of 50,000 movie reviews in the IMDB database, across four different algorithms; Nave Bayes, Maximum Entropy, Support Vector Machines and Stochastic Gradient Descent (Tripathy, Agrawal, & Rath, 2016).The observation made is that the accuracy of the algorithms decreases as the value of n in the n-grams increases. Where n is equal to 3 or greater, the accuracy rates decrease. The authors note that a weakness of unigram is that a sentence with "not good" is classified as neutral (one positive and one negative word) where in fact "not good" is an entirely negative sentiment. This is where bi-grams are more informative than unigrams for detecting true emotion. Both pieces of research were particularly useful when I was developing the initial model as it was immediately clear that application of n-grams would be unsuitable for the purposes of my experiment given that the body of text in job details is generally neutral and does not carry any emotion/sentiment.

A further study explored how Nave Bayes and SVM as classification rates of binary outcomes were benchmarked against each other (Bermingham & Smeaton, 2010).The datasets involved consisted of short form textual data from microblogging and micro review websites and longer form textual data in the form of movie reviews from Usenet and blog postings. SVM were found to perform better on the longer-form text and therefore increased dimensionality. The authors applied a linear kernel SVM with cost=1 in their SVM model. In reviewing different algorithms in the course of my research, SVM was found to be the most appropriate classifier for this experiment due to the long-form text of the subject matter - job descriptions are text-heavy. SVM can handle the high dimensionality associated with text mining of such long-form text.

In research conducted by Chinese researchers, classification of spam emails was examined in both English and Chinese, using the "lingsam" data set circa 3,000 emails and the CERENT email dataset of which 110 were selected (Xiu-li, Yu-qiang, & Wei, 2007).The approach taken was to vectorise the dataset, with the words of each email being the x value and the y value being a binary classifier to indicate for spam or not. The precision of various algorithms was compared under various states of text pre-processing. The SVM algorithm was found to consistently have a higher precision

value than other algorithms such as Maximum Entropy, and variants on Nave Bayes. With a SVM comparison conducted of different kernel types and parameter settings, the linear kernel matched or performed better than the various polynomial kernels. Furthermore, both linear and polynomial performed better than a radial kernel. All three kernel types are examined in the context of my experiment in the methodology section below.

Somprasertsi & Lalitojwong conducted a study to separate out product features from product opinion using part of speech tagging with Maximum Entropy on a dataset of reviews of digital cameras from www.amazon.com (Sompras & Lalitrojwong, 2010).The dataset contained 1,250 sentences in total. Noun words and noun phrases were considered to be product feature words and adjectives were opinion words. This allowed for a process of differentiation which I replicated when cleaning job titles in the course of my experiment.

In their study, Wang and Manning conducted SVM and NB analysis on 8 different datasets of varying length; from one sentence review to longer form reviews and other textual datasets (Wang & Manning, 2012). They concluded that bi-grams improved performance of sentiment analysis. Furthermore, the findings concluded that SVM performed better than Nave Bayes on longer form text, as I inferred in relation to job descriptions in the course of my own experiment.

Similar research was conducted by Koprinska et al in which they used two email datasets of 1,099 cases and 2,893 cases each (Koprinska, Poon, Clark, & Chan, 2007).The authors found that with features selection based on information gain, Random Forest out-performed in terms of classification accuracy and F1 score compared to three other models (Decision tree, SVM and Nave Bayes) at classifying spam on both datasets. On features selected on the basis of term frequency the Random Forest was second in performance based on those same metrics.

In their research, Noh et all conducted experiments on keyword strategy in the domain of patent document analysis (Noh, Jo, & Lee, n.d.). A review of the existing research found that the number of meaningful keywords tended to be around 30 keywords. Words in patents are very formal rather than natural language paradigms,

much like a job description is of a more formal nature. This is reflected in other research Onan, Korukolu, and Bulut (2016) conducted on a corpus from ACM of 3,606 documents broken into eight datasets. The authors assessed the number of keywords - ranging from 5 to 100 - across 21 different algorithm and ensemble models with five different feature selection strategies. The sharpest increase in accuracy was in the range keyword range 5 to 25 keywords, going from circa 0.62 to 0.83 accuracy respectively. The best accuracy was achieved in the 80-85 keyword range at an accuracy value of 0.93. A similar trend was seen with the F1 scores with the sharpest improvement up to 25-30 words with 85 being the best F1 score. However, the superior accuracy and F1 score of 80-85 over 25-30 keywords is potentially attributable to a higher level of dimensionality. In the course of my experiment, I therefore attempted to build a model using feature extraction of 30 key words on the basis of this research, tailored to the very long-form text in job descriptions. Going too high would have the potential to over-fit the model.

In an analysis of performance measures for classification tasks Sokolova and Lapalme state that for a multi-class classification experiment - an experiment where a single class is selected from a number of classes - two options are available (Sokolova & Lapalme, 2009). For the overall performance assessment these two options are macro averaging and micro averaging. Macro averaging is a simple average of the results of the individual class where micro is a weighted average-based measure. No conclusions have been made on the suitability of one over the other. The study notes in relation to text classification (and in a binary classification context) that the trend is to use precision and recall as performance measures. The justification is that negative classes in text classification is not a genuine negative with some underlying property(ies) that makes these cases negative. Negative is negative only because its not positive. Precision and recall measures ignore true negatives and therefore their suitability for text classification. When applying text classification to the data in my research, precision and recall measures are the basis of assessment. Accuracy is also important as found in many of the above studies.

In conclusion, the reviewed literature covered a range of supervised machine learn-

ing algorithms used in text mining, and evaluation. Some of the key takeaway's for the experiment design are;

- Support Vector Machines and Random Forest are frequently used supervised learning algorithms in text mining.
- N-grams are most useful for detecting sentiment and not a suitable approach for this experiment as the tone of job description textual data is neutral.
- For long form text mining that 25-30 words was optimum number of key features for modelling purposes and strikes a balance between accuracy and potential model over-fit.
- Precision and Recall are particularly apt for evaluation of text mining. This is because in text mining a negative (binary) outcome is not a genuine negative. Precision and Recall are measures orientated on True Positive outcomes.
- Evaluation of multi-class outcomes can be converted to single metric with Micro or Macro averaging. Neither of these two options is deemed to be more applicable than the other.

Chapter 3

Experiment design and methodology

3.1 Introduction

This chapter will cover the experiment's design in detail. There are four distinct phases to the experiment. This process is illustrated in figure 3.1.

1. Data is collected through web scraping of the job advertisements from the source website.

2. This is followed by the text pre-processing stage which includes transformations of the gathered data such as removal of stop words, special characters, and so on. The transformed data is then split into the training and testing datasets.

3. The process of feature extraction follows; whereby a list of keywords is generated using term frequency as a basis to identify candidate features.

4. Finally, the last step focuses on the generation of models from the features extracted using the Random Forest (RF) and Support Vector Machines (SVM) supervised learning algorithms. The test dataset is applied to the models to make predictions and the results are then evaluated.

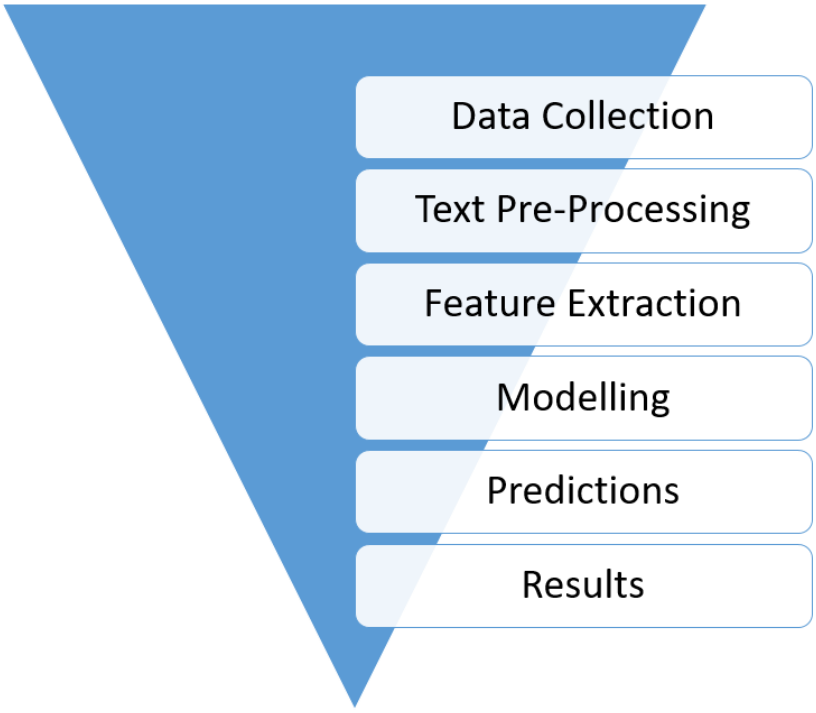

Figure 3.1: Overview of experiment design

3.2 Data Collection

The data was collected via web scraping from www.irishjobs.ie; a popular Irish job advertisement website. This website was selected as it has a broad range of job postings (10,294 total postings active as of 1 June 2017) spanning many industries including finance, IT and hotel and catering; examples of the 28 different categories available.

In the suite of developer tools in the *Microsoft Edge* web browser, a feature called the *DOM Explorer* allows for navigation of the HTML elements, see figure 3.2. From this the pertinent elements of a web page can be determined. This HTML information can be applied into the web scraping tool to collect the data.

The web scraping is done using the *rvest* package on *r*. A widely used web-scraping tool is *beautiful soup* on *Python*, the decision of using *rvest* over *beautfiul soup* was to keep all the processing end to end on a single application - *r* in this case (Wickham & RStudio, 2016).

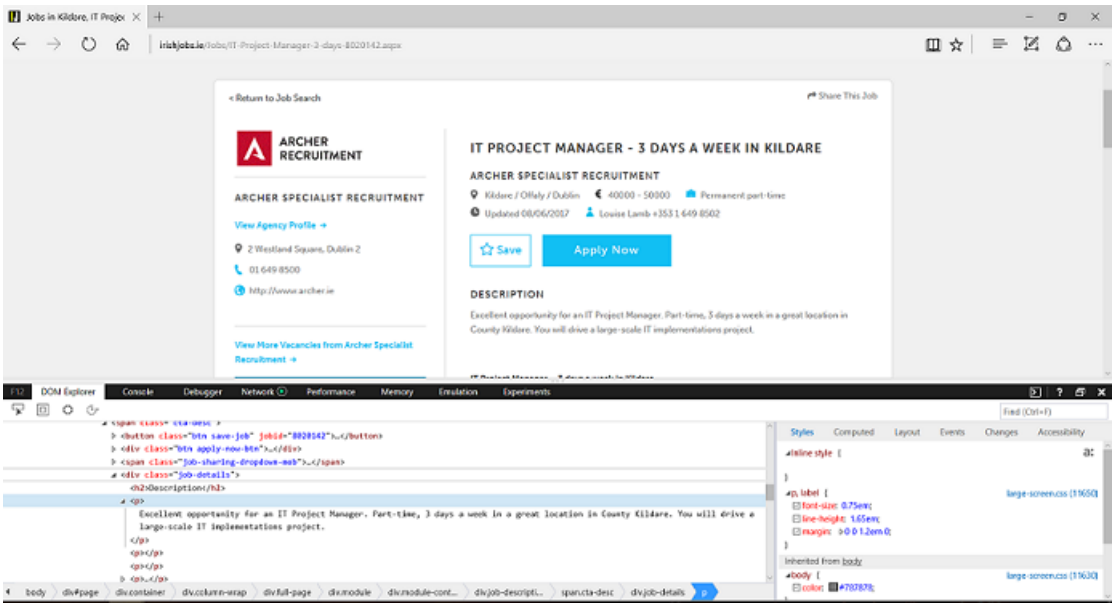

Figure 3.2: Microsoft Edge DOM Explorer

3.3 Text pre-processing

The next step is text pre-processing. This standard practice is largely an act to reduce dimensionality without affecting the effectiveness of the model. It took a number of forms; as summarised below.

3.3.1 Lower Case

The goal here is to standardise all characters to avoid case sensitive error. Some functions are case sensitive and a means to reduce the risk of error is to standardise all of the text to lower case. In this particular experiment, case is not meaningful to the model and it is therefore acceptable to apply lower case to all of the data.

3.3.2 Special characters

For the purpose of this experiment, special characters do not carry any particular significance in predicting a job title. Therefore, it is acceptable to remove special characters without influencing the effectiveness of the model.

3.3.3 Stop words

Stop words are words such as "the", "I", "that". Stop words carry no predictive influence of the model and can also be removed.

3.3.4 Whitespace

Any extra whitespace may cause a word to be interpreted incorrectly. In normal circumstances a word will follow a single space. A double space may lead to a word being concatenated with the extra space. For example, " today" instead of "today". Whitespace removal is simply the stripping out of instances of these extra spaces.

3.3.5 Stemming

Stemming is the process of reduction of words by different forms of the same words. For example *organise, organises, organising*. For most text mining applications this family of variants could be considered one of the same, the root word of *organise* is representative of the others. Stemming is the process of this type of reduction. The *r* package *SnowballC* Bouchet-Valat (2014) via a wrapper in the package *tm* uses the Martin Porter stemming algorithm for this operation (M.F. Porter, 2006).

3.3.6 Holdout datasets

There is no definitive guidance as to appropriate levels of splitting holdout datasets, as observed during a review of the relevant research. It can range from 50:50 to 90:10. For this experiment the ratio is 70:30, on a stratified basis on the labels (job titles).

3.3.7 Feature extraction

Feature extraction is the process of deriving/extracting informative values from the data. Term frequency is a common approach for extracting features in text mining. This is the process of extracting the most frequently used terms in any given corpus. A frequently used term may be an informative feature and support the modelling

process. Term frequency is also a relatively easy approach to execute compared to other feature extraction strategies such as importance values which can be generated with a random forest model.

The end point is to produce an Analytics Base Table (ABT), see table 3.1 for illustration of an ABT. This is a table consisting of the target (in this experiment the target is a numeric representation of a job title) and then a list of features which are the key terms that represent the target and facilitate the predictive process.

Target	**Term1**	**Term2**	**Term3**	**Term4**	**Term5**	**Term6**	**. . .**
41	accounting	work	team	accountant	will	financial	...
. . .							

Table 3.1: Analytics base table

A wordcloud visualisation of the ABT is presented in figure 3.3.

Figure 3.3: Wordcloud representation of ABT

3.4 Algorithms

3.4.1 Random Forest

Random Forest is an ensemble algorithm that can perform both classification and regression. The ensemble is generated with bootstrap samples of the training data. A random set of features is selected at the tree splitting stage. The predictions of all the individual trees are averaged out to get the optimal prediction. This approach reduces the impact of highly predictive variables and therefore of all the trees being highly correlated to each other (Breiman, 2001).

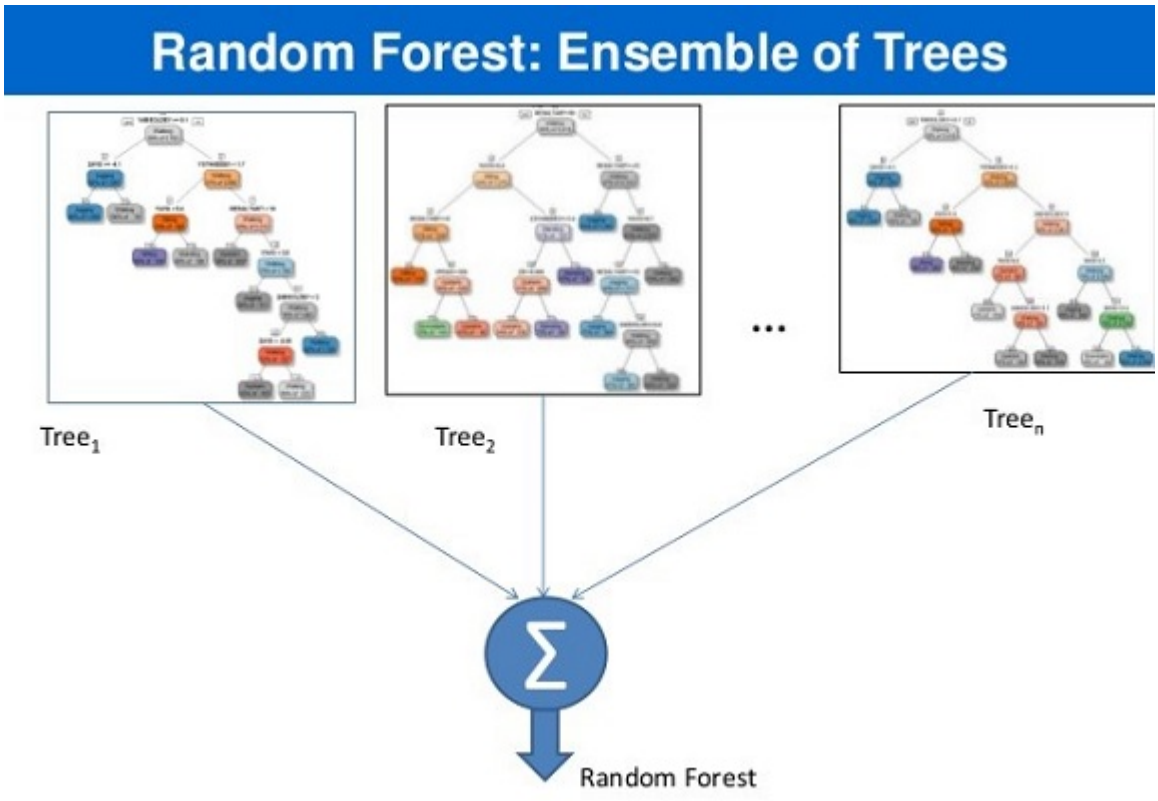

Figure 3.4: Illustration of Random Forest Algorithm

[1]

[1] https://www.slideshare.net/satnam74/india-software-developers-conference-2013-bangalore

3.4.2 Support Vector Machines

Support Vector Machines are a classification and regression algorithm. A SVM is a representation of the variables in a hyperspace. The prediction is made on the basis of determining the decision boundary which is a hyperplane that separates one class from another (Cortes & Vapnik, 1995). See figure 3.5 for a two dimensional illustration of SVM.

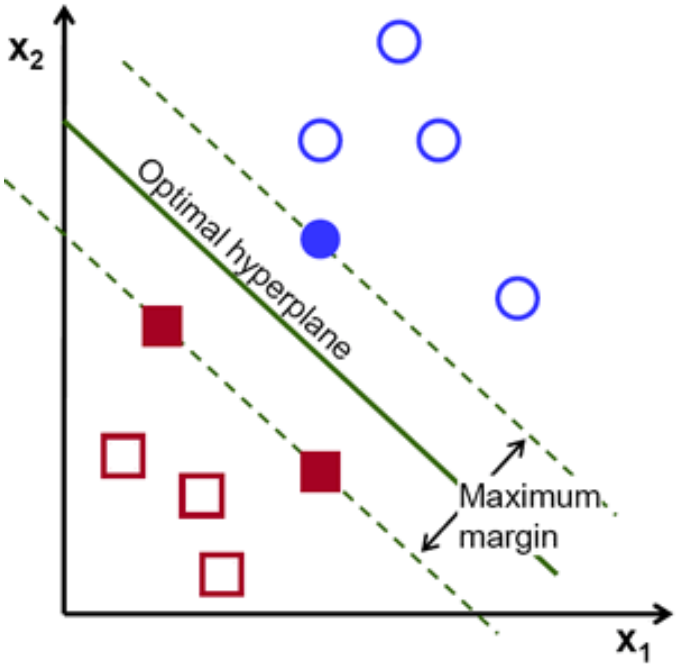

Figure 3.5: 2D Illustration of SVM

[2]

3.5 Basis of evaluation

The comparison of results has been done on the basis of the accuracy, precision, recall and F1 score in line with the corresponding research in the literature review Sokolova and Lapalme (2009) on multi-class classification for text classification.

[2]http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

Accuracy

$$\frac{\sum_{i=1}^{l} \frac{tp_i+tn_i}{tp_i+fn_i+fp_i+tn_i}}{l} \tag{3.1}$$

true positives are denoted as tp_i, true negatives are denoted as tn_i, false negatives are denoted as fn_i, false positives are denoted as fp_i. l is the per class identifier

Precision$_{\mathbf{M}}$

$$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fp_i}}{l} \tag{3.2}$$

true positives are denoted as tp_i, false positives are denoted as fp_i, l is the per class identifier

Recall$_{\mathbf{M}}$

$$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fn_i}}{l} \tag{3.3}$$

true positives are denoted as tp_i, false negatives are denoted as fn_i, l is the per class identifier

Fscore$_{\mathbf{M}}$

$$\frac{(\beta^2+1)Precision_M Recall_M}{\beta^2 Precision_M + Recall_M)} \tag{3.4}$$

The definitions of true positive, true negative, false positive and false negative are defined as per table 3.2.

		Predictions	
		Positive	**Negative**
Actual Class	**Positive**	TP	FN
	Negative	FP	TN

Table 3.2: Binary confusion matrix

3.6 Data storage

For the persistent storage, an *SQLite* database was employed. *SQLite* is a free open source *SQL* (Structured Query Language) engine. It can interface directly with *r* via the *SQLite* package and can also access the database direct via *DB Browser for SQLite*. A major advantage is having all of the data stored within a single sqlite file, rather than individual text files for each instance which is the more traditional approach.

The SQL DDL (Data Definition Language) statement to create the single table called scrapedata used to hold all the scraped data is as follows:

```
CREATE TABLE scrapedata
(`URL` TEXT, `Title` TEXT, `EmploymentType` TEXT,
`Salary` TEXT, `Location` TEXT, `Details` TEXT);
```

3.7 Hardware & Software

Name	Version	Function	URL
R	3.2.3	Statistical Computing	https://www.R-project.org/
R Studio	0.99.491	IDE for R	https://www.rstudio.com/
Rvest	0.3.2	Web scraping tool	https://CRAN.R-project.org/package=rvest
tm	0.7-1	Text mining package	https://CRAN.R-project.org/package=tm
RSQLite	1.1-2	SQLite interface	https://CRAN.R-project.org/package=RSQLite
sqldf	0.4-10	Perform SQL select queries	https://CRAN.R-project.org/package=sqldf
magrittr	1.5	Pipe operator	https://CRAN.R-project.org/package=magrittr
stringr	1.2.0	String manipulation	https://CRAN.R-project.org/package=stringr
dplyr	0.5.0	Data manipulation	https://CRAN.R-project.org/package=dplyr
caret	6.0-76	Classification & regression modelling	https://CRAN.R-project.org/package=caret
e1071	1.6-8	SVM modelling	https://CRAN.R-project.org/package=e1071
randomForest	4.6-12	Random forest modelling	http://CRAN.R-project.org/doc/Rnews/
SnowballC	0.5.1	Stemming based on Porters algorithm	https://cran.r-project.org/web/packages/SnowballC/index.html
wordcloud	2.5	Create word clouds	https://cran.r-project.org/web/packages/wordcloud/
Microsoft Edge	38.14393.1066.0	Web Browser	https://www.microsoft.com/en-ie/windows/microsoft-edge
DB browser for Sql Lite	3.9.1	SQLite db interface	http://sqlitebrowser.org/
Windows 10			
Dell XPS	Intel Core i5-4210U	1.70 GHz, 4.00 GB Ram	

Table 3.3: Software and Hardware used

Table 3.3 is a list of the software used, the packages installed within that software, and hardware specifications of the client machine.

Chapter 4

Implementation and results

4.1 Data Source

Exploring a sample job advertisement from the aforementioned website using *DOM Explorer*, the following key HTML elements were identified as per table 4.1.

Job element	**HTML type**	**Description**
Job description		Wrapper div containing all of the components below
Job title	<h1>	Job title
Employment type		Contract/temporary/Permanent
Salary		Salary of the position
Location		Location of the role
Job details		Wrapper for all the other particulars. Typically this would be a description of the role, responsibilities and requirements

Table 4.1: HTML Nodes

The important features are the job title - which will provide the labels for the modelling experiment - and the job details - which will provide the text/features used to train and test the models. See excerpt of r script below, beginning with the code

for parsing a web page with the *read_html* command. The *html_nodes* command select a node(s) and in this case the *h1* node for job title and *.job-details* node for the job details. The *html_text* command extracts the text from the node(s).

```
#read in the URL
link <- read_html(url)

#read the job title
linktitle <- html_nodes(link, "h1")
#Get the text of the job title
linktitle <-html_text(linktitle)

#read the job details
linkdetails <- html_nodes(link, ".job-details")
#Get the text of the job details
linkdetails <- html_text(linkdetails)
```

The <*div*> for job details was free-form in nature. Therefore there were no pre-defined subsets of differing aspects such as a section for role description, a section for job requirement, and so on. It was entirely at the user's discretion as to the design of the contents in the job details div. With the result of this HTML design, the only option is to scrape the entirety of this data with no natural boundaries.

4.1.1 Web search

To scrape the specific pages a list of URLs needs to be derived for the individual page scraping. A vanilla search of all jobs was generated on the website's own search tool. An example of the URL created by the search enquiry is as below.

`http://www.irishjobs.ie/ShowResults.aspx?Keywords=&Location=0&Category=&btnSubmit=+&PerPage=100`

The search feature had a maximum of one hundred results. However, the URL can be altered to return a larger number of results by changing the numeric value in this segment *&PerPage=100*. Despite the ability to run a single search of greater than one hundred, there was a security measure in place, and it was not possible to scrape any of the URLs when the value was greater than one hundred. To return a search of the next 101-200 postings, this was achieved by adding *&Page=2* to the end of the URL and thereby it was possible to iterate hundreds of postings.

`http://www.irishjobs.ie/ShowResults.aspx?Keywords=&Location=0&Category= &btnSubmit=+&PerPage=100&Page=2`

Subsequently, *rvest* commands *read_html* were used to to parse the web page and html_nodes to search for the node *.job-result-cta a*. Using the *html_attrs* command, a list of the attributes of the node were returned; see example below.

```
[[1]]
                                                                                                   class
                                                                                              "save-job"
                                                                                                   style
"cursor: pointer; background-color: #ffffff; color: #cad466; background-image: url('/img/icons/star-green-small.png');"
                                                                                                   jobid
                                                                                               "8021552"

       class                                          href
"show-more" "/Jobs/Technical-Business-Analyst-8021552.aspx"
```

Figure 4.1: HTML Attributes of *.job-result-cta a*

Each job has two sets of attributes. The first are attributes on visual appearance, colour, image and so on. The second attribute is the href value, and this gives a truncated version of an URL of an individual posting. The truncated URL is separated from the other attribute details by pushing the full list of attributes into a data frame and applying a subset command to slice" of only the URL data. After a list of URLs is extracted, a check is performed against previously scraped URLs to produce a resultant

list of only new candidate URLs. In the r coding, there are two loops encoded. The first loop is for the search aspect and extraction of the list of URLs to scrape. The second loop takes this list of URLs and individually scrapes for the job title and job details.

4.2 Text pre-processing

4.2.1 Titles

Job titles will be the labels for the supervised algorithms. Upon initial review, it was observed that the job titles frequently had other non-job title related information contained in them. See below sample of eighteen unique job titles from a *sqldf* statement on the scraped database for a title containing "financial accountant". Fifteen of the titles contained a location and/or industry type. The remaining three had other information to convey some other aspect of the job.

The most likely explanation is that when conducting a job search on the website, the font of the job title is much larger than the rest of the other detail so this tactic is used to make the advertisement have more impact and to get more "clicks" from perspective candidates. Research has found that both employees and employers gravitate towards the website(s) with a large number of postings Breni (2014). However, when they use the website, they generally do not examine a larger number of postings. This could be a reason for employers to enrich the title with more detail as there are more competing job advertisements yet the prospective candidate wont review a broader sample of advertisements. See table 4.2 for examples of the titles scraped for financial accountant.

Financial Accountant, Insurance, Dublin West	Financial Accountant - Revenue team.
Biomedical sector	
Financial Accountant — Recently Qualified ACCA	
or Finalist Dublin City Centre	Financial Accountant — Technology
Financial Accountant — Regulatory —	
Insurance	Aircraft Leasing-Financial Accountant
Financial Accountant - Healthcare, Dublin	Financial Accountant - Pharmaceuticals
Dublin West - January 2017 - Urgent	
Financial Accountant,Multinational,	Financial Accountant in large
multinational in Wicklow	
Financial Accountant, Manufacturing,	
Dublin North	Financial Accountant - Construction Industry
Financial Accountant, Retail, Dublin North	Financial Accountant - Portuguese and Spanish

Table 4.2: Sample job titles for financial accountant

This phenomenon of job title enlargement poses challenges due to the uniqueness of this extra information. Of the 7,151 job titles scraped, the unique instances were 5,532 or 77%. To address this issue, lists of words to exclude from the job titles were compiled based on visual inspection of the list of titles.
Some titles added extra complexity; take the example of the following job titles:

"Manufacturing Engineer" Vs "Financial Accountant manufacturing"

By removing all instances of the word "manufacturing", the result is to standardise the financial accountant job title but materially alter the manufacturing engineer title to a more generic title of engineer. The general rule of thumb in this scenario is to keep/remove the word in favour of the more frequently occurring title, i.e. in this example there are considerably more instances of the words "financial accountant" than "manufacturing engineer" so the word manufacturing was removed.
Another consideration was word removal order. The below example looks at three variants related to Dublin as a location.

Original	**Remove "Dublin"**
Dublin	
Dublin City	City
Dublin City Centre	City Centre
Dublin County	County

Table 4.3: Impact of removing "Dublin"

The optimal approach in this example to avoid error is to remove the words in the order of;

Order	**Word**
1st	Dublin County
2nd	Dublin City Centre
3rd	Dublin City
4th	Dublin

Table 4.4: Optimal word removal order

By taking this approach of removing the larger variant first and then iterating to the next largest variant optimises the effectiveness of the cleaning of the job titles process.

Original	**1st removal**	**2nd removal**	**3rd removal**	**4th removal**
Dublin	Dublin	Dublin	Dublin	
Dublin City	Dublin City	Dublin City	""	
Dublin City Centre	Dublin City Centre	"	""	
Dublin County	"	"	"	

Table 4.5: Optimal word removal order

Word removal was accomplished using the *gsub* command (Global substitute) in *r*, with the word removed replaced with a "" (blank string).

Other more standard text mining pre-processing transformations were applied, including:

- Converting all characters to lowercase;
- Removal of punctuation; and
- Removal of special characters.

Lowercase

Some string operations such as *gsub* command are case sensitive and require a prompt to ignore case. By standardising all characters to lower case, it mitigates the risk of forgetting a prompt for a given operation. This transformation has no impact on the predictive models as we are not looking to identify specific grammar constructs like proper nouns and start of sentences which are capitalised. The goal is to determine the keywords that have significance in predicting a job title and the rules of grammar are therefore irrelevant.

Removal of punctuation

Similarly to the lower case scenario above, punctuation has no particular significance to this modelling exercise. Punctuation such as full stops, commas, etc is not particularly informative or relevant to job titles.

Removal of special characters

Likewise, special characters dont carry any particular significance in predicting a job title.

In total, 242 words for removal were identified. Coupled with the other transformations, after these were removed, the number of unique titles was reduced from 5,532 to 4,722 or 66% of all titles.

4.2.2 Body of text

For the main text of the job details, some of the same operations were performed as with the job titles, in addition to other treatments such as:

- Lower case
- Special characters
- Stopwords
- Whitespace
- Stemming

Lowercase

As discussed in section 3.3.1 the goal is to standardise all characters to avoid case sensitive error. Lower-case does not impact on modelling effectiveness.

Special Characters

Similarly to lower-case, special characters are of no benefit to the modelling and their removal reduces dimensionality of the main body of text. For example:

"project" vs "project,"

These two words will be assessed to be different words, however for the purposes of this modelling experiment the word "project" is the salient bit of information and the comma brings no value. By removing the punctuation the number of dimensions (unique words) has reduced by one which aids processing efficiency and the model will be enhanced by reducing sparsity.

Stopwords

As mentioned in section 3.3.3, words such as "the", "is, "that" are stopwords. Stopwords carry no significance for the purpose of this model, using the example of:

"the project" vs "project"

In this example the word "the" does not bring any additional meaning in determining the job title, the important word is "project". There is also the ancillary benefit that comes with dimension reduction as mentioned above with special characters.

Whitespace

As per section 3.3.4, the removal of special characters and punctuation best practice is to replace a character with a space rather than a zero string. For example:

Original text	**Replace with zero string**	**Replace with whitespace**
"Accounting/Finance"	"AccountingFinance"	"Accounting Finance"
"completed. Technology"	"completed Technology"	"completed Technology"

Table 4.6: Approaches to stripping Whitespace

In the first example of punctuation being replaced with a zero string, the result is this new concatenated word. It is of little value to the model and does not facilitate dimensionality reduction as this new unique word has been created. However, by providing a replacement with a single white space we get two distinct words. Both words in all likelihood will exist already so dimensionality has been reduced. The flip side of this is the second example where replacement with a zero string yields the favourable outcome of two distinct words which will in all likelihood reduce dimensionality. The replacement with whitespace leads to a double space between the two words. The implication is that the second word is interpreted as being " Technology" rather than "Technology".

The solution is to run the removal of whitespace after running the replace punctuation with whitespace. See below example:

Original text	**1st operation: replace with whitespace**	**2nd operation: strip out whitespace**
"accounting/Finance"	"accounting finance"	"accounting finance"
"completed. technology"	"completed technology"	"completed technology"

Table 4.7: Optimal sequence to stripping whitespace

Stemming

As previously mentioned in section 3.3.5, stemming is the process of reduction of words by truncating the different forms of the same word. A reminder is the example organise, organises, organising.

4.2.3 Data partitioning

There were 4,722 unique job titles after the job titles process. The dataset was restricted to the top 30 frequently occurring titles. This was in part for the *Random Forest* package did not support greater than 32 predictors. And the number of instances at n= 30 is 18 and increasing n $>$ 30 yields smaller number of instances. See figure 4.2 for a bar chart illustrating the job title frequency.

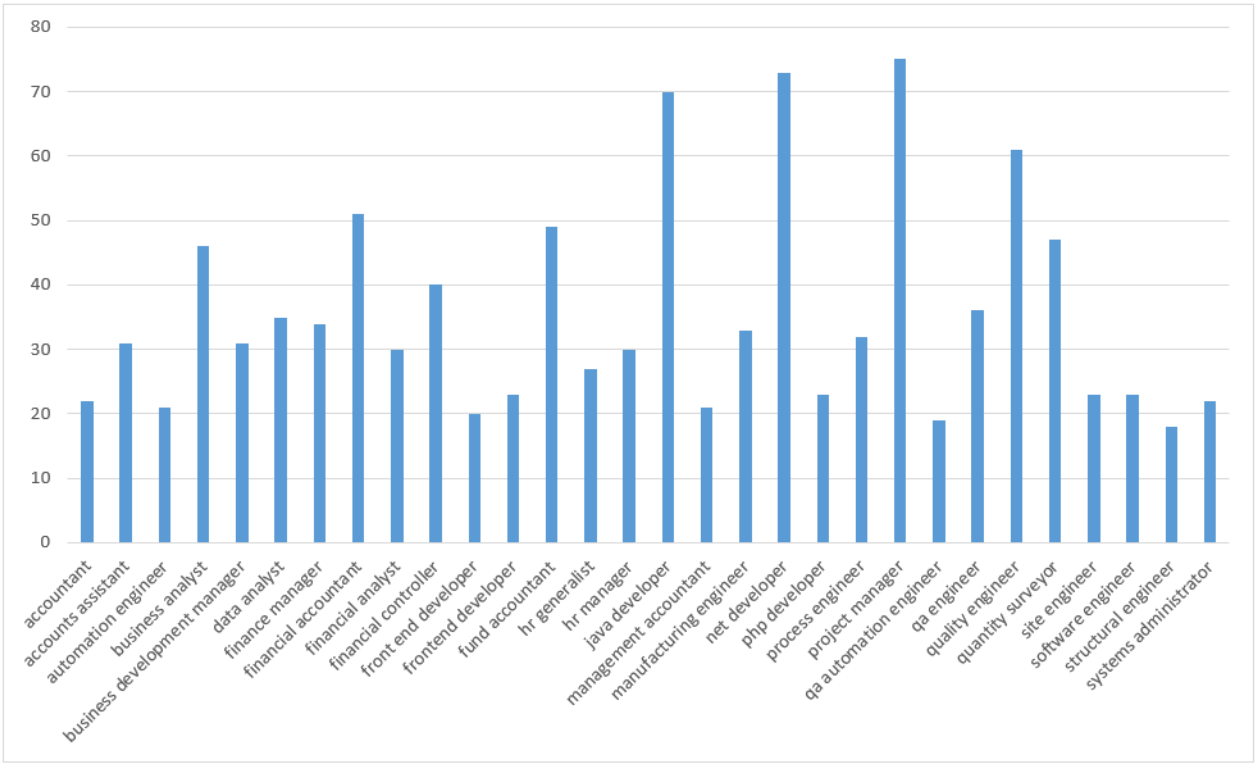

Figure 4.2: Frequency of Top 30 Job Titles

The dataset is converted into a test and train holdout sets using the *createDataPartition* command within the *caret* package in *r*. The split is set to 70/30 and the splitting is stratified for each label. This automatically occurs as a custom field is created called LabelNo. LabelNo is a numeric representation of the job title generated by *r*. In addition, this field is defined as a factor. A factor is a special data type within *r* that recognises unique values in a list and stores them in the background on that basis. The command looks as follows:

```
splitter <- createDataPartition(alldata$LabelNo,
p=.70,list = FALSE, times=1)
```

Table 4.8 illustrates a sample of the stratified partitioning of the data at or close to the 70/30 ratio.

Label	Test	Train	Total	% of total
accountant	6	16	22	73%
accounts assistant	9	22	31	71%
automation engineer	6	15	21	71%
business analyst	13	33	46	72%
business development manager	9	22	31	71%
data analyst	10	25	35	71%
finance manager	10	24	34	71%

Table 4.8: Example of stratified holdout data splitting

4.3 Feature extraction

In section 3.3.7 the features are extracted on the basis of term frequency. The *tm* package has a function to identify the most frequent terms. However this function runs on the entirety of a corpus. In order to create the analytics base table, the frequent terms should be identified for each label in isolation and not the entirety of the data set.

The first step is to stratify the dataset label by label. The following script illustrates this using the subset command to create a separate dataset for the instances of label *41* and another for *68*.

```
Alldata_41 <- subset(Alldata_train, LabelNo=="41")
Alldata_68 <- subset(Alldata_train, LabelNo=="68")
```

The *findMostFreqTerms* command in *tm* is applied to each dataset and the no of words was set to extract the top 50 most frequent terms.

```
FreqWords_41_Cleaned_Details <- findMostFreqTerms(
Alldata_41_Cleaned_Details_tdm,
noofwords,INDEX=rep(1:1,each=eval41))
```

In table 4.9 an example of some of the results with the frequent terms and no of instances identified for label 41.

word	**no of instances**
accounting	49
work	43
team	35
accountant	34
will	33
financial	32
accounts	30
client	29

Table 4.9: Label 41 sample of frequent terms

Additional transformations are conducted to this list, firstly the number of instances is discarded as the term itself is what's required for modelling not the frequency. The list is transposed in order to adhere to the ABT style table as discussed in section 3.3.7.

Using the previous example in 4.9 the ABT entry for label 41 would appear as follows:

Target	**Term1**	**Term2**	**Term3**	**Term4**	**Term5**	**Term6**
41	accounting	work	team	accountant	will	financial

Table 4.10: Analytics base table

Once the full ABT was created, it was observed that several words appear in every instance:

"experience, company, role, will, work"

These words were excluded as they did not bring any influence to the modelling process by being common to all and therefore not differentiating.

Besides not being informative, a common term to all instances would undermine the support vector machine algorithm. SVM operates on the basis of identifying a separating hyperplane to categorise items. In the scenario with a common dimension across all items, this separating plane cannot be derived.

4.4 Modelling

4.4.1 Random Forest

The first model generated was a Random Forest using the *randomForest* package in *r*. Using a baseline algorithm with no parameter tuning, the results are as follows:

Evaluation Metric	**Results**
Accuracy	56.21%
FScore_{M}	49.25%
$\text{Precision}_{\text{M}}$	54.56%
Recall_{M}	49.03%

Table 4.11: Results random forest algorithm

In section 3.5 the basis of evaluation was examined. The definition focuses on accuracy, $\text{precision}_{\text{M}}$, recall_{M} and Fscore_{M}.

Accuracy

$$\frac{\sum_{i=1}^{l} \frac{tp_i+tn_i}{tp_i+fn_i+fp_i+tn_i}}{l} \tag{4.1}$$

Precision$_{\mathbf{M}}$

$$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fp_i}}{l} \tag{4.2}$$

Recall$_{\mathbf{M}}$

$$\frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i+fn_i}}{l} \tag{4.3}$$

Fscore$_{\mathbf{M}}$

$$\frac{(\beta^2+1)Precision_M Recall_M}{\beta^2 Precision_M + Recall_M)} \tag{4.4}$$

4.4.2 Random Forest Tuning

The package *randomForest* has a tuning function called *tuneRF*. This function begins with a baseline default setting random forest parameters and iterates through various *mtry* values to find the optimal value. The *mtry* is the number of random samples used for bootstrapping. The outcome was the baseline model as per section 4.4.1 was the optimal model.

4.4.3 Support Vector Machines

Meyer et al. (2017) used the e1071 in r to produce models using SVM algorithm for the results of three models using the following parameters: SVM with a linear kernel, cost value = 1 SVM with Polynomial kernel, degree value =1, cost =1 SVM with Radial kernel, cost =1. The results are shown in table 4.12

Evaluation Metric	**SVM Linear Kernel**	**SVM polynominal Kernel**	**SVM radial Kernel**
Accuracy	63.07%	63.07%	63.07%
F-Measure$_{\mathrm{M}}$	62.1%	62.1%	62.1%
Precision$_{\mathrm{M}}$	62.66%	62.66%	62.66%
Recall$_{\mathrm{M}}$	61.21%	61.21%	61.21%

Table 4.12: Results SVM algorithms

After running numerous variants on the SVM algorithms with alternative cost and degree values, a parameter was incremented in steps of 1. The linear SVM maintained its performance whereas polynominal and radial either matched or had diminished preference with parameter tuning. The finding that linear was joint best or outright best performance is in line with the literature reviewed above. At this stage the polynomial and radial models were dropped from further analysis.

Looking at the model without stemming we get results as shown in table4.13. There is a negligible change in the Random Forest accuracy and recall measures. The F measure has improved whilst the precision reduced.

A SVM linear kernel model on the non-stemmed data did yield improvement across all metrics.

Evaluation Metric	**Random Forest**	**SVM linear**
Accuracy	57.84%	68.3%
F-Measure$_{\mathrm{M}}$	56.33%	65.41%
Precision$_{\mathrm{M}}$	55.83%	67.58%
Recall$_{\mathrm{M}}$	55.29%	65.99%

Table 4.13: Results Random Forest and SVM algorithms with no word stemming

Looking at the job title "software engineer", the stemmed corpus yielded three predictions of "automation engineer", one prediction of "java developer" and two of "php developer". In the non-stemmed corpus, the were no predictions of "automation engineer", the same number of predictions for "java developer" &php developer and it correctly predicted three instances of "software engineer", as summarised in Table 4.14.

	Prediction Software Engineer	
Actual Labels	**SVM Stemmed**	**SVM Non Stemmed**
automation engineer	3	0
java developer	1	1
php developer	2	2
software engineer	0	3

Table 4.14: Predictions of label software engineer stemmed and non stemmed corpus

Upon close examination, the stemmed features for software engineer are not common with the stemmed features of automation engineer. None of these words stand out as being particularly indicative and unique to the domain of software engineering as outlined in Table 4.15.

Word	**No of instances**	**Word**	**No of instances**
differ	1	qualiti	1
environ	1	skill	1
join	1	strong	1
medic	1	technolog	1
one	1	world	1
part	1	year	1

Table 4.15: Features of software engineer not in common with automation engineer, stemmed dataset

Looking at the non-stemmed features in Table 4.16 for software engineer not in common with automation engineer again the features do not appear to be particularly indicative of a software engineer. The inference is that the improved predictive performance is not linked to the impact of stemming/not stemming on the words unique to software engineer.

Word	**No of instances**	**Word**	**No of instances**
difference	1	quality	1
environment	1	skills	1
join	1	strong	1
medical	1	technology	1
one	1	working	1
part	1	world	1
position	1	years	1

Table 4.16: Features of software engineer not in common with automation engineer, non stemmed dataset

Upon examination of shared terms in the stemmed and non-stemmed corpus, one of the distinguishing differences is that related to the words develop/development. In the non-stemmed model, there is a single instance of the word "develop" and "development" for automation engineer, with a single instance of the word "development" for software engineer. In the stemmed version, the word development gets truncated to develop and as a result the stemmed model has "develop" with a value of 2.

The conclusion that can be inferred is that the existence of both the words "develop" and "development" in the non-stemmed model are not enough to influence the predictive model. However, the increased magnitude of the word "develop" in the stemmed model does have more influence and therefore gave rise to the three incorrect predictions of "automation engineer".

Non Stemmed Corpus		
Word	automation engineer	Software engineer
develop	1	0
development	1	1

Stemmed Corpus		
Word	automation engineer	Software engineer
develop	2	1

Table 4.17: Example of difference between stemming and non stemming

Taking a more granular look at each label, the diminished performance on certain labels is observed. This is confirmed by the precision value as per table 4.18.

The macro average for precision, ($\text{Precision}_{\text{M}}$) is calculated as 68%, and the scoring is consistently less than this value on the accounting and finance type roles as per Table 4.18 below.

An explanation for this is the difficultly in de-delineating between the various accounting type jobs, which will have significant overlap in terms of keyterms. The label "fund accountant" belies this trend and has a perfect 1.0 sensitivity score. The reason underpinning the 1.0 score is that this label has unique words such as "hedge", "nav", "equity" which are terms particularly relevant to the work of fund accounting and less applicable to other forms of accounting.

Label	**Precision**	**Label**	**Precision**
accountant	20.00%	java developer	80.95%
accounts assistant	66.67%	management accountant	75%
automation engineer	80.00%	manufacturing engineer	45.45%
business analyst	58.82%	net developer	100%
business development manager	100.00%	php developer	50%
data analyst	88.89%	process engineer	50%
finance manager	30.00%	project manager	76.47%
financial accountant	35.29%	qa automation engineer	33.33%
financial analyst	37.50%	qa engineer	57.14%
financial controller	69.23%	quality engineer	81.25%
front end developer	40.00%	quantity surveyor	100%
frontend developer	66.67%	site engineer	46.15%
fund accountant	100.00%	software engineer	75%
hr generalist	85.71%	structural engineer	100%
hr manager	77.78%	systems administrator	100%

Table 4.18: Precision value per label SVM non stemmed model

4.5 Manual Model

Upon examination of the predictions as per Fig 4.3, it is observed that for "site engineer" there are six instances predicted as "project manager". There is overlap in terms such as "management", "project","projects", but not enough features to delineate the two labels in order to produce a boundary hyperplane sufficient to make predictions greater than a precision value of 46.15%. By examining the features on both labels in the ABT, certain terms are noted such as "new", "now", which appeared as frequent terms for "site engineer". These terms are likely more descriptive of the advertisement rather than the role itself; for example "new role, need to hire now". This example is more indicative of the tight labour market around the construction sector rather than

informative of the role itself. By removing some of these terms the outcome was an improvement in correct predictions.

Another strategy is the additive where information was added with reference to frequent terms not included in the original ABT that would be domain specific. In an early version of this model, the words "opportunity" and "opportunities" were removed as they appeared to be quite common. Again, it was deemed to be a term about the advertisement rather than the role; for example "an exciting new opportunity ...". However "opportunities" is an informative term for the role of business development manager; i.e. to "identify new opportunities for growth". Therefore, this word was added (or added back after it had been removed).

Reference \ Prediction	accountant	accounts assistant	automation engineer	business analyst	business development manager	data analyst	finance manager	financial accountant	financial analyst	financial controller	front end developer	frontend developer	fund accountant	hr generalist	hr manager	java developer	management accountant	manufacturing engineer	net developer	php developer	process engineer	project manager	qa automation engineer	qa engineer	quality engineer	quantity surveyor	site engineer	software engineer	structural engineer	systems administrator
accountant	1	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
accounts assistant	0	8	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
automation engineer	0	0	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
business analyst	0	0	0	10	1	2	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1
business development manager	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
data analyst	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
finance manager	0	0	0	0	0	0	3	2	2	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
financial accountant	4	0	0	1	0	0	1	6	3	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
financial analyst	0	0	0	0	1	0	2	1	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
financial controller	1	1	0	0	0	0	1	1	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
front end developer	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
frontend developer	0	0	0	0	0	0	0	0	0	0	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fund accountant	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
hr generalist	0	0	0	0	0	0	0	0	0	0	0	0	0	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
hr manager	0	0	0	0	0	0	0	0	0	0	0	0	0	1	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
java developer	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	17	0	0	0	1	0	0	0	1	0	0	0	1	0	0
management accountant	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
manufacturing engineer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	4	0	0	0	2	0	0	0	0	0
net developer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0
php developer	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	5	0	0	0	0	0	0	0	2	0	0
process engineer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	5	1	0	0	2	0	0	0	0	0
project manager	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0
qa automation engineer	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	5	0	0	0	0	0	0
qa engineer	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	4	0	0	0	0	0	0
quality engineer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	13	0	0	0	0	0
quantity surveyor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0
site engineer	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	6	0	0	0
software engineer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	3	0	0
structural engineer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
systems administrator	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4

Figure 4.3: Predictions matrix SVM non stemmed model

Subsequent to continuing this strategy of adding and removing words and selective reference or prediction pairs, a new model was produced which presented results as per Table 4.19 below.

Evaluation Metric	**Random Forest**	**SVM linear**
Accuracy	56.86%	70.59%
F-Measure$_{\mathrm{M}}$	54.20%	65.89%
Precision$_{\mathrm{M}}$	55.19%	70.02%
Recall$_{\mathrm{M}}$	54.45%	67.02%

Table 4.19: Results Random Forest and SVM algorithms with manual model

Chapter 5

Evaluation and analysis

5.1 Introduction

This chapter evaluates the predictive strength of the model developed. The evaluation is based on the metrics discussed in section 3.5.

5.2 Evaluation

As the results in Table 4.11 demonstrate, the accuracy of the Random Forest is **56.21%**. This was after application of text pre-processing treatments of stopword removal and stemming. Specifically, the Fscore_{M} is **49.25%**, $\text{precision}_{\text{M}}$ is **54.56%** and recall_{M} is at **49.03%**.

Using the same dataset but without the application of stemming, the Random Forest algorithm has an accuracy value of **57.84%**. The Fscore_{M} is **56.33%**, the $\text{precision}_{\text{M}}$ is **55.83%** and the recall_{M} is **55.29%** as summarised in Table 4.13 .

The results upon application of the Support Vector Machine algorithm on the stemmed dataset has an accuracy rate of **63.07%** as per table 4.12. The Fscore_{M} is **62.1%**, the $\text{precision}_{\text{M}}$ is **62.66%** and the recall_{M} **61.21%**. These scores were replicated consistently across 3 different kernel types (linear, polynominal, radial).

Applying SVM on the non-stemmed dataset, the accuracy score of **68.3%** is noted. The Fscore_{M} is **65.41%**, the $\text{precision}_{\text{M}}$ is **67.58%** and the recall_{M} is **65.99%** as outlined in table 4.13 .

The overall performance is summarised in Table 5.1 below and the bar chart in Figure 5.1.

Model	Accuracy	$\text{Precision}_{\text{M}}$	Recall_{M}	Fscore_{M}
SVM linear stemmed	63.07%	62.66%	61.21%	62.10%
SVM linear non stemmed	68.3%	67.58%	65.99%	65.41%
Random Forest stemmed	56.21%	54.56%	49.03%	49.25%
Random Forest non stemmed	57.84%	55.83%	55.29%	56.33%
Random Forest manual	56.86%	55.19%	54.45%	54.20%
SVM linear manual	70.59%	70.02%	67.02%	65.89%

Table 5.1: Overall model summary

It is observed for table 5.1 and also figure 5.1 that SVM typically outperforms RF. In addition, the non stemmed corpus out performs the stemmed corpus as discussed in section 4.4.3, the reason was in the compression of the keywords post stemming. The weighting of the key terms were altered from 0/1 to 0/1/2. This had an impact on the SVM algorithm and the predictions made.

The performance of the SVM manual model was achieved from selectively looking at per class predictions and target terms that would facilitate the formation of the hyperplane boundary.

In a study of multi-class text classification of a variety of corpora, the classification errors range from 8% to 29% across a range of models (Nigam et al., 1999).
The best performing model in this study was the SVM linear manual model which achieved a result of 29%.

Classification Error

$$\frac{\sum_{i=1}^{l} \frac{fp_i+fn_i}{tp_i+fn_i+fp_i+tn_i}}{l} \tag{5.1}$$

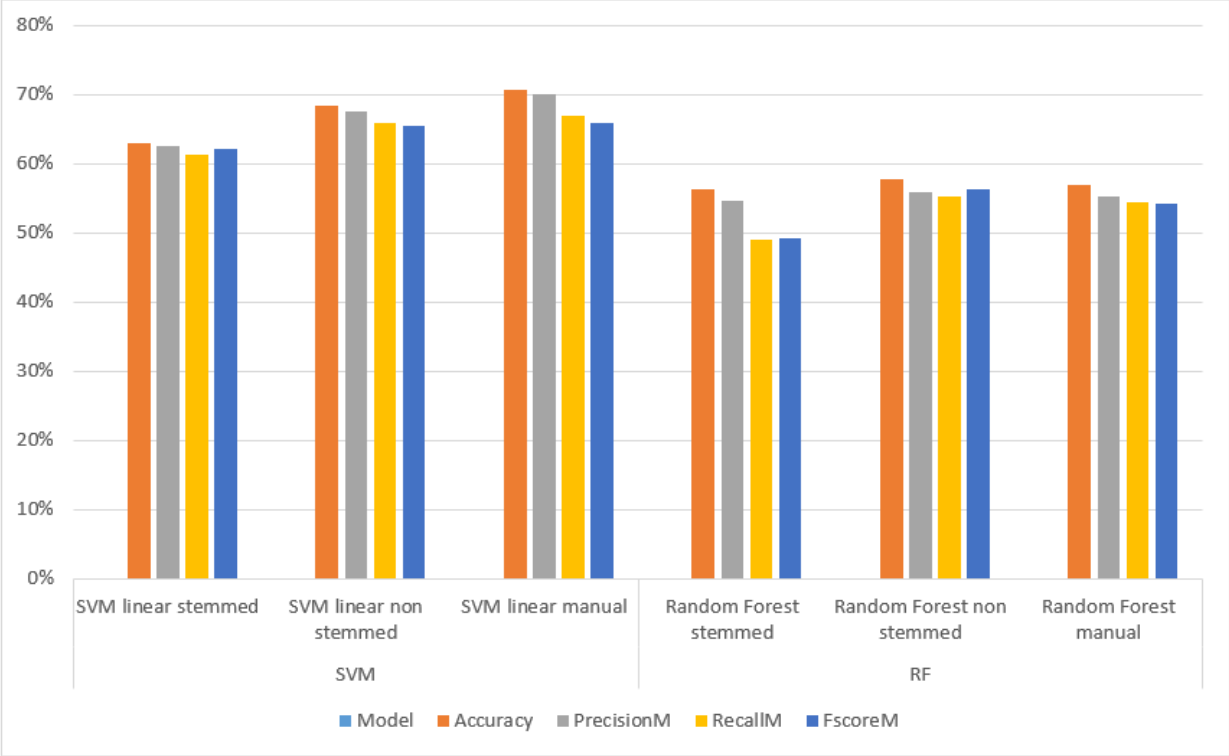

Figure 5.1: Model summary bar chart

This demonstrates that a model can be produced with an accuracy rate in the order of 71%, illustrating the extent of the linkages between job titles and job descriptions.

Chapter 6

Conclusion

This chapter summarises the findings of the research as it related to the problem identified, and the results of the design and implementation of the experiment.

6.1 Research Overview

The goal of this research study was to examine the extent that a job title can be predicted by the job description using supervised machine learning. The objectives as discussed in section 1.3 were:

- To investigate the extent to which predictions can be made of a job title using job details.
- To extract features and build a model for use with supervised machine learning algorithms to make predictions
- To refine the model by means of parameter tuning and feature selection.

6.2 Problem Definition

As outlined at the outset, the process of job classification is typically subjective when conducted by the responsible manager or Human Resources staff. There is potentially a considerable amount of information contained in the job description which somehow

a job title must encapsulate. By using a manual and non-objective approach, some of this information may not be taken into consideration during the advertisement and recruitment process and it may even be misunderstood. There are risks that personal bias and inconsistencies may arise and impact how the job is defined.

A job title has a bearing on where an individual sits within the organisational structure, and this has numerous implications for matters including salary scale, levels of responsibility, motivation and so on.

The challenge and opportunity, as presented in this paper, is in using machine learning to bring a systematic, algorithmic-based approach rooted in a corpus of real job data to address this classification problem.

6.3 Design/Experimentation, Evaluation & Results

As covered in this study, data was collected via web scraping from the web site "www.irishjobs.ie". Standard text pre-processing transformations were made to the data to strip out features like punctuation, whitespace, special characters and apply word stemming in order to reduce dimensionality.

Upon exploration of the data it was observed the job titles required some data cleaning. Many job titles contained text not applicable to the job itself only to make the job advertisement more attractive/impart as much as possible within the job title, eg "immediate start", "in Dublin city centre".

A model was generated using term frequency to identify keywords that were potentially informative to the job title. Predictions were made using the SVM and Random forest algorithms on the model.

Alternative predictions were made by means of parameter tuning for the algorithms. For the SVM algorithm the parameters altered were kernel types (Linear, Radial, Polynominal) and their related parameters (Cost, Degree). For Random Forest, a function to generate an optimal model by changing the sample sizes (*mtry*) also did not yield a better performing model than the baseline default version.

Upon review, additional models with produced without the application of word

stemming as this was found to affect the performance.

The best performing model was SVM linear model, with features selectively taken from a list of the most frequently used terms and targeted specify per class predictions that had poor performance due to

6.4 Contributions and impact

A review of the literature conducted in this work did not prove fruitful in finding similar studies of job title prediction from job descriptions. This study is therefore contributing to the research on application of supervised learning techniques related to the area of job titles and specifically their prediction.

A number of related studies were reviewed which explored the application of machine learning algorithms to other online data (such as spam emails and analysis of movie reviews). This research directed the author to use of specific type of algorithms which would be most useful in this context; for example SVM and Random Forest.

The impact of my conclusion that SVM is more effective that Random Forest could provide a useful guideline for practitioners in ensuring more accurate classification online.

A data model with enough predictive strength was obtained in this study (71% Accuracy rate on a SVM based model) and a methodology applied can serve as a guide for automated predictions.

This study has demonstrated the link between a job title and the job details (although difficult to assess at first glance) can be sized through machine learning.

A search of the academic repositories available did not yield any directly comparable research. The originality of the body of work lays in the application of well established machine learning and text mining processes to a new domain, specifically classification of job titles from job description textual data.

6.5 Future Work & recommendations

More advanced text processing methods could be established using NLP dictionaries, while more complex machine learning techniques can be employed such as ensemble methods to improve predictive power.

Other similar and parallel research on web based text mining, linking fields of activity to their intrinsic components may use the described method. This could be applied to domains such as marketing and medical categorisation.

References

Bermingham, A., & Smeaton, A. F. (2010). Classifying Sentiment in Microblogs: Is Brevity an Advantage? In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (pp. 1833–1836). New York, NY, USA: ACM. Retrieved 2017-02-15, from http://doi.acm.org/10.1145/1871437.1871741 doi: 10.1145/1871437.1871741

Bouchet-Valat, M. (2014, August). *SnowballC: Snowball stemmers based on the C libstemmer UTF-8 library.* Retrieved 2017-06-21, from https://cran.r-project.org/web/packages/SnowballC/index.html

Breiman, L. (2001, October). Random forests. *MACHINE LEARNING*, *45*(1), 5–32.

Breni, V. (2014, June). Search online: Evidence from acquisition of information on online job boards and resume banks. *Journal of Economic Psychology*, *42*, 112–125. Retrieved 2016-09-24, from http://www.sciencedirect.com/science/article/pii/S0167487014000051 doi: 10.1016/j.joep.2014.02.003

Cable, D., Grant, A., & Berg, J. (2013). What's in a Job Title? *Business Strategy Review*, *24*(4), 74–74. Retrieved 2016-09-29, from http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=92727356&site=ehost-live doi: 10.1111/j.1467-8616.2013.01007.x

Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. *Ann Arbor MI*, *48113*(2), 161–175.

Cortes, C., & Vapnik, V. (1995). Support Vector Networks. *MACHINE LEARNING*, *20*(3), 273–297. Retrieved from https://link.springer.com/article/10.1023/A:1022627411411

Giancola, F. (2014, May). Inflated Job Titles: When and How HR Bends the Rules. *Employee Benefit Plan Review*, *68*(11), 30–32. Retrieved 2016-09-29, from http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=95847456&site=ehost-live

Grant, A. M., Berg, J. M., & Cable, D. M. (2014, August). Job Titles as Identity Badges: How Self-Reflective Titles Can Reduce Emotional Exhaustion. *Academy of Management Journal*, *57*(4), 1201–1225. Retrieved 2016-10-01, from http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=97342265&site=ehost-live doi: 10.5465/amj.2012.0338

Guo, S., Alamudun, F., & Hammond, T. (2016, October). RsuMatcher A personalized rsum-job matching system. *Expert Systems with Applications*, *60*, 169–182. Retrieved 2017-02-14, from http://www.sciencedirect.com/science/article/pii/S0957417416301798 doi: 10.1016/j.eswa.2016.04.013

Koprinska, I., Poon, J., Clark, J., & Chan, J. (2007, May). Learning to classify e-mail. *Information Sciences*, *177*(10), 2167–2187. Retrieved 2017-06-14, from http://www.sciencedirect.com/science/article/pii/S0020025506003707 doi: 10.1016/j.ins.2006.12.005

Kosala, R., & Blockeel, H. (2000, June). Web Mining Research: A Survey. *SIGKDD Explor. Newsl.*, *2*(1), 1–15. Retrieved 2017-06-24, from http://doi.acm.org/10.1145/360402.360406 doi: 10.1145/360402.360406

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., C++-code), C.-C. C. l., & C++-code), C.-C. L. l. (2017, February). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien.* Retrieved 2017-06-20, from https://cran.r-project.org/web/packages/e1071/index.html

M.F. Porter. (2006, July). An algorithm for suffix stripping. *Program*, *40*(3), 211–218. Retrieved 2017-06-12, from http://0-www.emeraldinsight.com.ditlib.dit.ie/doi/full/10.1108/00330330610681286 doi: 10.1108/00330330610681286

Nigam, K., Lafferty, J., & McCallum, A. (1999). Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering* (Vol. 1, pp. 61–67).

Noh, H., Jo, Y., & Lee, S. (n.d.). *Keyword selection and processing strategy for applying text mining to patent analysis - ScienceDirect.* Retrieved 2017-06-20, from http://0-www.sciencedirect.com.ditlib.dit.ie/science/article/pii/S0957417415000652?via%3Dihub

Onan, A., Korukolu, S., & Bulut, H. (2016, September). Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, *57*, 232–247. Retrieved 2017-06-14, from http://www.sciencedirect.com/science/article/pii/S0957417416301464 doi: 10.1016/j.eswa.2016.03.045

Sokolova, M., & Lapalme, G. (2009, July). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. Retrieved 2017-06-12, from http://www.sciencedirect.com/science/article/pii/S0306457309000259 doi: 10.1016/j.ipm.2009.03.002

Sompras, G., & Lalitrojwong, P. (2010, August). Extracting product features and opinions from product reviews using dependency analysis. In (pp. 2358–2362). IEEE. Retrieved 2017-06-20, from http://ieeexplore.ieee.org/document/5569865/ doi: 10.1109/FSKD.2010.5569865

Tripathy, A., Agrawal, A., & Rath, S. K. (2016, September). Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications*, *57*, 117–126. Retrieved 2016-10-08, from http://www.sciencedirect.com/science/article/pii/S095741741630118X doi: 10.1016/j.eswa.2016.03.028

Wang, S., & Manning, C. D. (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2* (pp. 90–94). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved 2017-06-20, from `http://dl.acm.org/citation.cfm?id=2390665.2390688`

Wickham, H., & RStudio. (2016, June). *rvest: Easily Harvest (Scrape) Web Pages.* Retrieved 2017-06-20, from `https://cran.r-project.org/web/packages/rvest/index.html`

Xiu-li, P., Yu-qiang, F., & Wei, J. (2007, August). A Chinese Anti-Spam Filter Approach Based on Support Vector Machine. In *2007 International Conference on Management Science and Engineering* (pp. 97–102). doi: 10.1109/ICMSE.2007.4421831

Appendix A

Additional content

A.1 Stop words

a	both	hadn't	if	of	so	through	which
about	but	has	i'll	off	some	to	while
above	by	hasn't	i'm	on	such	too	who
after	cannot	have	in	once	than	under	whom
again	can't	haven't	into	only	that	until	who's
against	could	having	is	or	that's	up	why
all	couldn't	he	isn't	other	the	very	why's
am	did	he'd	it	ought	their	was	with
an	didn't	he'll	its	our	theirs	wasn't	won't
and	do	her	it's	ours	them	we	would
any	does	here	itself	ourselves	themselves	we'd	wouldn't
are	doesn't	here's	i've	out	then	we'll	you
aren't	doing	hers	let's	over	there	were	you'd
as	don't	herself	me	own	there's	we're	you'll
at	down	he's	more	same	these	weren't	your
be	during	him	most	shan't	they	we've	you're
because	each	himself	mustn't	she	they'd	what	yours
been	few	his	my	she'd	they'll	what's	yourself
before	for	how	myself	she'll	they're	when	yourselves
being	from	how's	no	she's	they've	when's	you've
below	further	i	nor	should	this	where	
between	had	i'd	not	shouldn't	those	where's	

Figure A.1: List of stop words