Automatically Coding Occupation Titles to a Standard Occupation Classification

by

Negin Nahoomi

A Thesis

presented to

The University of Guelph

In partial fulfilment of requirements
for the degree of

Master of Science

in

Computer Science

Guelph, Ontario, Canada

ABSTRACT

Automatically Coding Occupation Titles to a Standard Occupation Classification

Negin Nahoomi
University of Guelph, 2018

Advisors:
Dr. Fei Song
Dr. Gary Gréwal

Occupation Coding is the process of classifying job titles into one or multiple categories that are usually organized into a hierarchy. Historically, the task of classifying job titles to standard classifications was done manually. However, the drawbacks of manual coding have led researchers to develop automatic methods for occupation coding. We compare the classic machine learning approaches and the deep learning approaches on classifying job titles to Standard Occupational Classification (SOC). We implement flat and hierarchical models using Naïve Bayes, Maximum Entropy (MaxEnt), Support Vector Machines (SVM), and Convolutional Neural Networks (CNN) to code job titles to SOC. For this purpose, 65,962 SOC labeled job titles are collected from publicly available sources. These job titles are extremely short with an average of three words per job title. Our experimental results show that MaxEnt, SVM, and CNN perform similarly and are better than Naïve Bayes on coding job titles to SOC.

Acknowledgements

I would first like to express my sincere gratitude to my advisors, Dr. Fei Song and Dr. Gary Gréwal for their support, patience, and understanding. Most importantly, thank you for teaching me critical thinking and problem solving skills. Your guidance helped me throughout my study, research, and thesis writing.

I would also like to acknowledge Dr. Miana Plesca and Dr. Luiza Antonie for providing insightful knowledge throughout my research, and Dr. Andrew Hamilton-Wright for his valuable comments on my thesis.

I am forever grateful to my parents for supporting me throughout this journey in the hardest of times. To my sisters; Maral and Anahita, thank you for always supporting and encouraging me. I would also like to thank my extended family for helping me to adjust to a new life in Canada. Mahshid and Kevin, thank you for editing my writing. Pallak, thank you for taking care of me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Occupation Coding is the process of classifying occupation titles into one or multiple occupation classes that are usually organized into a hierarchical structure [63]. The occupation classifications typically categorize occupations based on skills and education. These classifications can also be designed to sort occupations internationally, historically, and nationally.

The previous work on occupation coding can be divided into manual, computer-assisted, and automatic methods. Manual coding of job titles to standard classifications is time-consuming and cost intensive. For example, it can take four to eight years for one trained coder to classify a database of 200,000 occupations [7]. As another example, the United States Census Bureau spent approximately seven million dollars categorizing 17 million occupation titles manually in 1980 [10]. In spite of the disadvantages, most organizations still depend on manual coding of occupations to an extent. Thus, there is a need to develop a method to automatically code job titles to standard classifications with high quality.

In computer-assisted approaches, computer programs are designed to help professional coders assign codes faster and with higher confidence. The research on automatic techniques for coding occupation titles can be further divided into three categories: rule-based, machine learning, and hybrid methods. In the rule-based approach, a set of rules are used for classifying occupation titles. While rule-based classification is more efficient than manual coding, huge amounts of data and handcrafted rules are needed to perform reliable classification. More recently, the occupation coding research has shifted towards machine learning algorithms due to the

success of machine learning in text classification. Therefore, in this thesis, we provide automated methods, based on supervised machine learning algorithms, to code job titles to a standard occupation classification.

1.1 Motivation

Occupation coding has applications in different fields such as statistical, social, and epidemiology studies. For instance, occupation coding is used for identifying diseases and injuries related to different occupation classes, inferring social classes, analyzing labour statistics and wage regressions [35, 63]. Furthermore, at the University of Guelph, social scientists are analyzing the gender wage gap in the Ontario's public sector salary disclosure. The Ontario public sector salary disclosure, also known as the *Sunshine List*, is a yearly report that is published by the government of Ontario on individuals working in Ontario's public sector who have an annual salary of higher than 100,000 CAD [44]. The information available in the Sunshine List includes *first name, last name, salary, employer, job title* and *taxable benefits*. A previous study has inferred the gender of individuals listed in the Sunshine List based on their first name [4]. Further study towards analyzing the gender pay gap in the Sunshine List could be done by comparing the occupation classes of the occupation titles. The models proposed in this work, can be used for coding the Sunshine List's occupation titles to Standard Occupational Classification (SOC) [73], which is a well-known occupation classification. We justify classifying occupation titles to SOC instead of National Occupational Classification (NOC) [17] based on two main reasons: (1) the amount of publicly available SOC labeled data that is available online. (2) compatibility with Occupational Information Network (ONET) [1], which is a well-known online database that links occupation codes to skills, abilities, knowledge, etc.

1.2 Thesis Statement

In this thesis, we present multiple supervised machine learning models for coding occupation titles to SOC. We explore the effectiveness of currently commonly used text classification techniques on coding occupation titles to SOC.

1.3 Approach

In this thesis, we compare classic supervised machine learning techniques and supervised deep learning techniques on classifying occupation titles. We implement flat models with Naïve Bayes [20], Maximum Entropy (MaxEnt) [13], Support Vector Machines (SVM) [12], and Convolutional Neural Networks (CNN) [36] as well as hierarchical models with SVM and MaxEnt. We use pre-processing techniques such as lowercasing, removing punctuation, numbers, and stop words. We also implement a set of experiments that show stemming and feature selection are not effective due to attributes of data. We use micro F1, macro F1, precision, recall, and hamming loss to evaluate the performance of the models.

To train the supervised classifiers, we collected 65,962 SOC labeled occupation titles from publicly available sources. Some of the job titles belong to more than one class. If the job title is detailed enough, a single class can be assigned to it. However, occupation coding is often done on *free-text* data. Free-text data refers to data that is not created based on a predefined structure and contains irregularities. Therefore, the titles are not always specific enough to assign a single label. Section 3 discusses the reasons for the multiple labels in detail. As a result, we use a multi-label classification scheme.

1.4 Contributions

The main contributions of this work are:

1. A multi-label flat classification model for coding occupation titles to SOC;
2. A multi-label hierarchical classification model for coding occupation titles to SOC; and,
3. A comparison of SVM, Maximum Entropy, Naïve Bayes, and Convolutional Neural Network on classifying short multi-label occupation titles.

1.5 Organization

The remainder of this thesis is organized as follows. Chapter 2 introduces the necessary background information. Chapter 3 provides detailed analysis of the SOC labeled data. Chapter 4 presents the proposed flat and hierarchical occupation coding models. Chapter 5 presents the deep learning model for coding occupation titles. Chapter 6 concludes this study along with possible directions for future research.

Chapter 2

Background on Automatic Occupation Coding

Occupation coding classifies job titles into one or multiple occupation classes that are typically organized into a hierarchical scheme. In this chapter, we introduce the necessary background information for understanding automatic occupation coding using supervised machine learning algorithms. Section 2.1 describes two well-known occupation classification systems used in North America. Section 2.2 provides information on automatic text classification. Section 2.3 reviews the machine learning classifiers used in this thesis. Finally, Section 2.4 reviews the previous studies on occupation coding.

2.1 Occupation Classification Systems

Occupation classification systems categorize occupation titles into multiple occupation classes that are usually organized into a hierarchical structure [63]. Occupation classification systems can be designed to sort occupations internationally, historically, or nationally based on skills, education. We look into two well-known occupation classifications: *National Occupational Classification (NOC)* [17] and *Standard Occupational Classification (SOC)* [73].

NOC categorizes jobs in the Canadian labor market based on skill and education levels [17]. NOC-2011 has 690 classes organized in a four level hierarchy: there are 10 *broad occupational categories* (first level), 40 *major groups* (second level), 140 *minor groups* (third level), and 500 *unit groups* (fourth level). Table 2.1 shows the 10 classes in the first level of the NOC hierarchy.

Table 2.1: NOC-2011 First Level Categories

NOC Code	Occupation Title
0	Management occupations
1	Business, finance and administration occupations
2	Natural and applied sciences and related occupations
3	Health occupations
4	Occupations in education, law and social, community and government services
5	Occupations in art, culture, recreation and sport
6	Sales and service occupations
7	Trades, transport and equipment operators and related occupations
8	Natural resources, agriculture and related production occupations
9	Occupations in manufacturing and utilities

NOC assigns a four digit code to each occupation title: the first digit indicates the first level category; the first two digits indicate the second level category, the first three digits indicate the third level category, and all four digits indicate the fourth level category. Figure 2.1 identifies the four levels of the hierarchy in the four-digit NOC code for a web developer.

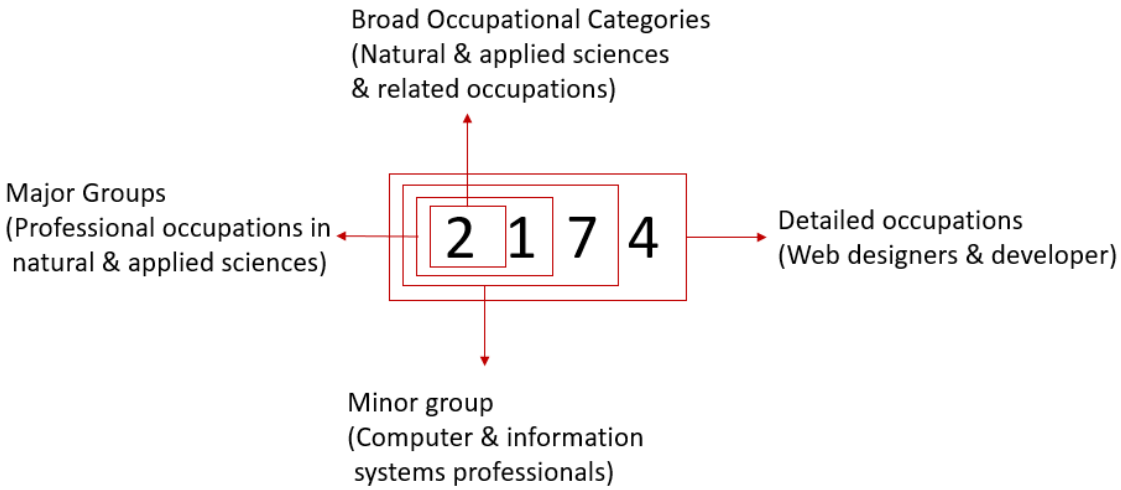

Figure 2.1: The NOC-2011 four-digit code for a web developer

SOC categorizes jobs in the American labor market based on skill and education levels [73]. SOC-2010 has 1,421 classes organized in a hierarchical structure with four levels: there are 23 *major groups* (first level), 97 *minor groups* (second level), 461 *broad occupations* (third level), 840 *detailed occupations* (fourth level). The 23 classes in the first level of the SOC-2010 hierarchy are listed in Table 2.2.

Table 2.2: SOC-2010 First Level Categories

SOC Code	**Occupation Title**
11	Management occupations
13	Business and financial operations occupations
15	Computer and mathematical occupations
17	Architecture and engineering occupations
19	Life, physical, and social science occupations
21	Community and social services occupations
23	Legal occupations
25	Education, training, and library occupations
27	Arts, design, entertainment, sports, and media occupations
29	Healthcare practitioners and technical occupations
31	Healthcare support occupations
33	Protective service occupations
35	Food preparation and serving related occupations
37	Building and grounds cleaning and maintenance occupations
39	Personal care and service occupations
41	Sales and related occupations
43	Office and administrative support occupations
45	Farming, fishing, and forestry occupations
47	Construction and extraction occupations
49	Installation, maintenance, and repair occupations
51	Production occupations
53	Transportation and material moving occupations
55	Military specific occupations

Each job title is given a six-digit code: the first two digits indicating the first level category, the first three digits indicating the second level category, the first five

digits indicating the third level category, and the entire six digits indicating the fourth level category. Figure 2.2 identifies the four levels of the hierarchy in the six-digit SOC code for a web developer. To put things into perspective, Fig. 2.3 shows a sub-hierarchy in SOC-2010 for computer-related occupation, which breaks down computer occupations into six broad occupations and 13 detailed occupations.

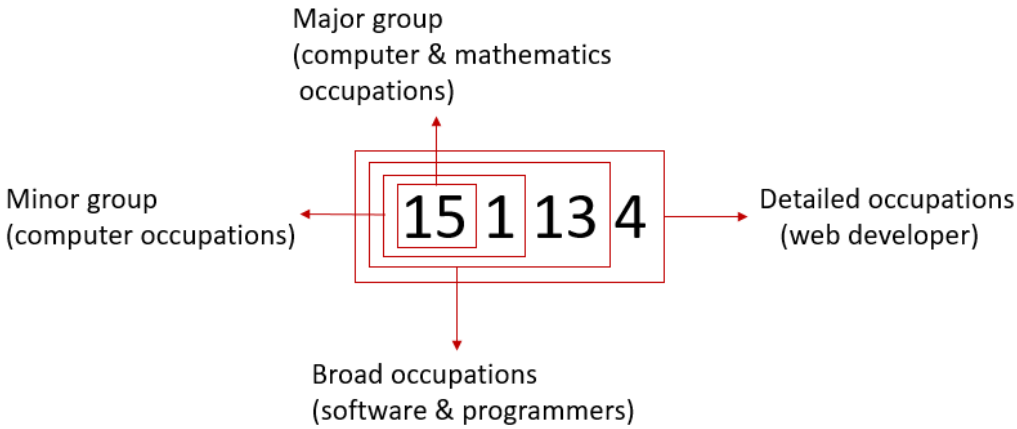

Figure 2.2: The SOC-2010 six-digit code for a web developer

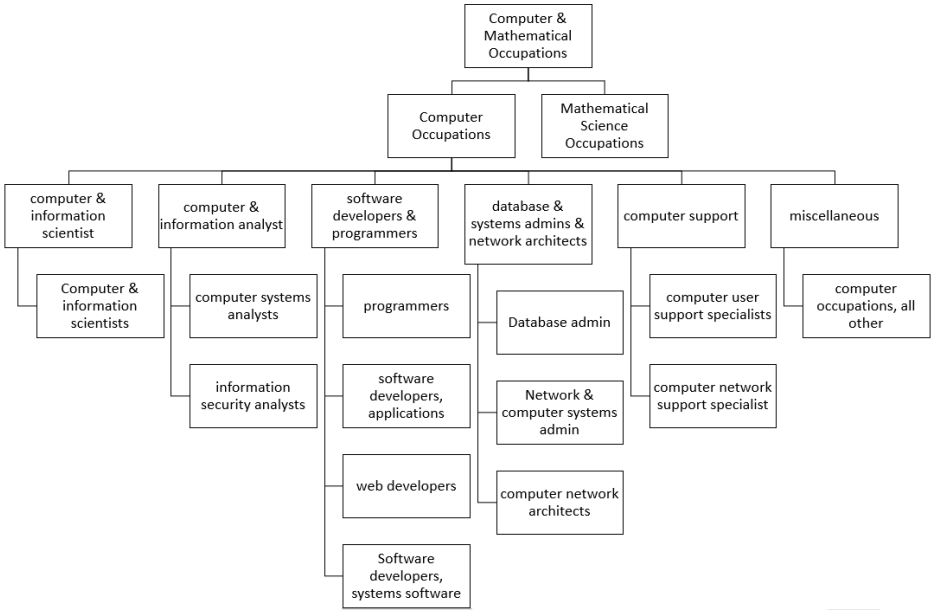

Figure 2.3: The SOC-2010 hierarchy for computer related occupations

SOC and NOC have a similar structure. They both classify occupations based on a four-level hierarchy. The first level of the hierarchy is the most general and the fourth level is the most detailed. In this work, we code occupation titles to SOC. We justify using SOC instead of NOC based on the following reasons: (1) The availability of SOC labeled occupations through publicly available datasets allowed us to train supervised machine learning algorithms on SOC codes. (2) the SOC codes are mapped to detailed skill, education, etc requirements by ONET.

If necessary, SOC codes can be mapped to NOC codes using crosswalks. Crosswalks provide guidelines for mapping occupation codes from one standard classification to another. While no crosswalk exists between SOC-2010 and NOC-2011, both classifications are already mapped to ISCO-2008. *International Standard Classification of Occupations (ISCO)* [24] is published by United Nations and classifies international occupations in a four-level hierarchy. Using officially published crosswalks SOC-2010 can be mapped to ISCO-2008 and then ISCO-2008 codes can be mapped to NOC-2011 [1].

2.2 Text Classification Overview

In this section, we review the common steps in automatic text classification, including text pre-processing (Section 2.2.1), feature representation and selection (Sections 2.2.2 and 2.2.3), and appropriate classification methods (Sections 2.2.4 and 2.2.5). Figure 2.4 shows the text classification process.

[1]The NOC crosswalks can be found in https://www.statcan.gc.ca/eng/concepts/concordances-classifications and SOC crosswalks can be found in https://www.bls.gov/soc/soccrosswalks.htm

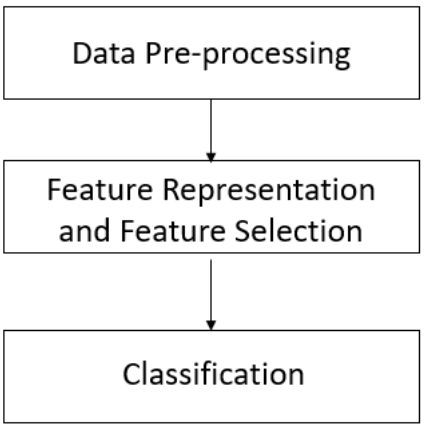

Figure 2.4: Text classification overview

2.2.1 Text Pre-processing

Text documents have a high degree of variability such as grammatical forms, formal or informal language, synonyms, spelling errors, standard and non-standard abbreviations. Automatic methods perform better when all entries are written with the same standard. Therefore, before text is automatically classified, a pre-processing step takes place. Some of the pre-processing techniques such as *abbreviation expansion*, *lowercasing*, and *removing non-alphabetical characters* and *extra white spaces* are commonly performed [9, 32, 55, 56, 58]. Others, such as *stemming* and *stop words removal*, depend on the type of data that is being used. For short text such as job titles, many studies suggest removing stop words, but not stemming [18, 21, 37, 74] or doing neither [76]. Stemming reduces words to their stems. For instance, "analytically" and "analytic" are both reduced to their stem "analyt". In many cases, stemming improves the classification accuracy by making more words connected through stems. On the other hand, stemming can lead to loss of information, especially for short text that already falls short of enough information. In every language, many common words appear in almost all documents such as 'the', 'a', and 'an' in English. These

words are called *stop words*, and can be removed from the text since they do not play a role in the distinction between the classes. Removing stop words can help with dimension reduction, avoiding overfitting, and improving the classification accuracy. However, short text tends to have less stop words in comparison with longer text. Therefore, removing stop words from short text tends to have less significant effect. Looking at the previous work on occupation classification, Gweon et al. [18] used stop words removal and stemming. However, Schierholz [58] and Kirby et al. [32] did not use stemming or stop words removal. We perform experiments on our data to analyze the effects of these two pre-processing methods.

2.2.2 Feature Representation

In Natural Language Processing (NLP), we split the text by white space into tokens. For instance, the text can be represented in terms of single tokens (unigrams), two consecutive tokens (bigrams), three consecutive words (trigrams), etc. A common feature representation method is the *Bag of Words (BOW)* [57]. BOWs represents text in terms of a bag of tokens or consecutive tokens (n-grams). The BOW feature vector represents frequencies or occurrences of tokens in a sparse vector. A binary feature vector only records occurrences of features. Figure 2.5 shows the BOWs representation for occupation titles "vice president", and "chief executive officer" with vocabulary = {officer, vice, chief, president, executive}.

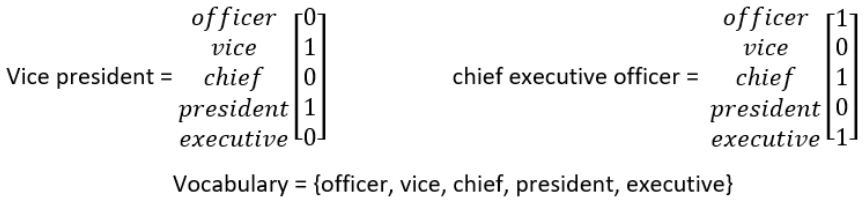

Figure 2.5: An example of BOWs feature vectors with unigrams for "vice president" and "chief executive officer" occupation titles.

BOWs cannot record context or grammar since it divides text into discrete units. While unigrams cannot record any word order, n-grams with $n > 1$ can record some order in the document. Previous research shows bigrams can improve text classification [6, 16, 76]. However, using n-grams with $n > 1$ adds sparsity to the feature vector because a mix of words occur fewer times in comparison with single words in a document. Despite the weaknesses of the BOWs model, it has been the standard approach for many years and produced good results. In our experiments with classic machine learning algorithms (i.e., Naïve Bayes, Maximum Entropy, and SVM), we compare unigram features with unigram + bigram features. We do not use n-grams with $n > 2$ for two reasons. First, our records are short with an average of three words per record. Second, short text suffers from sparsity. Using greater n-grams adds to the problem of sparsity. As will be discussed later in Section 3, our feature vector is binary. A binary feature vector only records occurrences of words (one if a word exists, and zero if it does not exist). The disadvantage of a binary feature vector is that all of the features are equally important, and feature weighting methods cannot be used. Feature weighting methods [79] assign weights to features based on their importance to the classification.

More recently, deep learning models use distributed text representations instead of BOWs. Distributed representations of words, also called *word embeddings*, map words to a real-valued dense vector so that the attributes of the features (in this case, words) can be represented through distance. As a result, words that are syntactically and semantically similar, are located close together in the embedding space. However, word embedding models have a considerably higher training time than BOWs models. Embeddings can be learned during the text classification process in a supervised manner. The vectors are initialized with small random numbers and are updated in the training process. Another approach is to use embeddings that are pre-trained in an unsupervised manner on a large corpus. In this case, the words in the text are replaced with their word embedding by a look-up table. If the em-

bedding for a word is not available, the vector for the word is randomly initialized. There are two generic methods to learn word embeddings, *count-based* and *predictive* methods. Count-based models count co-occurrences of neighboring words in a large corpus and then map these statistics to a dense vector for each word. *Global vectors for representation (GloVe)* [48] is an example of a commonly used count-based embedding. GloVe captures the global statistics of the co-occurrences of words in the entire corpus. Predictive models try to predict between a center word and its neighbors in terms of learned dense vectors. *Word2vec* [42] is an example of a commonly used predictive embedding. The *Skip-gram* and the *Continuous Bag of Words (CBOW)* are two algorithms proposed for learning word2vec vectors. Given a center word, the skip-gram model predicts words appearing within a window of the center word. For instance, given the sentence "the Director of school of computer science", the skip-gram model identifies the words and the context they appear in. Assuming a window size of one, the skip-gram model creates the dataset of (context, center) pairs such as V = {([the, of], Director), ([of, of], school), ([of, science], computer), ([computer], science)}. The skip-gram model predicts 'the' and 'of' from 'Director', 'of' from 'school', 'of' and 'science' from computer, and 'computer' from 'science'. Given the same data, the CBOW model, predicts the center word based on a number of previous or future words.

2.2.3 Feature Selection

For many classification problems, the data may have redundant or irrelevant features that may have a negative effect on the classification results. Removing less relevant features has proven to decrease the risk of overfitting as well as improve the accuracy and speed of the classification for long texts by removing noise. Therefore, *feature selection* techniques are used to select a subset of features from the training data that are the most relevant for prediction [60]. However, feature selection on short text is challenging since the text lacks enough context to confidently extract

the word relations and importance [68]. Short text also suffers from feature sparsity. Therefore, traditional feature selection methods can even decrease the classification accuracy [38, 49]. In the previous work on occupation coding, Kirby et al. [32] used feature selection, while many other researchers did not use feature selection at all [9, 18, 58]. To examine the effect of feature selection, we compare two of the well-known feature selection methods: *Document Frequency (DF)* and *Chi-Square (X^2)* on our data. DF suggests that terms that appear in more documents are more important. For each term, the frequency of documents in which it occurred are calculated. If document frequency of a feature is not within a threshold, that feature can be removed. X^2 examines the dependency between a feature and the target class. If a feature is independent of the target class, it can be removed. Both methods are further explained in [80]. Yang et al [80] report DF, X^2, and Information Gain (IG) as the best feature selection methods for text categorization. In another study on short text feature selection, Rosa and Ellen [53] reported that X^2 and DF preformed better than IG and Mutual Information feature selection methods. Rogati et al. [52] also reported that X^2 constantly outperformed other feature selection methods for text classification.

2.2.4 Multi-label Classification

In *Multi-label text classification* problems, a record can belong to more than one class. One approach to solving multi-label classification problems is to transform the problem to one or multiple single label classifications. The most common problem transformation methods are a random selection of one class per record; removal of all records associated with more than one class; combining the multiple labels associated with one record as one class, and implementing a binary classifier for each class. As discussed in Section 3, some of the occupations in our data are associated with more than one SOC code. Therefore, we transform the multi-label occupation coding problem to n binary classification problems, where n is the number of classes. Figure

2.6 shows the multi-label classification process. The binary classifiers output a level of confidence (probability or distance from the hyper-plane) for each class. The records are classified as positive or negative based on the confidence level. The classifiers are independent; therefore, there is the possibility that an occupation is assigned more than one code or an occupation is not assigned any code.

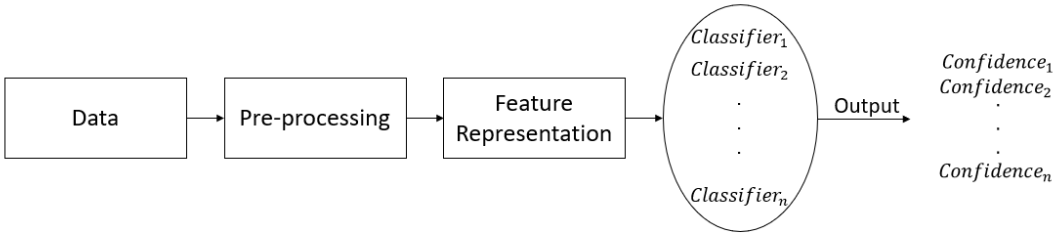

Figure 2.6: Multi-label classification

2.2.5 Hierarchical Classification

SOC is organized in a hierarchy of four levels, where each child node has only one parent node. There are two general approaches to solving hierarchical text classification problems. The *Flat Classification* method ignores the hierarchy. All classifiers are trained and tested only on one level of the hierarchy. The flat classification is simpler to implement, but its disadvantage is that one classifier needs to differentiate between a large number of classes without considering the parent-child relationship. Figure 2.7 shows the flat approach in a hierarchy with two levels. The flat classifier is trained and tested on the second level to differentiate between the four classes. The flat classifier is unaware of the existence of the higher levels of the hierarchy.

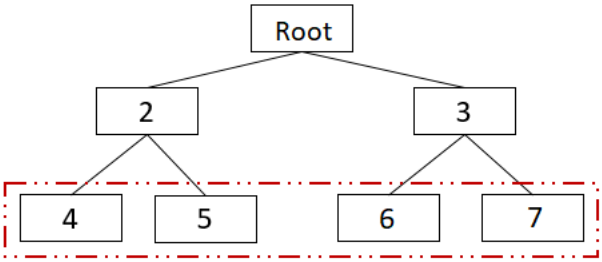

Figure 2.7: The Flat classification approach

The *Hierarchical Classification* method recognizes the hierarchy in that we train a binary classifier for each node of the hierarchy except the root. Each node represent a class in the SOC hierarchy. The hierarchical classifier can classify a record to a leaf node or any node in the hierarchy depending on the design. The disadvantage of hierarchical classification is that errors in the higher levels of the hierarchy are propagated to the rest of the classification. Figure 2.8 shows the hierarchical approach on a hierarchy with two levels. A binary classifier is trained for each of the nodes that chooses whether an example belongs to the node or not. In this work, we implement a hierarchical model that classifies records to the leaf nodes. A binary classifier is implemented for each node of the hierarchy. If the leaf node and its ancestor nodes create a path from the first level to the last level of the hierarchy, the path is assigned to the record . If at least one path does not exist, we implement a flat classifier on the last level and infer its ancestor nodes.

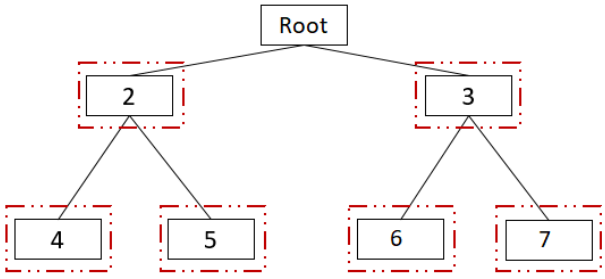

Figure 2.8: The hierarchical classification approach

2.3 Classifiers

We compare the performance of Naïve Bayes, Maximum Entropy (MaxEnt), and Support Vector Machines (SVM) classifiers on occupation coding. SVM maximizes a geometric margin between classes. Naïve Bayes and MaxEnt both are probability based, except that Naïve Bayes assumes that all features are independent. Kazama and Tsujii [29] report that SVM outperformed MaxEnt on the classification of abstracts. In another study, MaxEnt performed better than SVM with small training data but was outperformed by SVM with larger training data [77]. Both studies use unigram features. Other studies also report SVM performing better than Naïve Bayes on short text [53]. We also apply a Convolutional Neural Network (CNN) that has recently gained popularity in text classification. We explain each of the classifiers in the following subsections.

2.3.1 Naïve Bayes

As explained in [41], Naïve Bayes is a probabilistic learning algorithm that learns $p(X|y)$ and $p(y)$, where y is the class label, and X is the vector of features $(x_1, x_2, x_3, ..., x_n)$. In other words, Naïve Bayes learns the features conditioned on each of the classes and the prior probability of each of the classes. Given a new X, the model computes $p(y_i|X)$ and use this probability to pick the best class. According to the Bayes Theorem, $p(y_i|X)$ is computed as shown in equation 2.1.

$$p(y_i|X) = \frac{p(X|y_i)p(y_i)}{p(X)} \tag{2.1}$$

The numerator of the fraction is equal to:

$$
\begin{aligned}
p(x_1, x_2, ..., x_n|y_i)p(y_i) &= p(x_1, x_2, ..., x_n, y_i) \\
p(x_1, x_2, ..., x_n, y_i) &= p(x_1|x_2, ..., y_i)p(x_2, ..., x_n, y_i) \\
&= p(x_1|x_2, ..., y_i)p(x_2|x_3, ..., x_n, y_i)p(x_3, ..., x_n, y_i) \\
&= ... \\
&= p(x_1|x_2, ..., x_n, y_i)p(x_2|x_3, ..., x_n, y_i)...p(x_{n-1}|x_n, y_i)p(x_n|y_i)p(y_i)
\end{aligned}
$$

Naïve Bayes assumes that all features are independent of each other. Therefore, the conditional probability $p(x_j|x_{j+1}, ..., x_n, y_i)$ becomes equal to $p(x_j|y_i)$. So, the equation can be rewritten as:

$$p(y_i|X) \propto (\prod_{j=1}^{j=n} p(x_j|y_i))\frac{p(y_i)}{p(x_1, x_2, ..., x_n)}$$

Since $p(x_1, x_2, ..., x_n)$ is a constant for all the classes, the equation can be rewritten as:

$$p(y_i|X) \propto (\prod_{j=1}^{j=n} p(x_j|y_i))p(y_i)$$

Naïve Bayes algorithm is based on the naive assumption that features are independent, implying that the occurrence of one word does not affect the occurrence of another word. However, this assumption is generally incorrect. For example, the phrase "Hong Kong" contains two words that almost always occur together.

2.3.2 Support Vector Machines

The goal of SVM is to find a hyper-plane that maximizes the margin between the data points of different classes [19]. This margin is the distance of the hyper-plane from the closest data points of each class. These usually small subsets of data that

specify the hyper-plane are called support vectors. A large margin can be expected to produce a small number if incorrect classifications provided there are enough data samples in the margin area to accurately map this critical region. SVM algorithm performs well in the high dimensional space since it does not depend on all of the features [11]. Figure 2.9 shows a simple linear SVM that finds the optimum hyper-plane between empty and filled circles.

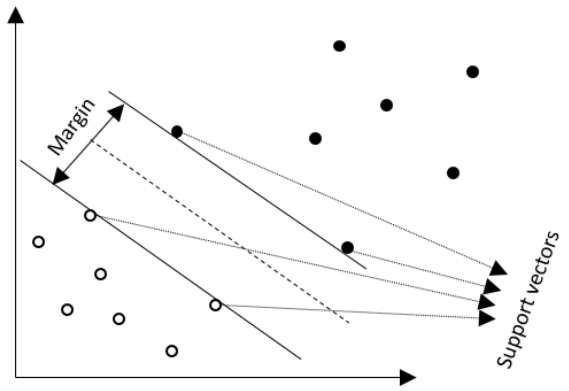

Figure 2.9: A linear SVM with the optimum hyper-plane between two classes

We use Scikit-learn [47] implementation of linear SVM that solves the problem in equation 2.2 where $y \in \{1, -1\}^n$. In this equation, n is the set of labels, w is the weight vector, and y represents the labels. The C parameter ensures a balance between maximizing the margin and correctly classifying as many points as possible. As recommended by Hsu et al. [22], we use *grid search* to find a reasonable value for C. Grid search is a technique that compares the performance of a classifier with different values of its parameters. The parameter values that lead to the best performance in the *validation set* are used for the classification. The validation set is a separate subset of the data that is used for parameter tuning.

$$min\frac{1}{2}w^2 + C\sum_{i=1}^{n} max(0, 1 - y_i(w.x_i - b)) \tag{2.2}$$

2.3.3 Maximum Entropy

Maximum Entropy is a probability estimation model that is based on the *maximum entropy principle*. This principle suggests that we model what we know and assume nothing about what we do not know. In other words, it models the constraints that are derived from the training data, and distributes the rest of features as uniform as possible. Entropy measures the uncertainty of a distribution. The MaxEnt algorithm finds the model that maximizes the entropy from the models that satisfy the constraints [29]. Given an event x, an output y, and a set of features $F(x, y)$, the empirical expectation of the model is equal to equation 2.3 and the model expectation is equal to equation 2.4. $\widetilde{p}(x)$ and $\widetilde{p}(y|x)$ are probabilities derived from the training set.

$$E_{\widetilde{p}}[f_i] = \sum_{x,y} \widetilde{p}(x)\widetilde{p}(y|x)f_i(x, y) \tag{2.3}$$

$$E_p[f_i] = \sum_{x,y} \widetilde{p}(x)p(y|x)f_i(x, y) \tag{2.4}$$

MaxEnt chooses the model that maximizes the entropy shown in equation 2.5 conditional on $E_{\widetilde{p}}[f_i] = E_p[f_i]$.

$$H(p) = -\sum_{x,y} \widetilde{p}(x)p(y|x)logp(y|x) \tag{2.5}$$

There are many algorithms for parameter estimation of MaxEnt. We used Limited-memory BFGS (LBFGS) algorithm as recommended by [40]. MaxEnt extracts all constraints from the training data. Consequently, it can be prone to overfitting, mainly, if the training data is small and sparse. Therefore, we use $L2$ regularization (equation 2.6). $L2$ adds a penalty of sum of squares of weights w to the MaxEnt's cost function.

$$L2 = \sum_{i=1}^{n} w_i^2 \tag{2.6}$$

We use the Scikit-learn's implementation of MaxEnt with L2 regularization, which minimizes the cost function in equation 2.7. In order to choose a reasonable value for the regularization variable C, which controls the strength of the L2 regularization, we used grid search on the validation set.

$$min\frac{1}{2}w^2 + C\sum_{i=1}^{n} log(exp(-y_i(x_i^T w + c)) + 1) \tag{2.7}$$

2.3.4 Convolutional Neural Network

CNN has been successful in computer vision for tasks such as image classification and object detection. More recently, CNNs have been used in NLP. CNN has been particularly successful in text classification tasks [28, 31]. Often, the equivalent of a pixel in an image classification is a vector representation for a word in text classification. However, the vector representation of characters and sentences can also be used. CNN consists of multiple convolution and pooling layers with a fully connected layer at the end. In text classification, CNN learns dense real valued vectors that represent words and then applies convolving filters over the word vectors [31]. We implemented a wide CNN that has one convolution, one max-pooling, and one fully connected layer. To choose the CNN architecture, we compared the performance of a CNN with multiple convolution and pooling layers with a CNN with one convolution and one pooling layer. We decided to use a CNN with one convolution, one pooling, and one fully connected layer since its performance was the same as the deeper CNN. Next, we use the mathematics notations used in [31] and [28] to describe CNN.

In the first layer, called the embedding layer, the words are replaced with their vector representation. The records are padded to have the same length n. If the i'th word in the record is represented with a k dimensional vector $x_i \in R^k$, the entire

record would be represented as in equation 2.8, where $x_{i:n}$ refers to concatenation of words $x_1\ x_2, x_3, ..., x_n$.

$$x_{1:n} = x_1 \cdot x_2 \cdot ... \cdot x_n \tag{2.8}$$

Next, the Convolution layer detects features in the data by applying convolution filters on different window sizes of words. For instance, we can have filters that are applied to bigrams and trigrams. Sequence c_i is computed by the dot product of filter $w \in R^{hk}$ and a window size h and is passed through a nonlinear function $f(x)$ as shown in equation 2.9.

$$c_i = f(wx_{i-h+1:i} + b) \tag{2.9}$$

In the wide type of convolution, the index i ranges from 1 to $n + h - 1$. The result of applying the filter w to all possible windows of length h is a feature map $C = [c_1, c_2, ..., c_{n+h-1}] \in R^{n+h-1}$.

The features extracted in the convolution layer are passed through the pooling layer to reduce the representation, aggregate the information in a feature map, and make sure that all of its outputs have a fixed dimension. The max pooling extracts the maximum value from the feature map resulting in $\hat{C} = max\{C\}$. CNN has filters with different sizes that create multiple feature maps. Therefore, given m filters, $Z = \hat{C_1} \cdot \hat{C_2} \cdot ... \cdot \hat{C_m}$ is passed to the next layer.

In the final layer, Z is passed through a fully connected Sigmoid layer as shown in equation 2.10. The Sigmoid function assigns an independent probability to each label allowing multi-label classification.

$$Y = Sigmoid(w^{(z)}Z + b) \tag{2.10}$$

Figure 2.10 shows the architecture of the CNN given p records with length n, and k dimensional word vectors. Filter size is represented by h, number of classes is represented by j, and the number of filters is represented by m.

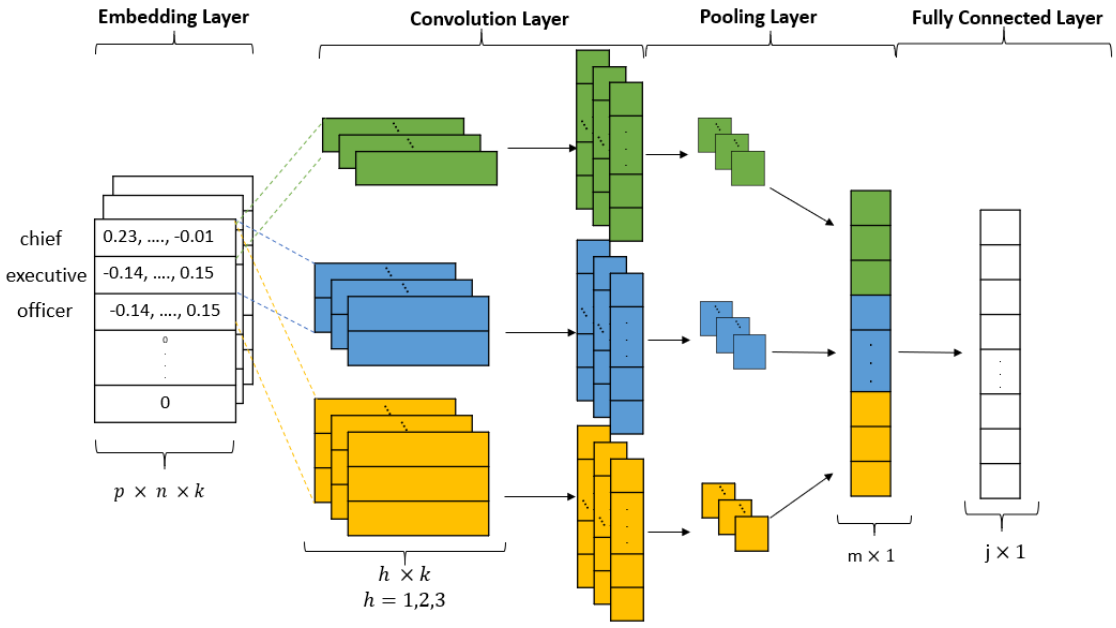

Figure 2.10: CNN Architecture

2.4 Related Work

Occupation coding methods are mainly divided into *manual*, *computer-assisted*, and *automatic*. Figure 2.11 shows an overview of occupation coding approaches.

Historically, the task of classifying job titles was done manually by professional coders based on specific instructions. Manual coding is time-consuming and costly. Restricted availability of professional coders is also a problem [46]. To demonstrate how challenging manual occupation coding is, Patel et al. [46] conducted a study where a human coder was given the same 165 job titles with a 15 month gap. The codes matched on only 76% of occupations across the two submissions. In another study conducted by Koeman et al. [33], two human coders classified 220 job titles. The two coders agreed on only 55% of the codes. To ensure high quality, training, specific coding instructions, and evaluation rounds are provided for human coders. For example, American Community Survey coders are expected to have less than 5% coding error [67]. Manual classification of occupation titles with high quality is very

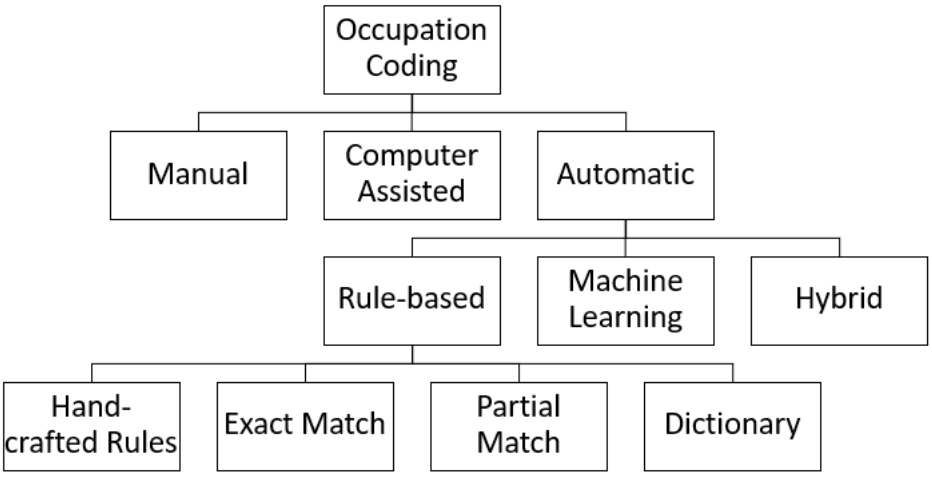

Figure 2.11: An overview of the occupation coding approaches

expensive. For instance, United States Census Bureau spent approximately seven million dollars on manually coding occupation titles in 1980 [10]. Automating the process of occupation coding can solve many of the issues discussed above.

In computer-assisted coding, a computer software is used to simplify coding. The software suggests a number of suitable categories, then the human coder decides the best option [58]. Computer-assisted coding is more time and cost effective than manual coding. However, it still depends on a human coder to make the final decision. The computer-assisted Structured Coding Tool (CASCOT) [71] was produced by the University of Warwick and is available for coding to the U.K. standard occupational classification.

In the case of automatic occupation coding, a computer program assigns a class to the job titles. Automatic occupation coding is time and cost effective, and it is not prone to human errors. However, achieving fully automatic occupation coding with high quality remains challenging [18]. One approach to solving this problem is

to automate the coding of easier to code entries with a high accuracy and to leave the harder to code records for manual coding [18, 46, 67]. In other words, if the accuracy of automatic classification falls below a certain threshold, a human coder is asked to categorize the occupation. Manual inspection can help with keeping a high accuracy rate as well as understanding the reasons that an algorithm was not able to classify the text correctly [67]. Previous work on automatic classification of occupations can be further divided into three main categories: the *rule-based* approach, the *machine learning* approach, and the *hybrid* approach. These three approaches are discussed in the following sections. Table 2.3 summarizes the performance of the previous studies on automatic occupation coding. Production rate represents the desired proportion of occupations to be coded automatically. For a given production rate, accuracy represents the proportion of occupations that were coded correctly.

Algorithm	**Production Rate**	**accuracy**	**Data Size**	**classification**	**Number of classes**
Dictionary (ACS)	43%	94.14%	2.3 million	SOC	840
Ensemble (SOCCER)	100%	44.5% 76.3%	76,983	SOC	840 23
Hierarchical SVM + KNN Flat KNN (Carotene)	100%	80% 68%	2 million	SOC-ONET	1,800
Naïve Bayes Boosting	100% 100%	63.23% 63.64%	32,882	German	1,286
Rules + MaxEnt	73%	98%	2 million data+3,800 rules	Korean	450
Rules + SVM	100%	74.8%	20,066 data+3,551 rules	Japanese	200
KNN Exact Match + SVM	100% 100%	65% 65%	9,137 (53% duplicates)	ISCO	399
SVM Perceptron Partial match Rules	100% 100% 100% 100%	55.6% 48.3% 50% 44.4%	412	CP2011	62

Table 2.3: A summary of the performance of occupation coding models

2.4.1 Rule-Based Approach

A *rule-based classifier* uses a number of rules, which are created based on expert knowledge and previous data analysis to classify free-text job titles. In a rule-based system, each rule can contribute to more than one class, and each class can be as-

sociated with more than one rule. If a rule leads a job title to be classified to more than one class, a scoring mechanism is used. In this scoring mechanism, if a job title matches a rule, the corresponding classes each receive a score. This procedure is repeated for each rule, and scores are accumulated for each class. In the end, the answer is classified to the class with the highest score [18].

A simple technique is to use a database of job titles and their codes. If the new job title is identical to a job title in the database, the corresponding code will be assigned [18, 58]. This method is known as the *exact match* or the *duplicate method*. Another technique uses language similarities between new job titles and the job titles in the database. Jung et al. [27] used Cosine Similarity and Kirby et al. [32] used Edit Distance to assign the occupation code that is the most similar to the job title that is being classified. This technique is known as the *partial match*. Partial matches can produce a score that shows the strength of the match. A method that is used by many organizations is to create dictionaries of keywords and occupation codes that are commonly associated with these keywords. The keywords usually include unigrams, bigrams, and trigrams in a job title. Different weights are assigned to entries in the dictionary based on their frequency. Phrases that correspond to fewer codes have a higher weight than phrases that correspond to many different occupation codes. Depending on the system's design, an exact match, a partial match, or both would be allowed.

Rule-based methods have been extensively used by governmental institutions in many countries. Automatic Coding by Text Recognition (ACTR) [69, 78] is a dictionary based software created by Statistics Canada to code survey responses to standard classifications. ACTR is a general system, meaning that it is not specific to a language or a classification, as long as the user prepares the relevant dictionary. ACTR achieved 42.7% production rate and 31.1 error rate when coding occupations in 1991 Census [54]. ACTR was later used in Italian language [39]. The CHUM research center, university of Bordeaux, and French National Public Health agency

created the Caps-Canada website [14]. Caps-Canada uses partial matching for coding occupations to different classifications. The U.S. Census Bureau has implemented numerous automated coding systems since the mid-1960s [30]. The Bureau created a dictionary based Automated Industry and Occupation Coding System (AIOCS) for coding industry and occupations of the 1990 Census [15]. The Washington State Department of Health used word matching with a dictionary [45]. The U.S. Census Bureau also used dictionaries to automatically assign 43% of SOC codes for 2012 American Community Survey (ACS) with 95% accuracy [67]. The U.S. Information Technology Support Center (ITSC) produced the OccuCoder software [43] for coding occupations to SOC. OccuCoder uses word matching with ONET database. The U.S. National Institute for Occupation Safety and Health (NIOSH) produced a web-based system that classifies industry and occupation titles called NIOSH Industry and Occupation Computerized Coding System (NIOCCS) [66]. NIOCCS uses exact and partial matches as well as logical rules, to code occupations to SOC.

Many rule-based techniques have been implemented by statistical agencies in different languages. However, expert knowledge, manual analysis, and huge amounts of labeled data are used for creating and updating the rules. Machine learning algorithms could be an efficient and reliable alternative to rule-based methods.

2.4.2 Machine Learning Approach

Supervised machine learning infers patterns and rules from a training dataset that is already labeled with correct labels, and then uses these patterns to classify a new instance. Various authors used supervised machine learning algorithms to code occupations to standard classifications. Different accuracy rates have been achieved, depending on the size of the training data, the level of clarity of the data, and the algorithm used. Gweon et al. [18] implemented K-Nearest Neighbor (KNN) with a cosine similarity measure to classify occupations from the German General Survey (ALLBUS) to ISCO. In this procedure, a new record was assigned the occupation

code of its nearest neighbor. Duplicates are considered nearest neighbors with zero distance. Their method uses a cosine similarity measure to identify how similar two entries are, with one indicating duplicates and 0 indicating no similarity. They achieved 65% accuracy rate by using 10-fold cross-validation on 9,137 records, which 52.6% of them were identical. Kirby et al. [32] used Naïve Bayes algorithm for coding Scottish occupation records to Historical International Standard Classification of Occupations (HISCO). They were able to code 9,400 records with a 0.75 micro F1. Amato et al. [3] compare the performance of hand-crafted rules, partial match, SVM, and Perceptron classifier for coding job advertisements to an Italian classification called CP2011. SVM outperformed the other three methods.

Some authors have explored the use of *ensemble models*. Ensemble classification models combine multiple classifiers to achieve a better performance than each of the classifiers alone. Russ et al. [55] developed an ensemble algorithm called Standardized Occupation Coding for Computer-assisted Epidemiologic Research (SOCCER) to assign SOC codes to job titles. This model combined the results of classifiers based on Job Title, Industry Title, and Task Description into one score that identifies how closely the occupation matches a SOC code. They achieved 44.5% agreement rate with human coders on the test set. Schierholz et al. [58] reported that Naïve Bayes and boosting methods achieved 63.23% and 63.64% accuracy respectively while coding the 32,882 job titles in the ALWA Survey to German national occupation classification. Javed et al. [25] created an occupation title classification software called Carotene. The system has a hierarchical structure. SVM is used for coarse classification in the first level of the SOC-ONET classification system and KNN is used for fine grained classification for the lower levels of SOC-ONET. Another version of Carotene used flat classification with KNN [26].

2.4.3 Hybrid Approach

Hybrid algorithms are a combination of rule-based and machine learning algorithms. For instance, Jung et al. [27] offered a web-based occupation and industry coding system in the Korean language. The system uses a mix of hand-crafted rules and a Maximum Entropy model. In the first step, logical rules are used to classify the input. If no class was assigned, a Maximum Entropy algorithm is used to assign a class. They were able to achieve 98% accuracy at 73% production rate on the Korean 2005 Census. Takahashi et al. [64] implemented hand-crafted rules and SVM in the same manner. The authors reported that the combination of the two methods performed better than each of them individually. Gweon et al. [18] proposed a hybrid algorithm that combines a rule-based method (exact match) and a machine learning algorithm (SVM). This method can be divided into two steps. First, exact matches after removing the stop words are identified between the training and testing data. Then the machine learning algorithm classifies the remaining occupations in the test data based on trends learned from the training data.

Chapter 3

Data Analysis and Evaluation Measures

This Chapter presents a detailed analysis of the SOC labelled data and the appropriate evaluation measures. Section 3.1 describes the data acquisition process. Section 3.2 discusses occupations that have more than one SOC code. Section 3.3 analyzes the word counts and repetitions in records. Section 3.4 discusses the data coverage for the four levels of the SOC hierarchy. Section 3.5 summarizes the data attributes. Finally, Sections 3.6 and 3.7 introduce the evaluation measures and the statistical analysis process respectively.

3.1 Data Acquisition

The SOC labelled data is aggregated from publicly available data from the *U.S. Census Bureau*, the *U.S. Department of Labor*, and the *Occupational Information Network (O*NET)* websites in the form of CSV files. These three sources overlap on some occupation titles. The data from these sources were compared to each other to keep the unique occupation title and codes. As a result of this comparison, a data set of 65,962 unique job titles and their corresponding SOC-2010 codes was created. The following is a detailed description of each of the sources.

3.1.1 U.S. Census Occupational index (U.S. Census Bureau)

Census occupational index [72] is a list developed for classifying occupations reported in Census Bureau demographic surveys. The Census occupational index lists 32,076 unique occupation titles with their SOC, North American Industry Classifica-

tion System (NAICS) [23], and Census occupation and industry codes. We use the 2010 U.S. Census occupational index since it is the most recent occupational index with SOC codes at the time of this research. The data includes occupations that are coded to more than one SOC code. Same occupation titles may have different SOC codes based on their industries. For instance, an account analyst in insurance related activities is assigned 13-1031 (Claims Adjusters, Examiners, and Investigators) SOC code, whereas an account analyst in banking occupations is assigned a 43-3021 (Billing and Posting Clerks) SOC code. Figure 3.1 shows a snapshot of the unprocessed Census data obtained from the website. The "2010 SOC CODE" and the "2010 OCCUPATION TITLE" columns were collected for the final dataset.

2010 Census Occupation Code	2010 SOC CODE	2010 OCCUPATION TITLE	2010 Census Industry Restriction	2007 NAICS Industry Restriction
8965	51-9199	A mill operator		
5020	43-2021	A operator	6680	5171
8965	51-9199	A operator	Exc. 6680	Exc. 5171
9300	53-5011	A.B. seaman	#6090	#483
5940	43-9199	A.C.P. clerk		
1050	15-1151	A.D.P customer liaison		
1060	15-1141	A.D.P planner		
1105	15-1142	A.D.P system coordinator		
1007	15-1122	A.D.P systems security		
2000	21-1019	A.S.A.T. C.O.R.E. counselor		
7010	49-2011	A.T.M specialist		
2000	21-1019	AIDS counselor		

Figure 3.1: The 2010 Census occupational index before pre-processing

3.1.2 SOC Structure and Direct Match Title Files (U.S. Department of Labor)

The SOC Structure File [73] contains 1,421 titles in all four levels of hierarchy, which are broken down into 23 major groups, 97 minor groups, 461 broad groups, 840 detailed occupations. Figure 3.2 shows a snapshot of SOC Structure File before pre-processing. The 'Detailed occupation' and 'Occupation Title' columns were collected

for the final dataset.

Major Group	Minor Group	Broad Group	Detailed Occupation	Occupation Title
11-0000				**Management Occupations**
	11-1000			**Top Executives**
		11-1010		Chief Executives
			11-1011	Chief Executives
		11-1020		General and Operations Managers
			11-1021	General and Operations Managers
		11-1030		Legislators
			11-1031	Legislators
	11-2000			**Advertising, Marketing, Promotions, Public Relations, and Sales Managers**
		11-2010		Advertising and Promotions Managers
			11-2011	Advertising and Promotions Managers
		11-2020		Marketing and Sales Managers
			11-2021	Marketing Managers
			11-2022	Sales Managers
		11-2030		Public Relations and Fundraising Managers
			11-2031	Public Relations and Fundraising Managers

Figure 3.2: SOC Structure File before pre-processing

Direct Match Title File [73] is published by the U.S. Department of Labor. It contains 6,576 unique job titles and their SOC codes. This file also includes some job titles that are coded to more than one SOC code. Figure 3.3 shows a snapshot of the direct match file before pre-processing. The '2010 SOC Code' and the '2010 SOC Direct Match Title' columns were collected for the final data set.

2010 SOC Code	2010 SOC Title	2010 SOC Direct Match Title
11-1011	Chief Executives	CEO
11-1011	Chief Executives	Chief Executive Officer
11-1011	Chief Executives	Chief Operating Officer
11-1011	Chief Executives	Commissioner of Internal Revenue
11-1011	Chief Executives	COO
11-1011	Chief Executives	County Commissioner
11-1011	Chief Executives	Government Service Executive
11-1011	Chief Executives	Governor
11-1011	Chief Executives	Mayor
11-1021	General and Operations Managers	Department Store General Manager
11-1021	General and Operations Managers	General Manager

Figure 3.3: Direct Match Title File before pre-processing

3.1.3 ONET Occupation Datasets

Occupational Information Network (ONET) [1] datasets are produced with the support of the U.S. Department of Labor/Employment and Training Administration (USDOL/ETA). ONET contains occupation titles with their corresponding SOC, Occupational Information Network-Standard Occupational Classification (ONET-SOC), and NAICS codes. It also includes information on knowledge, skills, abilities, education, experience, training, interests, tools and technology, work values, work styles, and tasks related to each occupation. ONET-SOC is developed based on SOC six-digit codes. ONET-SOC includes the six-digit SOC code followed by two decimal digits, which provide a more detailed occupation classification. ONET-SOC is compatible with the SOC system, and the SOC code is accessible by removing the decimal integers. Figure 3.4 shows ONET-SOC to SOC crosswalk.

O*NET-SOC 2010 Code	O*NET-SOC 2010 Title	SOC 2010	SOC 2010 Title
11-1011.00	Chief Executives	11-1011	Chief Executives
11-1011.03	Chief Sustainability Officers	11-1011	Chief Executives
11-1021.00	General and Operations Managers	11-1021	General and Operations Managers
11-1031.00	Legislators	11-1031	Legislators
11-2011.00	Advertising and Promotions Managers	11-2011	Advertising and Promotions Managers
11-2011.01	Green Marketers	11-2011	Advertising and Promotions Managers
11-2021.00	Marketing Managers	11-2021	Marketing Managers
11-2022.00	Sales Managers	11-2022	Sales Managers
11-2031.00	Public Relations and Fundraising Managers	11-2031	Public Relations and Fundraising Managers

Figure 3.4: ONET-SOC crosswalk to SOC

Figure 3.5 shows a snapshot of one of the ONET occupation dataset, the 'ONET-SOC code' and the 'Title' columns are collected from the data. The ONET-SOC codes were converted to SOC codes to create the final SOC labeled dataset. By combining all ONET databases, we collected 58,911 unique job titles and SOC codes. This dataset also contains job titles that are coded to more than one SOC code.

O*NET-SOC Code	Title	Element ID	Element Name	Scale ID	Scale Name	Data Value	N	Standard Error	CI Bound	Upper CI Bound	Recommend Suppress	Not Relevant	Date	Domain Source
11-1011.00	Chief Executives	2.C.1.a	Administration and Management	IM	Importance	4.75	27	0.09	4.56	4.94	N		07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.a	Administration and Management	LV	Level	6.23	27	0.17	5.88	6.57	N	N	07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.b	Clerical	IM	Importance	2.66	27	0.22	2.21	3.11	N		07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.b	Clerical	LV	Level	3.5	27	0.41	2.66	4.34	N	N	07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.c	Economics and Accounting	IM	Importance	3.7	27	0.28	3.11	4.28	N		07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.c	Economics and Accounting	LV	Level	4.36	27	0.34	3.67	5.06	N	N	07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.d	Sales and Marketing	IM	Importance	3.23	27	0.49	2.21	4.25	N		07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.d	Sales and Marketing	LV	Level	3.9	27	0.61	2.65	5.16	N	N	07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.e	Customer and Personal Service	IM	Importance	4.09	27	0.33	3.4	4.78	N		07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.e	Customer and Personal Service	LV	Level	5.55	27	0.44	4.65	6.44	N	N	07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.f	Personnel and Human Resources	IM	Importance	4.1	27	0.29	3.51	4.69	N		07/2014	Incumbent
11-1011.00	Chief Executives	2.C.1.f	Personnel and Human Resources	LV	Level	5.02	27	0.37	4.26	5.77	N	N	07/2014	Incumbent
11-1011.00	Chief Executives	2.C.2.a	Production and Processing	IM	Importance	2.63	27	0.44	1.73	3.53	N		07/2014	Incumbent
11-1011.00	Chief Executives	2.C.2.a	Production and Processing	LV	Level	2.92	27	0.68	1.54	4.31	N	N	07/2014	Incumbent

Figure 3.5: ONET dataset before pre-processing

3.1.4 Final Dataset

The final data is a collection of the datasets discussed above. It contains 65,962 unique job titles with their corresponding six-digit SOC codes. The data contains two fields, the *SOC-2010 code* and the *job title*. The final dataset has overall 11,251 unigram features. Figure 3.6 shows a snapshot of the final dataset.

code	title
111011	chief executives
111021	general and operations managers
111031	legislators
112011	advertising and promotions managers
112021	marketing managers
112022	sales managers
112031	public relations and fundraising managers
113011	administrative services managers
113021	computer and information systems managers
113031	financial managers
113051	industrial production managers
113061	purchasing managers
113071	transportation storage and distribution managers
113111	compensation and benefits managers
113121	human resources managers
113131	training and development managers
119013	farmers ranchers and other agricultural managers
119021	construction managers
119031	education administrators preschool and childcare center program

Figure 3.6: First 20 records of the final dataset

3.2 Occupations with Multiple SOC Codes

Since some of the records are associated with more than one class, we treat the coding problem as a multi-label classification. This means that some occupation titles can be associated with more than one SOC code due to two main reasons. The first reason is that the occupation titles are sometimes not as specific as the SOC codes. As can be seen in fig. 3.7, professors have different SOC codes based on their field of work. Therefore, 'professor' alone as a job title is very generic and can be associated with many codes. The second reason is that an occupation title can have a different SOC code based on the industry (Section 3.1.1). In this case, the SOC codes are different from the first level of the hierarchy.

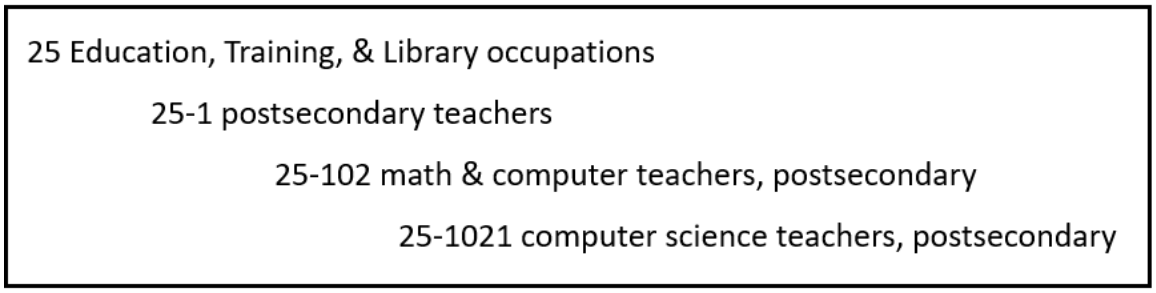

Figure 3.7: SOC hierarchy for computer science professors

As can be seen in Table 3.1 most of the occupations (85%) have only one SOC code. 11% of data has two SOC codes, 2% of data has three SOC codes, and 2% of data has more than three SOC codes. Table 3.2 shows some of the occupations that are associated with many codes.

Table 3.1: Number of SOC codes associated with a single job title

Number of SOC Codes	Number of Records	Percentage of Records
1	56,130	85%
2	7,053	11%
3	1,577	2%
4 to 57	1,202	2%

Table 3.2: A sample of occupation titles with the most SOC code variability

Occupation title	Number of Different SOC Codes
Machine operator	57
Professor	52
Instructor	43
Faculty member	28
Teacher	28
Installer	19
Production manager	16

3.3 Record Length

After lowercasing the text, removing numbers, and punctuation, we analyze the record lengths (number of unigrams as the unit) and term frequency in the records. Figure 3.8 shows the frequency of different record lengths in the data. The longest job title in the data has 24 unigrams, and the shortest job titles in the data have one unigram. The average and median number of unigrams per record is approximately 3.

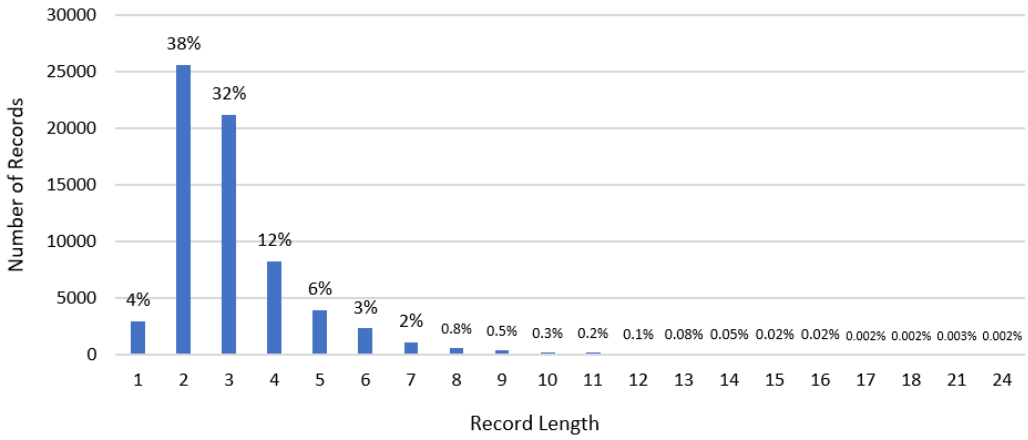

Figure 3.8: Frequency of record lengths in the data (number of unigrams as the unit)

Most of the records (70%) contain a short text with two or three unigrams, which leads to a sparse feature space. There are a few exceptions where the record contains a long description. Table 3.3 shows some of the longest records in the data. These records contain many abbreviations and repeated terms.

Table 3.3: Some of the longest records in the data

Job Title	**Number of unigrams**
ceo leed ap o m director of sustainability chief environmental officer leadership in energy and environmental design applications operations and management director of sustainability	24
aegis weapon system mk bl technician aegis fire control system mk operational readiness test system mk tk i or ii supervisor	21
hvac sensor and digital control designer heating ventilation and air conditioning sensor and digital control designer	16

3.3.1 Term Frequency

Term Frequency represents the number of times a term is repeated in one record. If there is no repetition of terms in a record, we say it has term frequency of one. In this thesis, if there is at least one term in the record that is repeated twice, we say the record has term frequency of two. As shown in Table 3.4, the records have term frequency of one for approximately 98% of the data. These records have an average record length of 3. Few records (2%) contain terms that are repeated two or three times. The average record length for this 2% of data is approximately 7. Furthermore, the average ratio of term counts to unique term counts in the record is 1.02.

Table 3.4: Term frequency and average length of records

	Percentage of Data	**Average Record Length**
Records with term frequency of 1	97.9%	3
Records with term frequency of 2 or 3	2.1%	7

Feature weighting techniques such as Term Frequency (TF) and Term Frequency-Inverse Document Frequency (TF-IDF) assign weights to terms based on their importance to the classification and the importance of a term is usually determined based

on the frequency of that term in a record. In the majority of records, there is no term repetition. Therefore, the feature vector is a binary vector that only captures occurrences of terms. As a result, feature weighting techniques do not have a notable effect on the classification.

3.4 Coverage

In this section, the data coverage for each of the four levels of the SOC hierarchy is analyzed. Records represent occupation titles and classes represent the SOC codes. As we move down the hierarchy, the number of records for each class decreases. Therefore, the classification quality is expected to decrease as we move down the SOC hierarchy. In the first level of the hierarchy, the least frequent class has 452 records, whereas in the fourth level of the hierarchy, the least frequent class has four records. Furthermore, the records are not evenly divided between classes (Fig. 3.9). For instance, the class 51-Production occupations and its child nodes have a significantly higher frequency than the rest of the classes: 27% of records belong to the class 51-Production occupations, while the rest of the 22 classes each have 0.6% to 9% of the records.

3.4.1 First Level of the Hierarchy

The data covers all 23 first level SOC classes (major groups). The most frequent class is the *51-Production occupations* with 19,103 records. The least frequent class is the *23-Legal occupations* with 452 records. Figure 3.9 shows the data coverage for the 23 first level classes. The average frequency per class is 3,110, and the median frequency per class is 2,250.

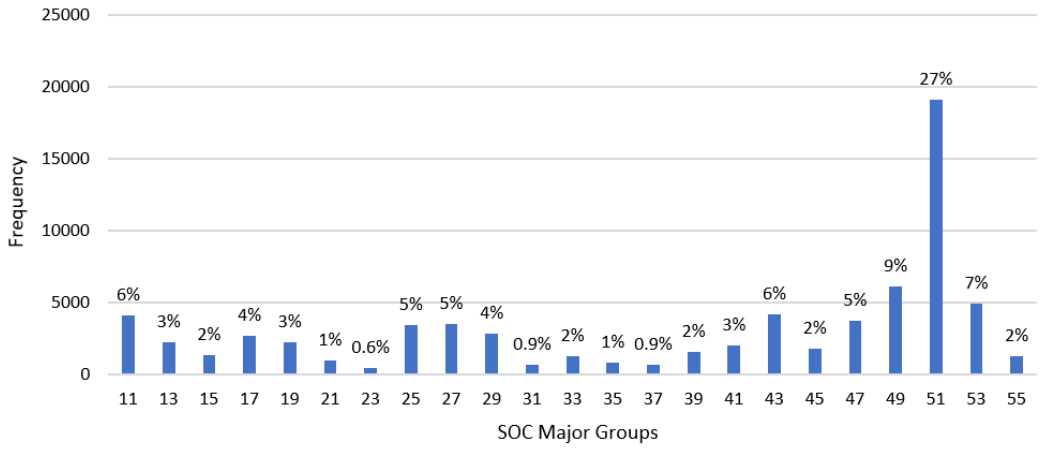

Figure 3.9: Frequency of SOC major groups

3.4.2 Second Level of the Hierarchy

The data covers all 97 second level SOC classes (minor groups). The most frequent class in the second level is *519-Other Production Occupations* with 9,824 records, and the least frequent class is *396-Baggage Porters, Bellhops, and Concierges* with 73 records. The average frequency is 781 records per class, and the median frequency is 460 records per class.

3.4.3 Third Level of Hierarchy

The data covers all 461 third level SOC classes (broad occupations). The most frequent class in the third level is *51919-Miscellaneous Production Workers* with 4,868 records, and the least frequent class is *25909-Miscellaneous Education, Training, and Library Workers* with 11 records. The average frequency of records per class is 173, and the median frequency is 98.

3.4.4 Fourth Level of the Hierarchy

The data covers all 840 fourth level SOC classes (detailed occupations). The most frequent class in the fourth level is *519199- Production Workers, All Other*

with 3,177 records and the least frequent class is *352019-Cooks, All Other* with four records. The average frequency of records per class is 98, and the median frequency of records per class is 59.

3.5 Summary of Data Attributes

We collected 65,962 unique job titles and their corresponding SOC codes from three publicly available sources. Since some of the occupation titles are associated with more than one SOC code, our classification problem is multi-labelled. Most of the records (74%) are short with less than four words leading to sparse feature vectors. Since all features in 98% of records have term frequency of one, the feature vector is binary. The data covers all SOC codes. However, the coverage decreases as we move down the hierarchy since the number of classes increases. Therefore, it is expected that accuracy will decrease in the lower levels. The data also has class imbalance, where the data is not divided equally between classes. Table 3.5 and 3.6 show a summary of the most important data attributes.

Table 3.5: Summary of data attributes

Number of Records	65,962
Number of Unigram features	11,251
Records with Single SOC Code	56,130 (85%)
Average Number of Unigrams per Record	3

Table 3.6: Summary of data coverage

	Number of Records per Class	
	Average	**Median**
First Level	3,110	2,250
Second Level	781	460
Third Level	173	98
Fourth Level	98	59

3.6 Evaluation Measures

The quality of automated occupation coding can be expressed by three measures [51]: *Production Rate*, *Accuracy*, and *Speed*. Production Rate represents the desired proportion of occupations to be coded automatically. For a given production rate, accuracy represents the proportion of occupations that were coded correctly (i.e., that agree with the human coder). Speed measures the time that is required to code one record. However, in this study, we are not focused on improving the speed of the classification. There is always a trade-off between production rate and accuracy. The higher the production rate, the lower the accuracy. Many researchers decide on a *cut-off* that determines the production rate. In other words, the cut-off determines the amount of data coded automatically, and the amount of data to be coded manually [18, 58]. The cut-off chooses the production rate that produces the most reasonable accuracy. At this point, a higher production rate leads to a notable fall in the accuracy. The accuracy of a classifier is determined based on how well it performs on the test data. We use *K-fold cross-validation* [34] for evaluating the classification models. Since our data has class imbalance, we use stratified sampling algorithm for multi-label data by Sechidis et al. [59] to create the k folds. The K-fold cross-validation steps are as follows.

1. Randomly divide the dataset into K folds
2. Use the first fold as the test set and the combination of the other $K-1$ folds as the training set
3. Calculate the accuracy on the test set
4. Repeat steps 2 and 3, K times using a different fold as the test set
5. Calculate the accuracy as the average of the accuracies of the k test sets

Accuracy is calculated with equation 3.1, where *True Positive* (TP_i) is the number of records that belong to class i, and are classified to class i; *True Negative* (TN_i)

is the number of records that do not belong to class i, and are not classified to class i either; *False Positive (FP_i)* is the number of records that do not belong to class i, but are incorrectly classified to class i; *False Negative (FN_i)* is the number of records that belong to class i, but are incorrectly not classified to class i. However, accuracy is not an appropriate measure where the test data are not evenly distributed across the classes. Since an algorithm that simply picks the majority class every time can easily achieve a high accuracy. As was discussed earlier, our data is not distributed equally among classes. Therefore, we need to use *Precision*, *Recall*, and *F1-score*, which are the three performance measures commonly used for evaluating classifiers with imbalanced test data [8]. Precision (P) is the fraction of records classified to class i that truly belong to class i. Recall (R) is the fraction of records in class i that are classified correctly to class i. There is a trade-off between recall and precision: the higher the precision, the lower the recall. Therefore, F1-score, which is a weighted harmonic mean of precision and recall, provides a composite measure for evaluating the classification. Equations 3.2 and 3.3 show precision and recall for class i.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

$$P_i = \frac{TP_i}{TP_i + FP_i} \tag{3.2}$$

$$R_i = \frac{TP_i}{TP_i + FN_i} \tag{3.3}$$

When multiple binary classifiers are used, *Macro-averaging* and *micro-averaging* are the two common methods for averaging the related precisions and recalls. Macro-average calculates the mean of the measures for binary classifiers with equal weights for all classes. Macro-average gives even weights to all classes. Therefore, macro F1 tends to over emphasize the smaller classes. Micro-average sums the corresponding TP_i, TN_i, FP_i, and FN_i to calculate an overall score. Micro-average gives equal

weights to each record in the test set. Therefore, micro F1 is dominated by the larger classes that have more records. In most research, micro F1 and macro F1 are both given to allow the classification to be fairly evaluated. Equation 3.4 and 3.5 show how macro F1 and micro F1 are calculated as described in [81].

$$MacroF1 = \frac{1}{|N|}\sum_{i=1}^{N}\frac{2P_iR_i}{P_i + R_i} \tag{3.4}$$

$$MicroF1 = 2\frac{PR}{P + R} \tag{3.5}$$

$$P = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N}(TP_i + FP_i)}$$

$$R = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N}(TP_i + FN_i)}$$

For multi-label classifications, hamming loss is often used in addition to precision, recall, and F1-score. Hamming loss evaluates how many times a label belonging to a record is not predicted or a label not belonging to a record is predicted [70]. The hamming loss value is always between zero and one, with hamming loss = 0 being the best performance. Equation 3.6 shows hamming loss where p is the number of records in the test set, q is the number of labels, and Δ is the symmetric difference between the predictions $h(x_i)$ and the true value y_i for each record.

$$HLoss = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{q}|h(x_i)\Delta y_i| \tag{3.6}$$

3.7 Statistical Analysis

We use statistical analysis to determine whether a significant difference exists between the performance of different models. Many statistical tests can be used for this purpose. These tests are divided into *parametric* and *non-parametric* tests based on the assumptions they make about the data. In the following subsections, two commonly used significance tests are described.

3.7.1 Significance Test

Parametric statistical tests have prior assumptions about the data and non-parametric statistical tests do not make specific assumptions about the data [75]. The *Student's T-test* is a parametric test that examines the null hypothesis that there is no statistically significant difference between the means of two samples. The *Paired T-test* compares the mean of two dependent groups (e.g., a comparison of two different treatments that are performed on the same subject). The student's t-test assumes that the samples are from a population that follows a normal distribution. As a result, we first need to check if our samples follow a normal distribution. If the data is not following a normal distribution, a non-parametric test should be used.

The *paired Wilcoxon signed rank test* is a non-parametric test that examines the null hypothesis that there is no statistically significant difference between the means of two dependent samples. It does not assume that the data is following a specific distribution. We use the normality test described below to test if our data comes from a normal distribution. Since we could not confidently prove normality, we use paired Wilcoxon signed rank test to compare the performance of the classification models.

3.7.2 Normality Test

Many statistical tests for normal distribution are available. The *Shapiro-Wilk normality test* [61] examines the null hypothesis that a sample follows a normal dis-

tribution. Razali and Yap suggest [50] to use Quantile-Quantile plots (Q-Q plots) in addition to the Shapiro-Wilk normality test. Therefore, we use Shapiro-Wilk normality test and Q-Q plots to determine normality of the data. The Q-Q plots, plot the data quantiles against a normal distribution quantiles. If the quantiles of the dataset and the normal distribution form a straight line, it means that the data is normally distributed.

Chapter 4

Classification with Classic Machine Learning Techniques

In this Chapter, we use Naïve Bayes, MaxEnt, and SVM classifiers for coding occupation titles to SOC. Figure 4.1 shows an overview of the classification scheme used in this chapter. Section 4.1 reviews the hyper-parameter tuning process for SVM and Maxent classifiers. Section 4.2 describes data pre-processing methods. Section 4.3 describes feature selection methods. Section 4.4 presents the flat classification model. Section 4.5 presents the hierarchical classification model. Finally, Section 4.6 presents the results and discussion.

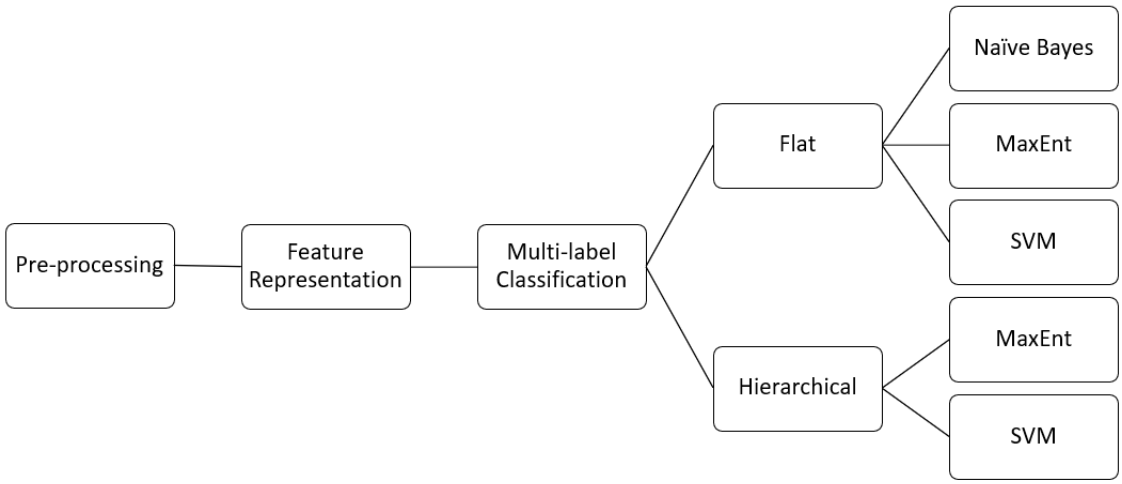

Figure 4.1: An overview of the classification scheme

4.1 Hyper-parameter Tuning

We use stratified sampling algorithm for multi-label data by Sechidis et al. [59] to create a validation set, which is used for tuning the hyper-parameters of the flat and hierarchical classification models. More specifically, 15% of all 840 classes in the fourth level of the SOC hierarchy are randomly sampled in our dataset, resulting in a validation set of 9,928 records. The higher level classes are embedded in the fourth level classes and can be inferred from the fourth level classes.

We use the validation set on the first level of the hierarchy to decide the best values for the hyper-parameters. Among the three classifiers, Naïve Bayes does not need parameter tuning. To choose a suitable value for SVM's penalty of the error term, we use grid search on a range of exponentially growing values of $C = (2^{-5}, 2^{-4}, ..., 2^4, 2^5)$ on the validation set. In the case of MaxEnt, we use grid search on the regularization parameter $C = (2^{-5}, 2^{-4}, ..., 2^4, 2^5)$ and the number of iterations $I = (10, 20, ..., 90, 100)$. The values that led to the best micro/macro F1 on the validation set are $C = 1$ for SVM and $C = 16$ and $I = 50$ for MaxEnt. The detailed results of the grid search are presented in Appendix A. To compare the performance, we evaluate the models using 10-fold cross-validation on the 56,034 remaining records of the dataset.

4.2 Data Pre-processing

We normalize the data by lowercasing the alphabetic letters, removing punctuation, and numbers. Stemming and stop words removal are known to have a positive effect on the classification of long text. However, they may not be as effective on short text. Therefore, we try to examine the effects of removing stop words and stemming on the first level of the SOC hierarchy with out dataset.

4.2.1 Removing Stop Words

We use the intersection of *NLTK*, *Sikit-Learn's*, *Terrier*, *snowball* [1] lists of stop words to create a stop word list. Since these words are shared among multiple sources, there is a higher chance that the useful content words are not removed in the process. The intersection of these four lists contains 107 stop words, among which 42 are detected in our dataset. The most frequent stop word is 'and', which is also the second most frequent word in the dataset after the word 'operator'. A summary of the stop words characteristics is shown in Table 4.1. The stop words are repeated 6,487 times and affected 6,312 records (10% of the data). In this 10%, the stop words appear with an average of 1.03 per record and the average record length is approximately five (the average record length for the entire data is approximately three). The longest record with stop words has 24 unigrams, and the shortest record has two unigrams. More detailed information about the stop words can be found in Appendix B.

Table 4.1: Summary of the characteristics of stop words in the dataset

Number of Records with Stop Words	6,312 (10% of entire data)
Frequency of Stop Words	6,487
Average Stop Words per Record in the 10%	1.03
Average Length of Records in the 10%	5

As discussed in Section 3.6, micro F1 is biased towards the popular classes and macro F1 is biased towards less popular classes. Therefore, we use both micro and macro F1. The effect of removing stop words on the micro/macro F1 is shown in Table 4.2. As can be seen, removing the stop words has no impact on the micro/macro F1 of the three classifiers. However, since stop words such as 'and' have high frequencies in the data we decide to remove the stop words.

[1] `http://snowball.tartarus.org/algorithms/english/stop.txt`, `https://bitbucket.org/kganes2/text-mining-resources/downloads/`, `https://www.nltk.org/`, `http://scikit-learn.org/stable/`

4.2.2 Stemming

We used the Porter Stemmer from NLTK package for stemming. Table 4.2 compares micro/macro F1 before and after stemming. Stemming the features slightly decreased the F1. In some cases one root replaced six or seven different words that have different meanings. For instance stem "anim" replace six words including "animator" and "animal", which are related to different job classes (Table 4.3). By taking into account that our data is very short, these issues can lead to over generalization and loss of information. Therefore, we decide not to use stemming.

Table 4.2: The effects of removing stop words and stemming

		NB	**MaxEnt**	**SVM**
None	Micro	0.72	0.74	0.75
	Macro	0.67	0.70	0.70
Stop words removal	Micro	0.72	0.74	0.75
	Macro	0.67	0.70	0.70
Stemming	Micro	0.71	0.74	0.74
	Macro	0.66	0.69	0.70

Table 4.3: Example of multiple words stemmed to the same root

words	**Stem**
animal	anim
animals	
animated	
animation	
animator	
anime	
author	author
authors	
authoring	
authority	
authorization	
authorizer	
authorizers	

4.3 Feature Selection

In this section, we test document frequency (DF) and Chi-square (X^2) feature selection methods on the first level of the SOC hierarchy.

4.3.1 Document Frequency

As discussed earlier, the data has 11,251 unigram features. In the first set of experiments, features with a minimum document frequency of N where $N = 1, 2, 3, ..., 10$ are used for the classification. Table 4.4 shows micro and macro F1 for the three algorithms after removing the features that have document frequency of less than N. All three classifiers achieve the highest score when all features are used ($N = 1$) and the micro/macro F1 of all three gradually decrease as more features are removed.

Table 4.4: Micro F1 (left) and Macro F1 (right) for Naïve Bayes (NB), MaxEnt, and SVM as we remove the features with document frequency less than $N = 1, 2, 3, ..., 10$

N	NB	MaxEnt	SVM
1	0.72	0.74	0.75
2	0.72	0.74	0.74
3	0.71	0.73	0.73
4	0.71	0.73	0.73
5	0.7	0.72	0.72
6	0.7	0.72	0.72
7	0.7	0.71	0.71
8	0.69	0.71	0.71
9	0.69	0.7	0.7
10	0.68	0.7	0.7

N	NB	MaxEnt	SVM
1	0.67	0.7	0.7
2	0.66	0.69	0.7
3	0.65	0.68	0.69
4	0.64	0.67	0.68
5	0.64	0.67	0.67
6	0.63	0.66	0.66
7	0.63	0.65	0.66
8	0.62	0.65	0.65
9	0.61	0.64	0.64
10	0.61	0.63	0.63

In the second set of experiments, N features with the highest document frequency are used for the classification. The results of the second experiment are shown in Table 4.5. As the number of features increases, the micro/macro F1 also increase. However, the highest score achieved by each classifier is the same score as in Table 4.2. Overall, DF does not improve the results.

4.3.2 Chi-Square

We also explore the use of X^2 feature selection. Table 4.6 shows micro and macro F1 after choosing the N features with the highest X^2 score. As is with DF, the performance of the classifiers increases as the number of features increases. However, the highest scores achieved by each classifier are the same as those in Table 4.2. As a result, X^2 does not improve the results. Overall, feature Selection does not improve the micro/macro F1. As mentioned in Section 2.2.3, feature selection on short text is often not as effective as on normal text. Therefore, we decide not to use feature selection in our experiments.

Table 4.5: Micro F1 (left) and Macro F1 (right) for Naïve Bayes (NB), MaxEnt, and SVM while using N best features based on document frequency

N	**NB**	**MaxEnt**	**SVM**
500	0.57	0.57	0.57
1,000	0.62	0.63	0.63
1,500	0.65	0.67	0.67
2,000	0.67	0.69	0.69
2,500	0.68	0.7	0.7
3,000	0.69	0.71	0.71
3,500	0.69	0.72	0.72
4,000	0.7	0.72	0.72
4,500	0.7	0.73	0.73
5,000	0.7	0.73	0.73
5,500	0.7	0.73	0.73
6,000	0.7	0.73	0.74
6,500	0.72	0.74	0.74
7,000	0.72	0.74	0.74
7,500	0.72	0.74	0.74
8,000	0.72	0.74	0.74
8,500	0.72	0.74	0.74
9,000	0.72	0.74	0.74
9,500	0.72	0.74	0.74
10,000	0.72	0.74	0.74
10,500	0.72	0.74	0.75
11,000	0.72	0.74	0.75
11,251	0.72	0.74	0.75

N	**NB**	**MaxEnt**	**SVM**
500	0.45	0.44	0.42
1,000	0.53	0.54	0.53
1,500	0.58	0.59	0.59
2,000	0.6	0.62	0.62
2,500	0.62	0.64	0.64
3,000	0.63	0.66	0.66
3,500	0.63	0.66	0.67
4,000	0.64	0.67	0.68
4,500	0.64	0.68	0.68
5,000	0.65	0.68	0.68
5,500	0.65	0.68	0.69
6,000	0.65	0.68	0.69
6,500	0.65	0.69	0.69
7,000	0.66	0.69	0.7
7,500	0.66	0.69	0.7
8,000	0.66	0.69	0.7
8,500	0.66	0.69	0.7
9,000	0.66	0.69	0.7
9,500	0.66	0.69	0.7
10,000	0.66	0.69	0.7
10,500	0.66	0.69	0.7
11,000	0.67	0.7	0.7
11,251	0.67	0.7	0.7

Table 4.6: Micro F1 (left) and Macro F1 (right) for Naïve Bayes (NB), MaxEnt, and SVM while using N best features based on X^2

N	**NB**	**MaxEnt**	**SVM**
500	0.56	0.57	0.56
1,000	0.62	0.66	0.66
1,500	0.67	0.68	0.68
2,000	0.68	0.7	0.7
2,500	0.69	0.71	0.71
3,000	0.69	0.72	0.72
3,500	0.69	0.72	0.72
4,000	0.69	0.72	0.72
4,500	0.69	0.73	0.73
5,000	0.7	0.73	0.73
5,500	0.7	0.73	0.74
6,000	0.7	0.73	0.74
6,500	0.72	0.74	0.74
7,000	0.72	0.74	0.74
7,500	0.72	0.74	0.74
8,000	0.72	0.74	0.74
8,500	0.72	0.74	0.74
9,000	0.72	0.74	0.74
9,500	0.72	0.74	0.74
10,000	0.72	0.74	0.74
10,500	0.72	0.74	0.75
11,000	0.72	0.74	0.75
11,251	0.72	0.74	0.75

N	**NB**	**MaxEnt**	**SVM**
500	0.53	0.53	0.53
1,000	0.59	0.61	0.60
1,500	0.62	0.63	0.63
2,000	0.63	0.65	0.65
2,500	0.64	0.66	0.66
3,000	0.64	0.67	0.67
3,500	0.65	0.67	0.68
4,000	0.65	0.67	0.68
4,500	0.65	0.68	0.69
5,000	0.66	0.68	0.69
5,500	0.66	0.68	0.69
6,000	0.66	0.68	0.69
6,500	0.66	0.69	0.70
7,000	0.66	0.69	0.70
7,500	0.66	0.69	0.70
8,000	0.66	0.69	0.70
8,500	0.66	0.69	0.70
9,000	0.66	0.69	0.70
9,500	0.66	0.69	0.70
10,000	0.66	0.69	0.70
10,500	0.66	0.69	0.70
11,000	0.67	0.70	0.70
11,251	0.67	0.70	0.70

4.4 Flat Classification Model

In this section, we use *Flat Multi-label* classification to code occupations to SOC. In multi-label classification (Section 2.2.4), a binary classifier is trained for each class to identify whether a job title belongs to that class or not. Since these binary classifiers are independent, multiple classes or no classes might be assigned to a record. If an occupation is not assigned a class, the recall decreases for the classes that it truly belongs to, hence decreasing the overall micro/macro F1. Production rate represents the percentage of the occupations that are assigned a class. There is a trade-off between Production rate and precision. The higher the production rate, the lower the precision. When the production rate is not 100%, a part of the data can be classified automatically, but the rest has to be classified manually. Since, we do not have resources to manually classify a portion of data, we propose a method that assigns at least one class to every occupation title.

The binary classifiers each output a confidence level. This confidence level is a probability value for Naïve Bayes and MaxEnt and distance from the hyper-plane for SVM. SVM uses a hyper-plane to separate positive examples from negative examples for a class. If a record is classified as negative by all classes, we simply choose the class for which the record has the least distance from the hyper-plane. For instance, 23 classifiers are used for the first level of the SOC hierarchy. If all 23 classifiers classify a record as negative, no code is normally assigned to that record. To increase the production rate, we instead pick the class for which the record is classified as negative, but has the least distance from the hyper-plane. MaxEnt and Naïve Bayes assign a class to a record if its probability for the class is higher than a threshold. Such a threshold is normally 50%, but can be lower if needed. Again when, no code is assigned to a record, we just select the class with the highest probability to maximize the production rate. We train a flat classifier with SVM, MaxEnt, and Naïve Bayes for the four levels of the SOC hierarchy and present the results in Section 4.6.

4.5 Hierarchical Classification Model

The SOC has a hierarchical structure. Therefore, a parent-child relationship exists between the four levels of the SOC hierarchy. However, flat classification does not use such parent-child information. Therefore, we explore using a multi-label hierarchical classification algorithm for coding occupations to SOC. We implement a binary classifier for each class in the hierarchy. The hierarchical approach assigns classes to a record if all the classes form a path from the first to the fourth level of the hierarchy. If no class is assigned to the record at a middle level, we roll back to the flat classification model to compute its class at the fourth level of the hierarchy, and then infer its ancestor classes along its path to the root. The reason is that in a hierarchical structure, a record in a lower level class also belongs to all of its ancestor classes. Since the classification is multi-labelled, more than one possible path can exist from the first level to the fourth level. As long as at least one class forms a path from the higher levels, the algorithm will continue to the next level of the hierarchy. Four scenarios are possible during the hierarchical classification.

1. The second level does not belong to the same path as the first level
2. The third level does not belong to the same path as the second level
3. The fourth level does not belong to the same path as the third level
4. All four levels form at least one path from the first to the fourth level

If at least one path exists for the majority of the records (fourth scenario), the hierarchical algorithm can improve the confidence of the decisions. However, if the other three scenarios happen for many records, the performance of the hierarchical algorithm might be the same or even worse than the flat models in some levels of the SOC hierarchy. Table 4.7 shows an example for each of the four scenarios. *Information technology analyst* is an example of the case where the second level does not belong to the same path as the first level. The first level predicts class 15, but the second level

predicts class 131. Therefore, the algorithm jumps to the fourth level and assigns the fourth level prediction (131111) and its ancestor classes to the record. In the case of *nuclear power plant engineer*, the first and second levels form a path (17, 172). However, the third level prediction does not belong to the same path (51801). Therefore, the fourth level prediction (172161) and its ancestor classes are assigned to the record. The first three levels form a path for *assembler refrigerator* (51, 512, 51202). However, the fourth level does not belong to the same path (519199). As a result, the fourth level prediction and its ancestor classes are assigned to the record. In the case of *funds development director*, all four classifiers agree on a path from the first level to the fourth level that is assigned to the record.

Table 4.7: Examples of the hierarchical classification process for four different scenarios

Occupation Title	**First Level**	**Second Level**	**Third Level**	**Fourth Level**	**Final Path**
information technology analyst	15	131	-	131111	(13, 131, 13111, 131111)
nuclear power plant engineer	17	172	51801	172161	(17, 172, 17216, 172161)
assembler refrigerator	51	512	51202	519199	(51, 519, 51919, 519199)
funds development director	11	112	11203	112031	(11, 112, 11203, 112031)

Algorithm 1 shows the hierarchical classification algorithm, which is implemented with MaxEnt and SVM. The input of the algorithm is the SOC codes tree S, the occupation titles X and their corresponding SOC codes Y. The SOC codes tree is a list containing each SOC code and its child codes, which can also be interpreted as each node and its adjacent nodes in a graph. X and Y are divided into training and testing sets: $X_{train}, X_{test}, Y_{train}, Y_{test}$. Furthermore, $Y_{train_1}, Y_{train_2}, Y_{train_3}, Y_{train_4}$ represent the first, second, third, fourth level SOC codes for the training set respectively. The same is true for Y_{test}. The algorithm outputs a list of paths from the first to fourth level for each of the records.

First, a classifier is trained on each level of the hierarchy (lines 4-7). Then, a for loop goes over the records in the test set. The classifier predicts the first level and second level codes for the record (lines 9-10). In line 11, we use a Depth First Search (DFS) to find the paths from the first level predicted codes to the second level

predicted codes. The DFS is explained in algorithm 2. For every path that exists, we move to the third level of the hierarchy. For every third level predicted code, we check for a path from the second level (lines 12-14). If a path exists, we move to the fourth level of the hierarchy. If at least one path exists from the third level predicted codes to the fourth level predicted codes, we turn on a flag (lines 15-19). If at any of the steps, the if statement is false, the algorithm checks the flag. If the flag is off (algorithm could not find at least one path), the fourth level predictions and their ancestors are chosen as the path (lines 20-22). We then add the *path* variable that contains all the paths from the first to fourth level for the record i to the output F. This process is repeated for every record i in the test set.

Algorithm 2 shows the Depth First Search [65] algorithm. The inputs of the algorithm are the SOC codes tree, a list of start nodes and a list of end nodes. The algorithm looks for paths from any of the start nodes to any of the end nodes. The output of the algorithm is a list of paths from the start nodes to the end nodes (if a path exists) and the list of end nodes that a path exists for them from the start nodes.

Starting from a Start node, the algorithm marks the current node as visited. Then, it explores the adjacent nodes that are not marked as visited. In lines 3 and 4, the algorithm loops over the start and end nodes. For every start node, the algorithm pushes the Start node and the visited nodes into the stack (line 5). In the beginning, the start and visited nodes are the same. As long as the stack is not empty, the pair containing the current node and the visited nodes are popped from the stack, and the adjacent nodes of the current node that are not visited are recorded in variable *unseen* (lines 6-8). The algorithm visits every node in *unseen*. If the End node is visited, the path from the Start node to the End node as well as the End node are recorded (lines 9-13). If the End node is not visited, the unvisited adjacent nodes of the current node are explored in the same manner until the End node is visited.

Algorithm 1 Hierarchical classification (finding all possible paths from the first to the fourth level of the SOC hierarchy for each record)

Input: S = {SOC codes tree}, X = {Occupation Titles}, Y = {Labels}
Output: F = {A list of paths from the first to the fourth level for each record}
$F = []$
Train_First_Level (X_{train}, Y_{train_1})
Train_Second_Level(X_{train}, Y_{train_2})
Train_Third_Level (X_{train}, Y_{train_3})
Train_Fourth_Level(X_{train}, Y_{train_4})
for $record \in X_{test}$ **do**
    Level_1 ← Predict_First_Level (*record*)
    Level_2 ← Predict_Second_Level (*record*)
    path, vertex_2 ← DFS (S, Level_1, Level_2)
    if vertex_2 is not empty **then**
        Level_3 ← Predict_Third_Level (*record*)
        path, vertex_3 ← DFS (S, vertex_2, Level_3)
        if vertex_3 is not empty **then**
            level_4 ← Predict_Fourth_Level (*record*)
            path, vertex_4 ← DFS (S, vertex_3, Level_4)
            if vertex_4 is not empty **then**
                Flag ← 1
    if Flag == 0 **then**
        Level_4 ← Predict_Fourth_Level (*record*)
        path ← DFS (S, Level_4/10000, Level_4)
    F.append(path)
return F

Algorithm 2 Depth First Search algorithm

```
Input: S = {SOC codes tree}, START = {start nodes}, END = {end nodes}
Output: F = {Paths from Start nodes to End nodes}, V = {The end nodes in F}
for start ∈ START do
    for end ∈ END do
        stack ← [(Start, Start)]
        while stack do
            (node, path) ← stack.pop()
            unseen ← S[node] - path
            for vertex ∈ unseen do
                if vertex == End then
                    F.append(path + [next])
                    V.append(vertex)
                    break
                else
                    stack.append((vertex, path + [vertex]))
return F, V
```

4.6 Results and Discussion

In this section, we present the results and discussions on the flat and hierarchical models with classic machine learning algorithms.

4.6.1 Experiments with the Flat Classification Model

Tables 4.8 to 4.11 present the results for the flat classification of the first to the fourth level of the SOC hierarchy respectively. The performance measures are micro and macro F1, precision, recall, production rate, and hamming loss. Numbers in **bold** are the best performance, and number with a * have no statistically significant difference from the best performance. Each classifier is first implemented with unigram features and then with unigram + bigram features. The results are compared with the same models with Full Production Rate (FPR). The FPR classifiers assign at least one class to every record. The normality test and Wilcoxon signed rank test details are shown in Appendix C.

Table 4.8: The results of the first level for Naïve Bayes (NB), SVM, and MaxEnt

Classifier	Micro-f1	Macro-f1	Precision	Recall	Production Rate	Hloss
NB-unigram	0.72	0.67	0.71	0.73	88%	0.03
NB-bigram	0.75	0.70	0.73	0.76	92%	0.02
NB-FPR-unigram	0.72	0.68	0.68	0.77	100%	0.03
NB-FPR-bigram	0.75	0.71	0.71	**0.79**	100%	0.02
SVM-unigram	0.75	0.70	0.82	0.69	83%	0.02
SVM-bigram	0.76	0.71	0.83	0.70	83%	0.02
SVM-FPR-unigram	0.76	0.73	0.76	0.76	100%	0.02
SVM-FPR-bigram	**0.77**	**0.74**	0.77	0.78	100%	0.02
MaxEnt-unigram	0.74	0.70	0.81	0.69	82%	0.02
MaxEnt-bigram	0.75	0.70	**0.84**	0.68	80%	0.02
MaxEnt-FPR-unigram	0.76	0.73	0.76	0.76	100%	0.02
MaxEnt-FPR-bigram	**0.77**	0.73	0.77	0.77	100%	0.02

Table 4.9: The results of the second level for Naïve Bayes (NB), SVM, and MaxEnt

Classifier	Micro-f1	Macro-f1	Precision	Recall	Production Rate	Hloss
NB-unigram	0.60	0.56	0.57	0.64	82%	0.01
NB-bigram	0.62	0.57	0.57	0.67	88%	0.01
NB-FPR-unigram	0.60	0.57	0.54	0.69	100%	0.01
NB-FPR-bigram	0.62	0.58	0.55	**0.70**	100%	0.01
SVM-unigram	0.63	0.59	0.75	0.55	71%	0.01
SVM-bigram	0.64	0.58	0.76	0.55	71%	0.01
SVM-FPR-unigram	0.66	**0.63**	0.66	0.66	100%	0.01
SVM-FPR-bigram	**0.67**	**0.63**	0.67	0.66	100%	0.01
MaxEnt-unigram	0.62	0.58	0.73	0.54	70%	0.01
MaxEnt-bigram	0.62	0.55	**0.77**	0.52	66%	0.01
MaxEnt-FPR-unigram	0.66	**0.63**	0.65	0.66	100%	0.01
MaxEnt-FPR-bigram	0.66	0.62	0.67	0.65	100%	0.01

Table 4.10: The results of the third level for Naïve Bayes (NB), SVM, and MaxEnt

Classifier	Micro-f1	Macro-f1	Precision	Recall	Production Rate	Hloss
NB-unigram	0.52	0.46	0.49	0.56	78%	0.003
NB-bigram	0.51	0.44	0.45	0.60	83%	0.003
NB-FPR-unigram	0.52	0.48	0.46	0.61	100%	0.003
NB-FPR-bigram	0.51	0.45	0.43	**0.64**	100%	0.003
SVM-unigram	0.56	0.48	0.71	0.47	64%	0.002
SVM-bigram	0.55	0.45	0.71	0.45	62%	0.002
SVM-FPR-unigram	**0.59**	**0.53**	0.60	0.58	100%	0.002
SVM-FPR-bigram	**0.59**	0.52	0.60	0.58	100%	0.002
MaxEnt-unigram	0.55	0.46	0.69	0.46	62%	0.002
MaxEnt-bigram	0.52	0.41	**0.72**	0.41	55%	0.002
MaxEnt-FPR-unigram	**0.59**	0.52	0.59	0.58	100%	0.002
MaxEnt-FPR-bigram	0.58*	0.51	0.60	0.57	100%	0.002

Table 4.11: The results of the fourth level for Naïve Bayes (NB), SVM, and MaxEnt

Classifier	Micro-f1	Macro-f1	Precision	Recall	Production Rate	Hloss
NB-unigram	0.49	0.43	0.46	0.53	76%	0.002
NB-bigram	0.46	0.39	0.39	0.56	81%	0.002
NB-FPR-unigram	0.49	0.44	0.43	0.58	100%	0.002
NB-FPR-bigram	0.46	0.41	0.38	**0.60**	100%	0.002
SVM-unigram	0.53	0.43	0.69	0.43	61%	0.001
SVM-bigram	0.51	0.40	0.68	0.41	58%	0.001
SVM-FPR-unigram	**0.56**	**0.49**	0.57	0.55	100%	0.001
SVM-FPR-bigram	0.55*	0.48	0.57	0.54	100%	0.001
MaxEnt-unigram	0.52	0.41	0.68	0.42	58%	0.001
MaxEnt-bigram	0.48	0.35	**0.70**	0.36	50%	0.001
MaxEnt-FPR-unigram	0.55*	0.48	0.56	0.55	100%	0.001
MaxEnt-FPR-bigram	0.55*	0.47	0.56	0.53	100%	0.001

- SVM and MaxEnt performed similarly, with SVM having a slightly higher recall.
- The overall performance of Naïve Bayes was poor in comparison with the other two classifiers. Therefore, hierarchical classification was not implemented with Naïve Bayes. One of the reasons for this poor performance may be the feature dependency in the data.
- Naïve Bayes had the highest recall and MaxEnt had the highest precision at all levels of the hierarchy.
- Recall is notably improved in FPR classifiers. However, this improvement comes at the cost of loosing precision. Increasing the production rate to 100%, improved the overall performance of SVM and MaxEnt since the increase in recall was more notable than the decrease in precision.
- If macro F1 is notably lower than micro F1, it means that the performance of the less frequent classes are significantly lower than the more frequent classes. As we move down the hierarchy, the difference between micro F1 and macro F1 increases. In the first level, the maximum difference is 5%, while in the fourth level, the maximum difference is 13%. The performance of the classification generally decreases in the lower levels since the data per class decreases,

and the increase in the difference of micro F1 and macro F1 indicates that the performance of less frequent classes decreases more significantly than more frequent classes.

- Naïve Bayes performs better with unigram + bigram features in the first and second level of the SOC hierarchy. However, it performs better with unigram features in the third and fourth level of the SOC hierarchy. We cannot confidently draw any conclusions on the effect of the type of features for SVM and MaxEnt since the effect of the features differ based on levels and classifiers.
- Hamming loss is the fraction of incorrectly classified classes to the total number of classes. Therefore, it is natural that hamming loss decreases as we move down the hierarchy.
- Table 4.12 shows the running time for each classifier measured by the sum of the running times for all four levels.

Table 4.12: Running time of the flat models

Classifier	**Time (minute)**
NB-unigram	3.1
NB-bigram	4.8
NB-FPR-unigram	3.6
NB-FPR-bigram	6.1
SVM-unigram	11.2
SVM-bigram	18.2
SVM-FPR-unigram	11.3
SVM-FPR-bigram	19.8
MaxEnt-unigram	36
MaxEnt-bigram	94
MaxEnt-FPR-unigram	37
MaxEnt-FPR-bigram	95

As shown in Table 4.13, the frequencies of the classes are not in the same range. Therefore, we look into the performance of the individual classes in the first level of

the hierarchy with SVM-FPR-bigram. Class 51 (production occupations), which has the highest frequency, has the highest recall (0.91) and the second highest F1-score (0.85). Class 25 (education, training, and library occupations), which has a relatively high frequency, has the highest F1-score (0.9) and precision (0.9). Class 45 (farming, fishing, and forestry occupations) with an average of 136 records in the training data, has the lowest F1-score (0.63) and recall (0.56). The lowest precision (0.63) belongs to class 13 (business and financial operations occupations), which has an average frequency of 177 records in the training data.

Table 4.13: Frequency, F1-score, precision, and recall of the 23 classes in the first level of SOC hierarchy implemented with SVM-FPR-bigram

Class	**Frequency**	**F1-score**	**Precision**	**Recall**
11	325	0.68	0.66	0.71
13	177	0.64	0.65	0.63
15	107	0.74	0.75	0.73
17	210	0.75	0.75	0.74
19	177	0.68	0.74	0.63
21	72	0.7	0.72	0.69
23	36	0.77	0.81	0.73
25	273	**0.9**	**0.9**	0.9
27	284	0.81	0.83	0.78
29	227	0.82	0.84	0.8
31	54	0.66	0.71	0.61
33	98	0.74	0.75	0.73
35	60	0.75	0.79	0.72
37	53	0.7	0.77	0.64
39	123	0.67	0.73	0.62
41	162	0.77	0.81	0.74
43	329	0.69	0.71	0.68
45	136	0.63	0.71	0.56
47	297	0.72	0.77	0.67
49	486	0.83	0.82	0.84
51	**1556**	0.85	0.79	**0.91**
53	389	0.67	0.71	0.64
55	100	0.8	0.81	0.79

Next, we divide the data into *high frequency* and *low frequency* groups and analyze the performance within these two groups. If the class has less than 100 samples, it is categorized as low frequency and if the class has frequency of equal or greater than 100 samples, it is categorized as high frequency. Table 4.14 shows the average performance of the two groups in the first level of the hierarchy. As expected, the high frequency group performed better than the low frequency group.

Table 4.14: Micro and macro F1 of SVM-FPR-bigram for the high frequency and low frequency groups in the first level of the SOC hierarchy

Group	**Micro F1**	**Macro F1**
High Frequency	0.78	0.74
Low Frequency	0.72	0.72

4.6.2 Experiments with the Hierarchical Classification Model

Table 4.15 to 4.18 provides the results of the hierarchical classification for each level of the SOC hierarchy and Table 4.19 shows the running time of the hierarchical classifiers. We did not implement hierarchical Naïve Bayes since flat Naïve Bayes performed poorly. Hierarchical classification (hier) is implemented for MaxEnt and SVM using unigram features and unigram + bigram features. The hierarchical classifiers assign at least one SOC code to each record.

Table 4.15: The results of the first level for SVM and MaxEnt

Classifier	**Micro f1**	**Macro f1**	**Precision**	**Recall**	**Production Rate**	**HLoss**
SVM-hier-unigram	**0.76**	**0.73**	**0.78**	**0.75**	100%	0.02
SVM-hier-bigram	**0.76**	0.72*	0.77	**0.75**	100%	0.02
MaxEnt-hier-unigram	**0.76**	**0.73**	**0.78**	**0.75**	100%	0.02
MaxEnt-hier-bigram	**0.76**	0.72	0.77	0.74	100%	0.02

Table 4.16: The results of the second level for SVM and MaxEnt

Classifier	Micro-f1	Macro-f1	Precision	Recall	Production Rate	HLoss
SVM-hier-unigram	**0.66**	**0.63**	**0.68**	**0.64**	100%	0.01
SVM-hier-bigram	0.65	0.62	0.67	**0.64**	100%	0.01
MaxEnt-hier-unigram	**0.66**	**0.63**	**0.68**	**0.64**	100%	0.01
MaxEnt-hier-bigram	0.65	0.62	0.67	0.63	100%	0.01

Table 4.17: The results of the third level for SVM and MaxEnt

Classifier	Micro-f1	Macro-f1	Precision	Recall	Production Rate	HLoss
SVM-hier-unigram	**0.59**	**0.53**	**0.62**	**0.57**	100%	0.002
SVM-hier-bigram	**0.59**	0.52	0.61	**0.57**	100%	0.002
MaxEnt-hier-unigram	**0.59**	**0.53**	0.61	**0.57**	100%	0.002
MaxEnt-hier-bigram	0.58	0.51	0.61	0.56	100%	0.002

Table 4.18: The results of the fourth level for SVM and MaxEnt

Classifier	Micro-f1	Macro-f1	Precision	Recall	Production Rate	HLoss
SVM-hier-unigram	**0.56**	**0.49**	**0.59**	**0.54**	100%	0.001
SVM-hier-bigram	0.55	0.48	0.58	0.53	100%	0.001
MaxEnt-hier-unigram	**0.56**	**0.49**	0.58	**0.54**	100%	0.001
MaxEnt-hier-bigram	0.55	0.47	0.57	0.53	100%	0.001

Table 4.19: The running time of hierarchical classifiers

Classifier	Time (minute)
SVM-hier-unigram	57
SVM-hier-bigram	61.8
MaxEnt-hier-unigram	88.8
MaxEnt-hier-bigram	150

- SVM and MaxEnt performed similarly in all levels of the hierarchy
- We cannot draw any conclusions on the effect of the type of features (unigram or unigram + bigram) since the effect of the features differ based on levels and classifiers.
- Macro F1 is less than micro F1 in all levels of the hierarchy, indicating that the algorithm is weaker on classifying less frequent classes. As we move down

the hierarchy, the difference between micro F1 and macro F1 increases. In the first level, the maximum difference is 4% and in the fourth level, the maximum difference is 8%. The algorithm becomes weaker on classifying low frequency labels in the lower levels of the hierarchy.

4.6.3 Comparison of the Flat and Hierarchical Models

To compare flat and hierarchical classification, we compare SVM-FPR-unigram, SVM-FPR-bigram, MaxEnt-FPR-unigram, and MaxEnt-FPR-bigram with SVM-hier-unigram, SVM-hier-bigram, MaxEnt-hier-unigram, and MaxEnt-hier-bigram respectively.

- Flat and hierarchical models performed similarly in the third and fourth levels of the hierarchy. The hierarchical model has a lower micro F1 in the first and second levels of the hierarchy. In the hierarchical models, levels affect the performance of other levels. Since lower levels have a lower performance, they can have a negative effect on the higher levels.
- Overall, the flat models have higher recall and the hierarchical models have higher precision.
- The running time of the hierarchical models are notably higher than flat models. Therefore, the running time of the hierarchical models is an important factor in deciding between flat and hierarchical classification.

In the hierarchical model, all four levels need to form at least one path to classify the record based on the hierarchy. On average this case happens for 55% of the records with SVM (unigram features). If all four levels form a path, the model uses the flat classification in the fourth level of the hierarchy. On average this case happens for 45% of the records with SVM. Since just slightly more than half of the records take advantage of the hierarchy, we see a small difference in performance of the flat

and hierarchical models. Table 4.20 shows the average number and percentage of remaining records in each of the four levels of the hierarchical classification (in each level, some records are dropped since child and parent nodes do not form a path).

Table 4.20: Average number of the remaining records in the four levels of the hierarchical classification with SVM-hier-unigram

Level	**Number of records**	**Percentage of records**
First	5,405	100%
Second	3,775	70%
Third	3,070	57%
Fourth	3,068	55%

We put aside the 45% of records that were not classified hierarchically and analyze the difference of performance of the hierarchical and the flat model on the other 55%. Tables 4.21 to 4.24 show the results of the flat and hierarchical models on the 55% of the records that were classified based on the hierarchical information. On these records, macro F1 and micro F1 of the hierarchical models are 1% to 2% and precision is 4% to 5% higher than the flat models. The recall of the flat models are 1% to 2% higher than the hierarchical modes.

Table 4.21: The performance of flat and hierarchical models on the 55% of the records that were classified with SVM based on the hierarchical information (first level)

Level	**Classifier**	**Micro F1**	**Macro F1**	**Precision**	**Recall**
First	SVM-hier-unigram	**0.90**	**0.88**	**0.91**	0.89
First	SVM-flat-unigram	0.89	0.87	0.87	**0.90**

Table 4.22: The performance of flat and hierarchical models on the 55% of the records that were classified with SVM based on the hierarchical information (second level)

Level	**Classifier**	**Micro F1**	**Macro F1**	**Precision**	**Recall**
Second	SVM-hier-unigram	**0.83**	**0.80**	**0.85**	0.82
Second	SVM-flat-unigram	0.82	0.78	0.80	**0.84**

Table 4.23: The performance of flat and hierarchical models on the 55% of the records that were classified with SVM based on the hierarchical information (third level)

Level	Classifier	Micro F1	Macro F1	Precision	Recall
Third	SVM-hier-unigram	**0.78**	**0.64**	**0.79**	0.77
Third	SVM-flat-unigram	0.77	0.63	0.74	**0.79**

Table 4.24: The performance of flat and hierarchical models on the 55% of the records that were classified with SVM based on the hierarchical information (fourth level)

Level	Classifier	Micro F1	Macro F1	Precision	Recall
Fourth	SVM-hier-unigram	**0.74**	**0.57**	**0.75**	0.73
Fourth	SVm-flat-unigram	0.73	0.56	0.71	**0.75**

In the case of MaxEnt, on average, 53% of records are classified using the parent child information. Table 4.25 shows the average number and percentage of remaining records in each of the four levels of the hierarchical classification with MaxEnt-hier-unigram. We put aside the 47% that were not classified based on the hierarchical information and analyze the performance of the hierarchical and the flat classification for the remaining 53% (Tables 4.26 to 4.29).

Table 4.25: Average number of remaining records in the four levels of the hierarchical classification with MaxEnt-hier-unigram.

Level	Number of records	Percentage of records
First	5,405	100%
Second	3,716	69%
Third	2,996	55%
Fourth	2,885	53%

Table 4.26: The performance of flat and hierarchical models on the 53% of the records that were classified with MaxEnt based on the hierarchical information (first level)

Level	Classifier	Micro F1	Macro F1	Precision	Recall
First	MaxEnt-hier-unigram	**0.90**	**0.88**	**0.91**	0.90
First	MaxEnt-flat-unigram	0.89	0.86	0.87	**0.91**

Table 4.27: The performance of flat and hierarchical models on the 53% of the records that were classified with MaxEnt based on the hierarchical information (second level)

Level	Classifier	Micro F1	Macro F1	Precision	Recall
Second	MaxEnt-hier-unigram	**0.84**	**0.80**	**0.84**	0.83
Second	MaxEnt-flat-unigram	0.82	0.78	0.78	**0.85**

Table 4.28: The performance of flat and hierarchical models on the 53% of the records that were classified with MaxEnt based on the hierarchical information (third level)

Level	Classifier	Micro F1	Macro F1	Precision	Recall
Third	MaxEnt-hier-unigram	**0.78**	**0.63**	**0.78**	0.78
Third	MaxEnt-flat-unigram	0.76	0.61	0.72	**0.80**

Table 4.29: The performance of flat and hierarchical models on the 53% of the records that were classified with MaxEnt based on the hierarchical information (fourth level)

Level	Classifier	Micro F1	Macro F1	Precision	Recall
Fourth	MaxEnt-hier-unigram	**0.74**	**0.55**	**0.74**	0.74
Fourth	MaxEnt-flat-unigram	0.72	**0.55**	0.69	**0.76**

For both SVM and MaxEnt classifiers, the overall performance of the hierarchical model was slightly better than the flat model (1% to 2%). The hierarchical model had 4% to 6% higher precision and 1% to 2% lower recall. Considering the higher running time of the hierarchical model, using the hierarchical model instead of the flat model is only reasonable in cases where precision is very important.

Our decision to use the fourth level predictions to modify the ancestor classes was based on the assumption that this scenario happens for a few records. However, analysis of the results indicate that in the hierarchical approach, almost half of the records are classified using the flat classification in the fourth level of the hierarchy. Therefore, the performance of the flat and hierarchical models are not considerably different. As a future work, the hierarchical approach can be implemented by a fully top down process.

Chapter 5

Classification with Deep Learning Techniques

In this chapter, we describe a set of experiments on using Convolutional Neural Networks (CNNs) for coding occupation titles to SOC. We compare a CNN trained on randomly initialized word vectors with a CNN trained on pre-trained word vectors. In Section 5.1, we describe the CNN architecture. In Section 5.2, we use 5-fold cross-validation to tune the CNN hyper-parameters. In Section 5.3, we present the results and discussion.

5.1 CNN Architecture

We use CNN for coding occupation titles to the first level of the SOC hierarchy. We implemented a wide CNN that has one embedding, one convolution, one max-pooling, and one fully connected layer, which is similar to [31]. This simple CNN architecture has shown good performance on text classification. Furthermore, we compared the performance of our CNN with deeper CNN architectures. The deeper models did not outperform our CNN model although the deeper models are more complex and have higher training time. Figure 5.1 shows the architecture of the CNN model. In the embedding layer, each batch of data has 50 records, and the maximum record length is 24 words. The data is represented in a 50×24 matrix. Each of the words of the records are replaced with a dense vector of size 300, resulting in a $50 \times 24 \times 300$ matrix.

In the convolution layer, 100 filters of sizes of one, two, and three (300 overall) are used. The elementwise multiplication of each of the filters with the vector repre-

sentation of a record creates a feature map. For instance, given the 'chief executive officer' job title, the elementwise multiplication of the filter with size 2 with each bigram in the job title leads to a feature map with 25 scalar values. Since we have 300 filters, we have 300 feature maps with different sizes.

Next, the resulting 300 feature maps are fed into the pooling layer. The pooling layer extracts the maximum value of each of the feature maps. Then the pooled features are concatenated to create a single feature map. Some of the features are randomly dropped out (set to zero) to avoid overfitting as suggested in [62]. Finally, the remaining features are fed into the fully connected (FC) layer that has a Sigmoid function, which assigns an independent probability to each class. This process is repeated for every batch.

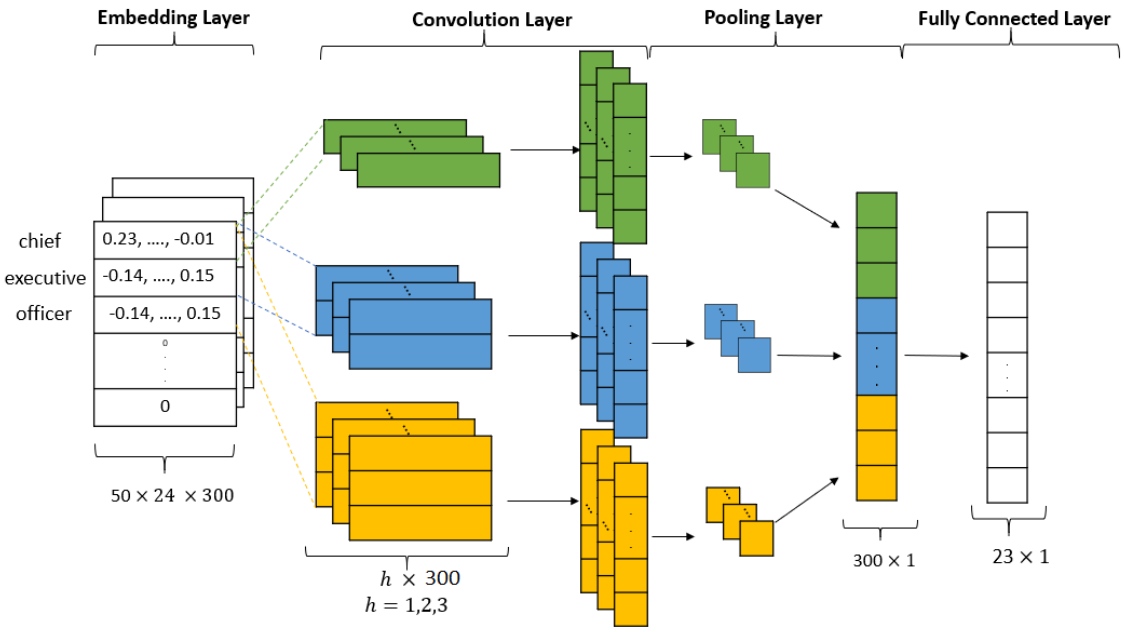

Figure 5.1: The CNN architecture

We compare CNN-Random that learns word vectors during the classification process in a supervised manner with CNN-Word2vec that uses word vectors that were pre-trained in an unsupervised manner on a large corpus.

5.1.1 CNN-Random

In the CNN-Random, the word embeddings are initialized with a 300 dimensional real valued dense vector and are updated during the training process. The values of the vector are initialized in range [-0.25,0.25] and are randomly sampled from a uniform distribution.

5.1.2 CNN-Word2vec

In the CNN-Word2vec, the words are initialized with word2vec vectors that were pre-trained by Mikolov et al. [42] on the Google News Word2vec dataset [1]. The Google News dataset has 3 million 300-dimension English word vectors. From the 11,251 words in the SOC labeled data, 9,288 (83%) words were found in the Google News dataset. The words in the text are replaced with their embedding by a look-up table. If the embedding for a word is not available, the vector for the word is randomly initialized in range [-0.25,0.25]. We also compare CNN-Word2vec-static that does not update word vectors during the training with CNN-Word2vec that updates word vectors during the training.

5.2 Hyper-parameter Tuning

The performance of the CNN is highly dependent on its hyper-parameters. These are some of the important CNN hyper-parameters:

1. **Learning rate** decides how much to update the weights during the optimization process.

2. **Number of iterations** is the number of times the optimization algorithm goes over the batch size.

[1] https://code.google.com/archive/p/word2vec/

3. **Number of epochs** is the number of times the entire training data is fed into the Neural Network.

4. **Batch size** is the number of training examples used in each iteration. In the *full-batch mode*, the entire dataset is used in an iteration (in this case, epoch and iteration are the same). In the *Mini-batch mode*, the size of the training data in one iteration is less than the entire data and more than one. The weights are updated after each batch.

5. **Activation function** adds non-linearity to the CNN.

6. **Regularization** is used to avoid overfitting. Two common regularization techniques are *drop out* and *L2 regularization*.

7. **Number of layers**

8. **Number of filters**

9. **Size of filter**

10. **Pooling strategy**

11. **Stride size** is how much we want to move the filter across the text in each step. Most researchers use stride size of one. Therefore, filters have overlaps.

Although Grid Search is often used for machine learning algorithms, the high number of hyper-parameters in deep learning models and the combinations of these hyper-parameters can grow exponentially. Therefore, *Random Search* has been more successful with deep learning models [5]. Random Search samples a specified number of values for the parameters based on a predefined distribution.

Since CNN has many hyper-parameters, tuning the hyper-parameters is time consuming. In case of hyper-parameters that greatly depend on the data, we use 5-fold cross-validation to find the optimum value. We use random search to tune the value of learning rate in range [0.0001, 1]; filter sizes of [{1}, {2}, {3}, {1,2}, {2,3}, {1,2,3},

{2,3,4}, {1,2,3,4}], drop out probability in range [0.1, 0.5]; and L2 regularization parameter (λ) values in range $[2^{-5}, 2^{5}]$. For the rest of the hyper-parameters, we choose the value based on the data characteristics and what is recommended in the literature. These hyper-parameters with a default value are stride size = 1, batch size = 50, Rectified Linear Units (ReLu) activation function, max-pooling, and 100 filters for each filter size as suggested in [31, 82]. Table 5.1 summarizes the default hyper-parameters and Table 5.2 shows the hyper-parameters that are tuned using 5-fold cross-validation.

Hyper-parameter	**Value**
Activation Function	ReLu
Mini Batch Size	50
Stride Size	1
Pooling	Max
Number of Filters	100 per filter size

Table 5.1: The default hyper-parameter values for CNN

Hyper-parameter	**Value**
Learning Rate	[0.0001, 1]
Filter size	[{1}, {2}, {3}, {1,2}, {2,3}, {1,2,3}, {2,3,4}, {1,2,3,4}]
L2	$[2^{-5}, 2^{5}]$
Dropout Rate	[0.1,0.5]

Table 5.2: The ranges for hyper-parameter tuning of CNN

We use the stratified sampling algorithm for multi-label data by Sechidis et al. [59] to divide the data into testing and training set. 20% of each of the fourth level labels are randomly selected for the testing set (13,222 records). The training set includes the remaining 52,740 records. We use 5-fold cross-validation on the training set to tune the hyper-parameters. Then we test the CNN with the best hyper-parameters on the testing set. Since the deep learning models have high training time and multiple hyper-parameters, we use 5-fold instead of 10-fold cross-validation.

Algorithm 3 shows the random search used for tuning the hyper-parameters.

Algorithm 3 Random Search Algorithm

Output: results = {n sets of parameters and their corresponding classification score}

filter = $[\{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}, \{2, 3, 4\}, \{1, 2, 3, 4\}]$

results = []

for $i = 1$ to n **do**

dropout ← random_number(0.1, 0.5)

LR ← random_number(0.0001, 1)

λ ← random_number (0.03, 32)

j ← random_number (0,7)

results[i] ← CNN.py p1=dropout p2=LR p3=λ p4=filter[j]

return results

The hyper-parameter values leading to the highest micro/macro F1 for CNN-Random were Learning Rate = 0.001, Filter Size = [1,2,3] , Dropout Rate = 0.3, and $\lambda = 0.03$ (Table 5.3). The hyper-parameter values leading to the highest micro/macro F1 for CNN-Word2vec were Learning Rate = 0.001 , Dropout Rate = 0.5 , $\lambda = 0.03$, and Filter Size = [1,2,3] (Table 5.4).

Hyper-parameter	**Value**
Activation Function	ReLu
Mini Batch Size	50
Stride Size	1
Pooling	Max
Epochs	100
Embedding Initialization	[-0.25,0.25]
Embedding Dimension	300
Learning Rate	0.001
Optimizer	Adam
Number of Filters	100
Filter Size	[1,2,3]
λ	0.03
Dropout Rate	0.3

Table 5.3: The final hyper-parameters used in CNN-Random

Hyper-parameter	**Value**
Activation Function	ReLu
Mini Batch Size	50
Stride Size	1
Pooling	Max
Epochs	100
Embedding Initialization	word2vec
Embedding Dimension	300
Learning Rate	0.001
Optimizer	Adam
Number of Filters	100
Filter Size	[1,2,3]
λ	0.03
Dropout Rate	0.5

Table 5.4: The final hyper-parameters used in CNN-Word2vec

5.3 Results and Discussion

Table 5.5 shows the performance of CNN classifiers.

Table 5.5: The results of first level flat classification with CNN

Classifier	Micro F1	Macro F1	Precision	Recall	Production Rate	HLoss
CNN-Random	0.75	0.70	0.82	0.70	86	0.02
CNN-Random-FPR	0.77	0.73	0.77	0.76	100	0.02
CNN-Word2vec	0.75	0.71	0.83	0.69	85	0.02
CNN-Word2vec-FPR	0.77	0.73	0.78	0.75	100	0.02
CNN-Word2vec-static	0.54	0.28	0.83	0.40	51	0.03
CNN-Word2vec-static-FPR	0.61	0.43	0.64	0.59	100	0.04

- The CNN models with static word vectors performed poorly in comparison with the other models, showing that the training of word vectors with the CNN is effective.

- Pre-trained word vectors are known to improve the performance of CNN. Despite what we expected, CNNs with random word vectors performed similarly to CNNs with pre-trained Word2vec word vectors. We believe one reason is that only 83% of the words in the data are found in the Word2vec database and the remaining 13% are replaced with random word vectors. Some of the words that are not found are abbreviations.

- CNN did not outperform SVM and MaxEnt although it is a more complex classifier.

- Regardless of the classifier the micro F1 does not exceed 0.77. We believe the attributes of the data has made the classification challenging. The data is extremely short with an average of 3 words per record. Classifying short text is more difficult than long text. Moreover, common feature selection and feature weighting methods do not work well on short text. Another issue is the quantity of data. The performance of the first level classification, which has an average of 3,110 records per class is better than the fourth level, which has

an average of 98 records per class. If we look at previous studies on job title classification (Table 2.3) that used machine learning algorithms, the accuracy results range between 44.5% to 76.3%. Our classification results are similar to these numbers.

5.3.1 Running time

Table 5.6 shows the running time of the CNN models. The running time is for the first level of the SOC hierarchy. Deep learning models typically have higher running time than classic machine learning algorithms. Time constraints was one of the reasons we did not implement the deep learning models for the lower levels of the hierarchy. The implementation of the deep learning models for the lower levels of the hierarchy was left as part of the future work.

We use TensorFlow [2] software library with Sharcnet clusters[2] to run our CNN codes on GPUs and take advantage of their parallel programming capabilities. CNN was run on Sharcnet's Graham cluster with 3 CPUs and 2 GPUs. Furthermore, to tune the hyper-parameters, we ran two serial farms (one for CNN-random and one for CNN-Word2vec) using META package on Sharcnet's Graham cluster. Each serial farm tests 1,000 combinations of random values in the selected range for the four hyper-parameters. As a result, each serial farm ran 100 jobs and each job tested 10 hyper-parameter combinations. The jobs used 3 CPUs, 2 GPUs, and had an average running time of 12 hours.

[2]https://www.sharcnet.ca

Table 5.6: Running time of CNN classifiers

Classifier	**Time (hour)**
CNN-Random	18.5
CNN-Random-FPR	18.6
CNN-Word2vec	18.9
CNN-Word2vec-FPR	18.92
CNN-Word2vec-static	6.9
CNN-Word2vec-static-FPR	6.9

Chapter 6

Conclusions and Future Work

In this thesis, we use supervised machine learning algorithms to code occupation titles to SOC. We compare classic machine learning algorithms such as Naïve Bayes, SVM, and MaxEnt and deep learning algorithms such as CNN on occupation coding. We use the Scikit-learn library to implement the models with classic machine learning algorithms. Furthermore, we use the TensorFlow library and Sharcnet clusters to use serial farming and GPUs to implement and tune the hyper-parameters of the CNN models.

To train the classifiers, we collected 65,962 job titles labeled with SOC codes from publicly available online sources. Since some of the occupation titles were labeled with more than one SOC code, we use multi-label classification. SOC has a hierarchical structure with four levels. Therefore, we implement a hierarchical classification model and compare it to a flat classification model at each level of the SOC hierarchy.

We implemented a flat classifier for each level of the SOC hierarchy with Naïve Bayes, SVM, and MaxEnt. We report that SVM and MaxEnt performed similarly on all levels of the hierarchy with SVM having a slightly higher recall. Naïve Bayes performed poorly in comparison with the other two classifiers. Therefore, the hierarchical model is only implemented with SVM and MaxEnt. Flat and hierarchical models performed similarly, with the flat model having a higher recall and the hierarchical model having a higher precision. The hierarchical model has a longer running time as well. Hence, either of the models can be used depending on the importance of running time, precision, and recall. The models above are implemented once with unigram features and once with unigram + bigram features.

We transform the multi-label classification into N binary classifications where N is the number of classes. Since N independent binary classifiers are used, multiple classes or none might be assigned to a record. We propose assigning the class with the highest probability in cases where no class is assigned (none of the class probabilities are high enough) or in case of SVM assigning the class with the least distance from the hyper-plane. This model significantly increases the recall and slightly decreases the precision.

We also implemented a CNN with random word vectors, a CNN with static Word2vec word vectors, and a CNN with non-static Word2vec word vectors on the first level of the SOC hierarchy. CNN with static Word2vec word vectors performed poorly in comparison with the other two classifiers, which shows that the training of the word vectors with the CNN model was effective. Although Word2vec word vectors are known to improve the classification, the CNN with random word vectors performed similarly to the CNN with non-static Word2vec word vectors. We believe one of the reasons is that 83% of the word in the data are found in the Word2vec database and the remaining 13% is replaced with random word vectors.

The performance of SVM, Maximum Entropy, and CNN was similar even though they have different approaches to classification. Regardless of the model, micro F1 did not exceed 0.77 in the first level, 0.67 in the second level, 0.59 in the third level, and 0.56 in the fourth level of the SOC hierarchy. We believe that the attributes of data made the classification challenging. Job titles are extremely short with an average of three words per record. Furthermore, there is not many repetitive words in a job title. Therefore, common feature selection and feature weighting methods are not effective in the classification. There is a need to develop techniques that are effective on classification of short text. The higher levels of the hierarchy, which have more records per class are classified more accurately in comparison with lower levels of the hierarchy. Furthermore, the more frequent classes had a higher classification score on average. Therefore, we believe more labeled data can improve the classification to

some extent.

6.1 Future Work

As part of the future work, we are interested in implementing a CNN classifier for the lower levels of the SOC hierarchy to see if it can outperform the classic machine learning algorithms in the lower levels of the hierarchy. We are also interested in comparing the performance of our models to ensemble algorithms, a multi-channel CNN, and a CNN trained on top of GloVe word vectors.

We believe designing a method to confidently expand the abbreviations can improve the quality of automatic occupation coding. There are many abbreviations used in the data. For instance, 'A.S.A.T. C.O.R.E. counselor' is one of the occupation titles in the data. After the pre-processing, the job title is recorded as 'a s a t c o r e counselor'. The single letters are removed as part of the tokenization. Therefore, the final job title that is fed to the classifier is 'counselor'. This is particularly problematic for the fourth level of the hierarchy where counselors have different SOC codes based on their field of expertise. As a result, by expanding the abbreviations, we can avoid loss of information and improve the classification.

Analysis of the results of the hierarchical classification indicates that in the hierarchical approach, slightly less than half of the records are classified using the flat classification in the fourth level of the hierarchy. As a result, the performance of the flat and hierarchical models are not considerably different. In the future, the hierarchical approach can be implemented by a fully top-down algorithm. In the top-down approach, we start from the highest level of the hierarchy and work down the hierarchy. The top-down approach gives the higher levels significantly more control than the lower levels of the hierarchy. Since we know that the classification in the higher levels of the SOC hierarchy is more accurate, using a fully top-down approach can be beneficial.

Finally, social scientists at the University of Guelph can use the models created in this work to assign SOC-2010 codes to Sunshine List's occupation titles. The SOC codes then can be mapped to their skill and experience from the ONET database or they can be mapped to NOC using available crosswalks. Eventually, the occupation codes can be used for analyzing the gender wage gap in the Sunshine List. However, the application of occupation coding system provided in this work is not limited to studying gender wage gap. Another application is analyzing the job market based on online job posts.

Bibliography

[1] U.S. Department of Labor/Employment and Training Administration (USDOL/ETA). Occupation Information Network. `http://www.onetonline.org/`. [Online; accessed 5-October-2016].

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[3] F. Amato, R. Boselli, M. Cesarini, F. Mercorio, M. Mezzanzanica, V. Moscato, F. Persia, and A. Picariello. Classification of web job advertisements: A case study. In *SEBD 2015-Italian Symposium on Advanced Database Systems June 14-17*, pages 144–151, 2015.

[4] L. Antonie, A. D'Angelo, G. Grewal, and M. Plesca. Analyzing the gender wage gap in ontario's public sector. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pages 465–470. IEEE, 2015.

[5] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[6] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447, 2007.

[7] I. Burstyn, A. Slutsky, D. G. Lee, A. B. Singer, Y. An, and Y. L. Michael. Beyond crosswalks: Reliability of exposure assessment following automated coding of free-text job descriptions for occupational epidemiology. *The Annals of Occupational Hygiene*, 58(4):482–492, 2014.

[8] S. Büttcher, C. L. Clarke, and G. V. Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2016.

[9] J. K. Carson, G. N. C. Kirby, A. Dearle, L. Williamson, E. Garrett, A. Reid, and C. J. L. Dibben. Exploiting historical registers: Automatic methods for

coding c19th and c20th cause of death descriptions to standard classifications. *New Techniques and Technologies for Statistics*, pages 598–607, 2013.

[10] B.-C. Chen, R. H. Creecy, and M. V. Appel. Error control of automated industry and occupation coding. *Journal of Official Statistics*, 9(4):729–745, 1993.

[11] R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan, and Y. Wang. The properties of high-dimensional data spaces: implications for exploring gene and protein expression data. *Nature Reviews Cancer*, 8(1):37, 2008.

[12] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[13] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE transactions on pattern analysis and machine intelligence*, 19(4):380–393, 1997.

[14] Environmental Epidemiology, Population Health Research Group, CHUM Research Centre, InVS, and CREDIM. Caps-canada. `http://www.caps-canada.ca/`. [Online; accessed 5-June-2018].

[15] D. Gillman and M. V. Appel. Automated coding research at the census bureau. *Statistical Research Report Series No. RR94*, 4, 1994.

[16] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning*, ICML'11, pages 513–520. Omnipress, 2011.

[17] Government of Canada, Employment and Social Development Canada. National occupational classification - resources and tools. `http://noc.esdc.gc.ca/English/home.aspx`. [Online; accessed 5-October-2016].

[18] H. Gweon, M. Schonlau, L. Kaczmirek, M. Blohm, and S. Steiner. Improving the accuracy of automated occupation coding at any production rate. *SSRN Electronic Journal*, 2016.

[19] J. Han, M. Kamber, and J. Pei. Classification: Advanced methods. In *Data mining: concepts and techniques*. Elsevier, 2011.

[20] D. J. Hand and K. Yu. Idiot's bayes—not so stupid after all? *International statistical review*, 69(3):385–398, 2001.

[21] S. A. Hayati, A. F. Wicaksono, and M. Adriani. Short text classification on complaint documents. *Int. J. Comput. Linguistics Appl.*, 7(2):129–143, 2016.

[22] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification. 2003.

[23] Instituto Nacional de Estadística y Geografía (INEGI), Statistics Canada, and the United States Office of Management and Budget (OMB). North american industry classification system. `https://www.census.gov/cgi-bin/sssd/naics/naicsrch?chart=2012`. [Online; accessed 5-March-2017].

[24] International Labour Organization (ILO). Isco - international standard classification of occupations. [Online; accessed 5-October-2016].

[25] F. Javed, Q. Luo, M. McNair, F. Jacob, M. Zhao, and T. S. Kang. Carotene: A job title classification system for the online recruitment domain. In *2015 IEEE First International Conference on Big Data Computing Service and Applications*, pages 286–293. IEEE, 2015.

[26] F. Javed, M. McNair, F. Jacob, and M. Zhao. Towards a job title classification system. *CoRR*, abs/1606.00917, 2016.

[27] Y. Jung, J. Yoo, S.-H. Myaeng, and D.-C. Han. A web-based automated system for industry and occupation coding. In J. Bailey, D. Maier, K.-D. Schewe, B. Thalheim, and X. S. Wang, editors, *Web Information Systems Engineering - WISE 2008*, pages 443–457. Springer Berlin Heidelberg, 2008.

[28] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188, 2014.

[29] J. Kazama and J. Tsujii. Maximum entropy models with inequality constraints: A case study on text categorization. *Machine Learning*, 60(1):159–194, Sep 2005.

[30] A. T. Kearney and M. E. Kornbau. An automated industry coding application for new us business establishments. In *Proceedings of the American Statistical Association*, 2005.

[31] Y. Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[32] G. Kirby, J. Carson, F. Dunlop, C. Dibben, A. Dearle, L. Williamson, E. Garrett, and A. Reid. Automatic methods for coding historical occupation descriptions

to standard classifications. In *Population Reconstruction*, pages 43–60. Springer International Publishing, 2015.

[33] T. Koeman, N. S. Offermans, Y. Christopher-de Vries, P. Slottje, P. A. Van Den Brandt, R. A. Goldbohm, H. Kromhout, and R. Vermeulen. Jems and incompatible occupational coding systems: effect of manual and automatic recoding of job codes on exposure assignment. *Annals of occupational hygiene*, 57(1):107–114, 2013.

[34] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143. Morgan Kaufmann Publishers Inc., 1995.

[35] D. Laurison and S. Friedman. The class pay gap in higher professional and managerial occupations. *American Sociological Review*, 81(4):668–695, 2016.

[36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[37] H. Lin, B. Sun, J. Wu, and H. Xiong. Topic detection from short text: A term-based consensus clustering method. In *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*, pages 1–6. IEEE, 2016.

[38] Z. Liu, W. Yu, W. Chen, S. Wang, and F. Wu. Short text feature selection for micro-blog mining. In *2010 International Conference on Computational Intelligence and Software Engineering*, pages 1–4. IEEE, 2010.

[39] S. Macchia, M. Murgia, and P. Vicari. Integration between automatic coding and statistical analysis of textual data systems. *Journée d'Analyse des Données Textuelles JADT, Rome*, 2010.

[40] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *proceedings of the 6th conference on Natural language learning-Volume 20*, COLING-02, pages 1–7. Association for Computational Linguistics, 2002.

[41] C. D. Manning, P. Raghavan, H. Schütze, et al. Text classification and naive bayes. In *Introduction to information retrieval*, volume 1, chapter 13. Cambridge university press Cambridge, 2008.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[43] NASWA Information Technology Support Center - In Partnership with USDOL. Occucoder. http://www.itsc.org/Pages/pub_aocode.aspx.

[44] Ontario Ministry of Finance. Public sector salary disclosure. https://www.ontario.ca/page/public-sector-salary-disclosure. [Online; accessed 1-September-2018].

[45] E. M. Ossiander and S. Milham. A computer system for coding occupation. *American journal of industrial medicine*, 49(10):854–857, 2006.

[46] M. D. Patel, K. M. Rose, C. R. Owens, H. Bang, and J. S. Kaufman. Performance of automated and manual coding systems for occupational data: A case study of historical records. *American journal of industrial medicine*, 55(3):228–231, 2012.

[47] 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.

[48] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[49] P. C. Rafeeque and S. Sendhilkumar. A survey on short text analysis in web. In *2011 Third International Conference on Advanced Computing*, pages 365–371. IEEE, 2011.

[50] N. M. Razali and Y. B. Wah. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

[51] P. Rivière. Automated coding. *Statistical data editing: Methods and techniques*, pages 173–180, 1997.

[52] M. Rogati and Y. Yang. High-performing feature selection for text classification. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 659–661. ACM, 2002.

[53] K. D. Rosa and J. Ellen. Text classification methodologies applied to micro-text in military chat. In *2009 International Conference on Machine Learning and Applications*, ICMLA'09, pages 710–714. IEEE, 2009.

[54] E. Rowe, C. Wong, and A. C. Staff. *An Introduction to the ACTR Coding System*. Bureau of the Census, 1994.

[55] D. E. Russ, K.-Y. Ho, J. S. Colt, K. R. Armenti, D. Baris, W.-H. Chow, F. Davis, A. Johnson, M. P. Purdue, M. R. Karagas, K. Schwartz, M. Schwenn, D. T. Silverman, C. A. Johnson, and M. C. Friesen. Computer-based coding of free-text job descriptions to efficiently identify occupations in epidemiological studies. *Occupational and Environmental Medicine*, 2016.

[56] D. E. Russ, K. Y. Ho, C. A. Johnson, and M. C. Friesen. Computer-based coding of occupation codes for epidemiological analyses. In *2014 IEEE 27th International Symposium on Computer-Based Medical Systems*, pages 347–350, 2014.

[57] G. Salton and J. Michael. Mcgill. 1983. *Introduction to modern information retrieval*, 1983.

[58] M. Schierholz. Automating survey coding for occupation. Phd dissertation, Ludwig Maximilian University of Munich, 2014.

[59] K. Sechidis, G. Tsoumakas, and I. Vlahavas. On the stratification of multi-label data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 145–158. Springer, 2011.

[60] F. P. Shah and V. Patel. A review on feature selection and feature extraction for text classification. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 2264–2268. IEEE, 2016.

[61] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[62] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[63] A. 't Mannetje and H. Kromhout. The use of occupation and industry classifications in general population studies. *International Journal of Epidemiology*, 32(3):419–428, 2003.

[64] K. Takahashi, H. Takamura, and M. Okumura. Automatic occupation coding with combination of machine learning and hand-crafted rules. In T. B. Ho,

D. Cheung, and H. Liu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 269–279. Springer Berlin Heidelberg, 2005.

[65] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[66] The National Institute for Occupational Safety and Health (NIOSH). Nioccs: Niosh industry and occupation computerized coding system. `https://www.cdc.gov/niosh/topics/coding/how.html`. [Online; accessed 5-October-2017].

[67] M. Thompson, M. E. Kornbau, and J. Vesely. Creating an automated industry and occupation coding process for the american community survey. *rapport inédit*, 2012.

[68] A. Tommasel and D. Godoy. Short-text feature construction and selection in social media data: a survey. *Artificial Intelligence Review*, 49(3):301–338, 2018.

[69] J. Tourigny and J. Moloney. The 1991 canadian census of population experience with automated coding. *United Nations Statistical Commission, Statistical Data Editing*, 2, 1995.

[70] G. Tsoumakas, I. Katakis, and I. Vlahavas. *Mining Multi-label Data*, pages 667–685. Springer US, Boston, MA, 2010.

[71] University of Warwick Institute for Employment Research. Cascot: Computer assisted structured coding tool. `http://www2.warwick.ac.uk/fac/soc/ier/software/cascot/`. [Online; accessed 10-October-2017].

[72] U.S. Census Bureau. U.S. Census industry and occupations. `http://www.census.gov/people/io/methodology/`. [Online; accessed 5-October-2016].

[73] U.S. Department of Labor. Standard occupation classification. `http://www.bls.gov/soc/`. [Online; accessed 5-October-2016].

[74] D.-T. Vo and C.-Y. Ock. Learning to classify short text from scientific documents using topic models with various types of knowledge. *Expert Systems with Applications*, 42(3):1684–1698, 2015.

[75] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye. *Probability and statistics for engineers and scientists*, volume 5. Macmillan New York, 1993.

[76] S. Wang and C. D. Manning. 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, ACL '12, pages 90–94. Association for Computational Linguistics, 2012.

[77] X. Wang, R. Chen, Y. Jia, and B. Zhou. Short text classification using wikipedia concept based document representation. In *2013 International Conference on Information Technology and Applications*, pages 471–474. IEEE, 2013.

[78] M. Wenzowski. Actr–a generalised automated coding system. *Survey Methodology*, 14(2):299–308, 1988.

[79] D. Wettschereck, D. W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11(1):273–314, Feb 1997.

[80] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, 1997.

[81] M.-L. Zhang and Z.-H. Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2014.

[82] Y. Zhang and B. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

Appendix A

SVM and MaxEnt Hyper-Parameter Tuning

The hyper-parameter tuning is done on the first level of the SOC hierarchy. Table A.1 shows the micro/macro F1 when a grid search is implemented on the validation set (9,928 records) to tune the hyper-parameter C of SVM. 10-fold cross validation with values of $C = (2^{-5}, 2^{-4}, ..., 2^4, 2^5)$ is used. $C = 1$ resulted in the best score and is shown in bold.

Table A.1: Results of grid search using micro F1 (left) and macro F1 (right) as the measure to tune the hyper-parameter C of SVM

C	Micro F1	Standard Deviation
2^{-5}	0.46329	0.01323
2^{-4}	0.52057	0.01096
2^{-3}	0.57395	0.01430
2^{-2}	0.60831	0.01330
2^{-1}	0.63228	0.01048
2^{0}	**0.63905**	0.00769
2^{1}	0.63416	0.00695
2^{2}	0.62319	0.00759
2^{3}	0.61072	0.00830
2^{4}	0.59642	0.00922
2^{5}	0.58079	0.00895

C	Macro F1	Standard Deviation
2^{-5}	0.25566	0.0093
2^{-4}	0.33295	0.0115
2^{-3}	0.42332	0.0213
2^{-2}	0.49134	0.0188
2^{-1}	0.54338	0.0119
2^{0}	**0.56241**	**0.0105**
2^{1}	0.56156	0.0073
2^{2}	0.55203	0.0105
2^{3}	0.53722	0.012
2^{4}	0.52107	0.0151
2^{5}	0.50492	0.0156

Figures A.1 and A.2 show the micro/macro F1 for values of the regularization parameter $C = (2^{-5}, 2^{-4}, ..., 2^4, 2^5)$ with different number of iterations $I = (10, 20, ..., 90, 100)$. The figures show that MaxEnt converges before 50 number of iterations. Therefore, we chose $C = 16$ and $I = 50$ for MaxEnt's hyper-parameters.

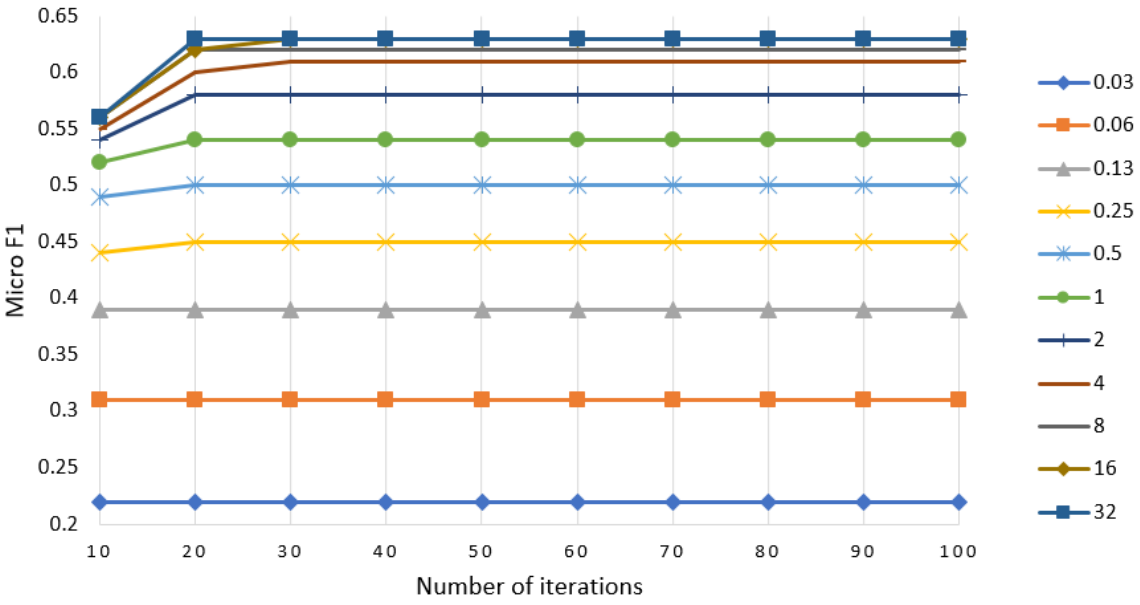

Figure A.1: Results of grid search on MaxEnt using micro F1 as the measure.

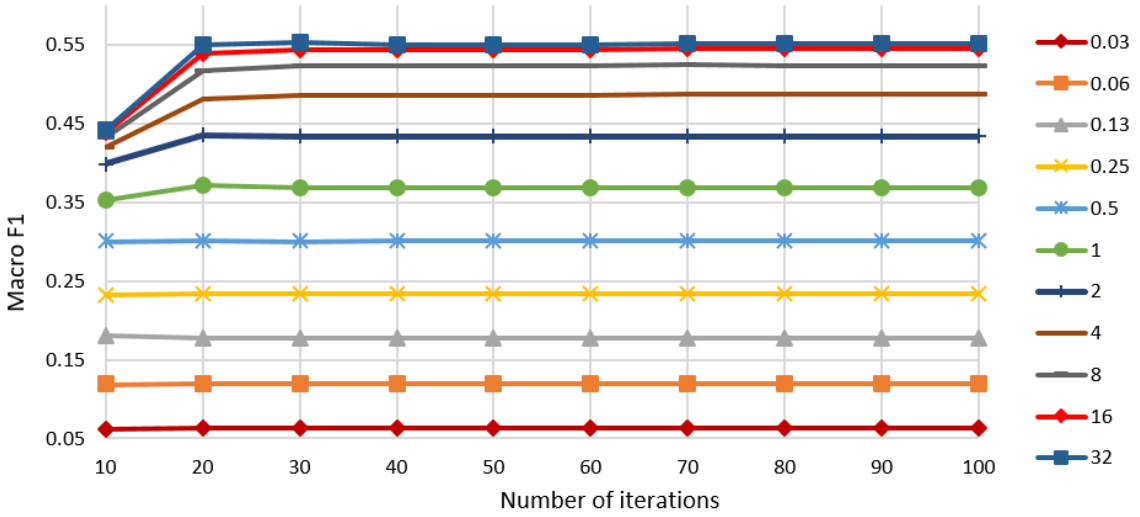

Figure A.2: Results of grid search on MaxEnt using macro F1 as the measure.

Appendix B

Pre-processing

Table B.1 shows the stop words list used by the classic machine learning classifiers.

Table B.1: Stop words list

i	its	a	before	any
me	itself	an	after	both
my	they	the	above	each
myself	them	and	below	few
we	their	but	to	more
our	themselves	if	from	most
ours	what	or	up	other
ourselves	which	because	in	some
you	this	as	out	such
your	that	until	on	no
yours	these	while	off	nor
yourself	those	of	over	not
yourselves	is	at	under	own
he	are	by	again	same
him	was	for	further	so
his	were	with	then	than
himself	be	about	there	too
she	been	against	when	very
her	have	between	where	should
hers	has	into	why	
herself	had	through	how	
it	do	during	all	

42 stop words were detected in the data, these stop words with their frequency

and document frequency are listed in table B.2. The frequency represents the frequency of the stop words in the entire data and the document frequency represents the frequency of documents containing each stop word.

Table B.2: Stop words found in the data and their frequency and document frequency

Stop words	**Frequency**	**Document Frequency**	**Stop words**	**Frequency**	**Document Frequency**
and	3,673	3,521	under	11	11
of	763	759	as	9	9
or	528	518	own	8	8
up	306	304	through	8	8
other	162	162	is	7	7
an	146	142	if	3	3
in	142	139	above	2	2
all	120	120	after	2	2
out	109	109	do	2	2
off	74	74	no	2	2
any	64	64	so	2	2
on	57	57	such	2	2
for	53	53	against	1	1
the	50	50	before	1	1
it	44	44	below	1	1
to	34	34	both	1	1
over	24	24	he	1	1
at	19	19	him	1	1
not	19	19	its	1	1
than	18	18	some	1	1
with	15	15	very	1	1

Appendix C

Statistical Analysis

C.1 Normality Test

Table C.1 shows the results of Shapiro-Wilk normality test. The test is implemented on the first level micro F1 resulted from 10-fold cross validation on SVM classifiers. If the P-value is less than the significance level 0.05, we can reject the null hypothesis that the samples come from a normally distributed population. However, the test cannot confirm that the samples come from a normally distributed population if the P-value is greater than 0.05. Since the P-values are greater than the significance level, we cannot reject the null hypothesis. Therefore, we look at the Q-Q plots of the samples. Figure C.1 shows the Normal Q-Q plots for first level micro F1 of SVM classifiers. If the samples are aligned on the red line, they come from a population with normal distribution. Since there is a difference in the alignment of the Q-Q plots in Fig. C.1, we are unable to confidently say all samples come from a normally distributed population. Therefore, we use the Wilcoxon signed rank test that does not assume the samples come from a population with normal distribution.

Table C.1: Shapiro-Wilk normality test on the first level micro F1 of the flat SVM classifiers

Classifier	**W**	**P-value**	**Sample Size**
SVM	0.86907	0.09749	10
SVM-bigram	0.89101	0.1741	10
SVM-fpr	0.86907	0.09749	10
SVM-fpr-bigram	0.96676	0.8593	10
SVM-hier	0.92771	0.4257	10

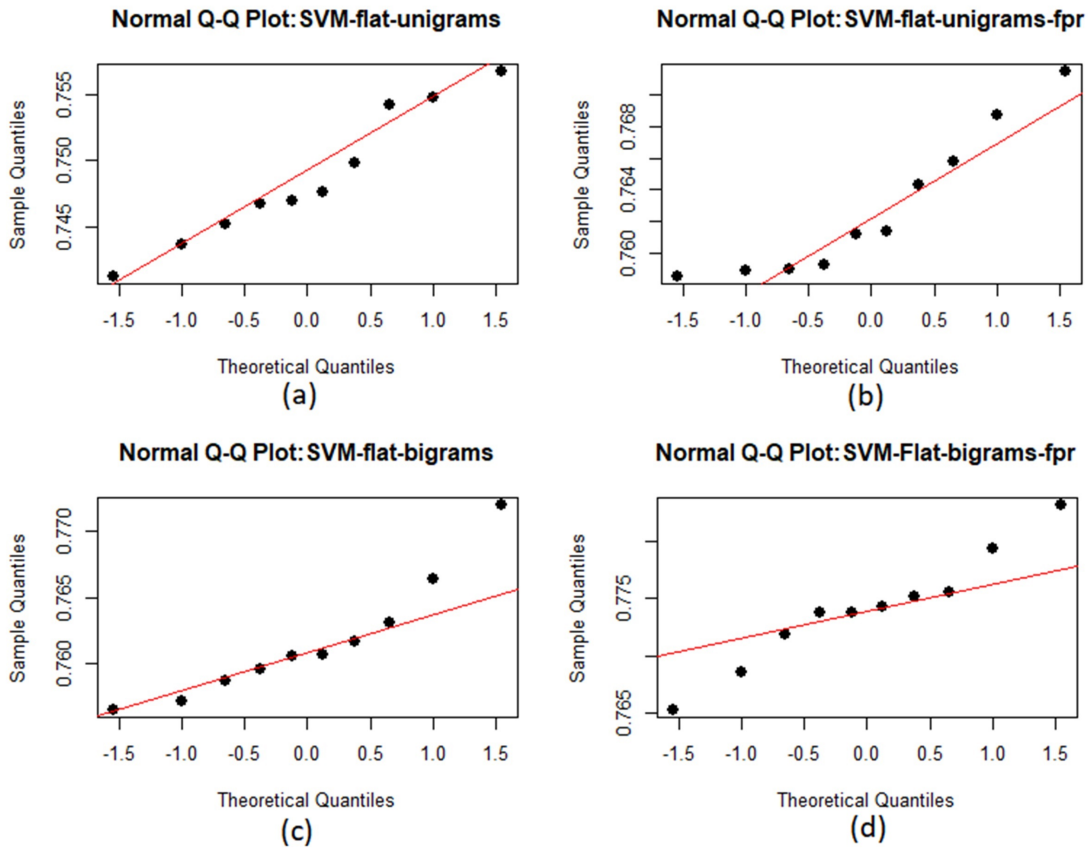

Figure C.1: Normal Q-Q plots for first level micro F1 of flat SVM classifiers

C.2 Significance Test

In this section, we use Wilcoxon signed rank test to compare the flat and hierarchical models. Section C.2.1 compares flat models; Section C.2.2 compares hierarchical models; Section C.2.3 compares flat and hierarchical models.

C.2.1 Comparison of Flat Models

Tables C.2 to C.8 show the results of the significance test on micro/macro F1 for the flat classification models.

Table C.2: Wilcoxon signed rank test for first level flat classifiers with micro F1 as the measure

Hypothesis	V-value	P-value
SVM-fpr-bigram > SVM-unigram	55	0.002367
SVM-fpr-bigram > SVM-bigram	55	0.001762
SVM-fpr-bigram > SVM-fpr-unigram	55	0.001371
SVM-fpr-bigram > NB-unigram	55	0.002353
SVM-fpr-bigram > NB-bigram	55	0.002367
SVM-fpr-bigram > NB-fpr-unigram	55	0.002118
SVM-fpr-bigram > NB-fpr-bigram	55	0.002078
SVM-fpr-bigram > MaxEnt-unigram	55	0.001774
SVM-fpr-bigram > MaxEnt-bigram	55	0.002118
SVM-fpr-bigram > MaxEnt-fpr-unigram	55	0.001762

Hypothesis	V-value	P-value
MaxEnt-fpr-bigram>SVM-unigram	55	0.00241
MaxEnt-fpr-bigram>SVM-bigram	45	0.002382
MaxEnt-fpr-bigram>SVM-fpr-unigram	36	0.004167
MaxEnt-fpr-bigram>NB-unigram	55	0.002572
MaxEnt-fpr-bigram>NB-bigram	55	0.002498
MaxEnt-fpr-bigram>NB-fpr-unigram	55	0.002572
MaxEnt-fpr-bigram>NB-fpr-bigram	55	0.002353
MaxEnt-fpr-bigram>MaxEnt-unigram	55	0.002367
MaxEnt-fpr-bigram>MaxEnt-bigram	55	0.002118
MaxEnt-fpr-bigram>MaxEnt-fpr-unigram	36	0.005139

Table C.3: Wilcoxon signed rank test for first level flat classifiers with macro F1 as the measure

Hypothesis	V-value	P-value
SVM-fpr-bigram > SVM-unigram	55	0.002724
SVM-fpr-bigram > SVM-bigram	55	0.002283
SVM-fpr-bigram > SVM-fpr-unigram	21	0.01527
SVM-fpr-bigram > NB-unigram	55	0.002367
SVM-fpr-bigram > NB-bigram	55	0.002587
SVM-fpr-bigram > NB-fpr-unigram	55	0.002132
SVM-fpr-bigram > NB-fpr-bigram	55	0.002647
SVM-fpr-bigram > MaxEnt-unigram	55	0.002724
SVM-fpr-bigram > MaxEnt-bigram	55	0.002817
SVM-fpr-bigram > MaxEnt-fpr-unigram	36	0.00577
SVM-fpr-bigram > MaxEnt-fpr-bigram	10	0.03593

Table C.4: Wilcoxon signed rank test for second level flat classifier with micro F1 as the measure

Hypothesis	**V-value**	**P-value**
SVM-fpr-bigram > SVM-unigram	55	0.002353
SVM-fpr-bigram > SVM-bigram	55	0.001762
SVM-fpr-bigram > SVM-fpr-unigram	36	0.002981
SVM-fpr-bigram > NB-unigram	55	0.002498
SVM-fpr-bigram > NB-bigram	55	0.002678
SVM-fpr-bigram > NB-fpr-unigram	55	0.002632
SVM-fpr-bigram > NB-fpr-bigram	55	0.002572
SVM-fpr-bigram > MaxEnt-unigram	55	0.002118
SVM-fpr-bigram > MaxEnt-bigram	55	0.002353
SVM-fpr-bigram > MaxEnt-fpr-unigram	45	0.001677
SVM-fpr-bigram > MaxEnt-fpr-bigram	15	0.01844

Table C.5: Wilcoxon signed rank test for second level flat classifiers with macro F1 as the measure

Hypothesis	**V-value**	**P-value**
SVM-fpr-unigram > SVM-unigram	55	0.002572
SVM-fpr-unigram > SVM-bigram	55	0.002739
SVM-fpr-unigram > NB-unigram	55	0.002724
SVM-fpr-unigram > NB-bigram	55	0.002367
SVM-fpr-unigram > NB-fpr-unigram	55	0.00241
SVM-fpr-unigram > NB-fpr-bigram	55	0.001774
SVM-fpr-unigram > MaxEnt-unigram	55	0.002498
SVM-fpr-unigram > MaxEnt-bigram	55	0.002647
SVM-fpr-unigram > MaxEnt-fpr-bigram	21	0.009828

Hypothesis	**V-value**	**P-value**
SVM-fpr-bigram > SVM-unigram	55	0.002632
SVM-fpr-bigram > SVM-bigram	55	0.002786
SVM-fpr-bigram > NB-unigram	55	0.002881
SVM-fpr-bigram > NB-bigram	55	0.002849
SVM-fpr-bigram > NB-fpr-unigram	55	0.002849
SVM-fpr-bigram > NB-fpr-bigram	55	0.002849
SVM-fpr-bigram > MaxEnt-unigram	55	0.002678
SVM-fpr-bigram > MaxEnt-bigram	55	0.00277
SVM-fpr-bigram > MaxEnt-fpr-bigram	21	0.01601

Hypothesis	**V-value**	**P-value**
MaxEnt-fpr-unigram > SVM-unigram	55	0.001762
MaxEnt-fpr-unigram > SVM-bigram	55	0.002817
MaxEnt-fpr-unigram > NB-unigram	55	0.002865
MaxEnt-fpr-unigram > NB-bigram	55	0.002724
MaxEnt-fpr-unigram > NB-fpr-unigram	55	0.002817
MaxEnt-fpr-unigram > NB-fpr-bigram	55	0.002617
MaxEnt-fpr-unigram > MaxEnt-unigram	55	0.002283
MaxEnt-fpr-unigram > MaxEnt-bigram	55	0.002587
MaxEnt-fpr-unigram > MaxEnt-fpr-bigram	24	0.0363

Table C.6: Wilcoxon signed rank test for third level flat classifiers with micro F1 as measure

Hypothesis	**V-value**	**P-value**
SVM-fpr-unigram > SVM-unigram	55	0.00241
SVM-fpr-unigram > SVM-bigram	55	0.002367
SVM-fpr-unigram > NB-unigram	55	0.002572
SVM-fpr-unigram > NB-bigram	55	0.001371
SVM-fpr-unigram > NB-fpr-unigram	55	0.002118
SVM-fpr-unigram > NB-fpr-bigram	55	0.001774
SVM-fpr-unigram > MaxEnt-unigram	55	0.002367
SVM-fpr-unigram > MaxEnt-bigram	55	0.002724
SVM-fpr-unigram > MaxEnt-fpr-bigram	36	0.062981

Hypothesis	**V-value**	**P-value**
SVM-fpr-bigram > SVM-unigram	55	0.002724
SVM-fpr-bigram > SVM-bigram	55	0.002572
SVM-fpr-bigram > NB-unigram	55	0.002587
SVM-fpr-bigram > NB-bigram	55	0.002367
SVM-fpr-bigram > NB-fpr-unigram	55	0.002572
SVM-fpr-bigram > NB-fpr-bigram	55	0.00241
SVM-fpr-bigram > MaxEnt-unigram	55	0.002367
SVM-fpr-bigram > MaxEnt-bigram	55	0.002724
SVM-fpr-bigram > MaxEnt-fpr-bigram	15	0.07844

Hypothesis	**V-value**	**P-value**
MaxEnt-fpr-unigram > SVM-unigram	55	0.002724
MaxEnt-fpr-unigram > SVM-bigram	55	0.002498
MaxEnt-fpr-unigram > NB-unigram	55	0.002498
MaxEnt-fpr-unigram > NB-bigram	55	0.002353
MaxEnt-fpr-unigram > NB-fpr-unigram	55	0.002367
MaxEnt-fpr-unigram > NB-fpr-bigram	55	0.002283
MaxEnt-fpr-unigram > MaxEnt-unigram	55	0.002632
MaxEnt-fpr-unigram > MaxEnt-bigram	55	0.002724
MaxEnt-fpr-unigram > MaxEnt-fpr-bigram	3	0.1729

Table C.7: Wilcoxon signed rank test for third level flat classifiers with macro F1 as the measure

Hypothesis	V-value	P-value
SVM-fpr-unigram > SVM-unigram	55	0.002498
SVM-fpr-unigram > SVM-bigram	55	0.002283
SVM-fpr-unigram > SVM-fpr-bigram	45	0.002382
SVM-fpr-unigram > NB-unigram	55	0.002396
SVM-fpr-unigram > NB-bigram	55	0.002587
SVM-fpr-unigram > NB-fpr-unigram	55	0.002498
SVM-fpr-unigram > NB-fpr-bigram	55	0.001774
SVM-fpr-unigram > MaxEnt-unigram	55	0.002724
SVM-fpr-unigram > MaxEnt-bigram	55	0.002739
SVM-fpr-unigram > MaxEnt-fpr-unigram	24.5	0.03538
SVM-fpr-unigram > MaxEnt-fpr-bigram	55	0.001371

Table C.8: Wilcoxon signed rank test for fourth level flat classifiers with micro F1 (left) and macro F1 (right) as the measure

Hypothesis	V-value	P-value
SVM-fpr-unigram > SVM-unigram	55	0.002587
SVM-fpr-unigram > SVM-bigram	55	0.00241
SVM-fpr-unigram > SVM-fpr-bigram	28	0.08941
SVM-fpr-unigram > NB-unigram	55	0.002132
SVM-fpr-unigram > NB-bigram	55	0.002132
SVM-fpr-unigram > NB-fpr-unigram	55	0.002118
SVM-fpr-unigram > NB-fpr-bigram	55	0.00241
SVM-fpr-unigram > MaxEnt-unigram	55	0.00277
SVM-fpr-unigram > MaxEnt-bigram	55	0.002632
SVM-fpr-unigram > MaxEnt-fpr-unigram	21	0.09828
svm-fpr-unigram > MaxEnt-fpr-bigram	20	0.0812

Hypothesis	V-value	P-value
SVM-fpr-unigram > SVM-unigram	55	0.002587
SVM-fpr-unigram > SVM-bigram	55	0.00241
SVM-fpr-unigram > SVM-fpr-bigram	55	0.002617
SVM-fpr-unigram > NB-unigram	55	0.00241
SVM-fpr-unigram > NB-bigram	55	0.002498
SVM-fpr-unigram > NB-fpr-unigram	55	0.002367
SVM-fpr-unigram > NB-fpr-bigram	55	0.00241
SVM-fpr-unigram > MaxEnt-unigram	55	0.00241
SVM-fpr-unigram > MaxEnt-bigram	55	0.002849
SVM-fpr-unigram > MaxEnt-fpr-unigram	24	0.0363
svm-fpr-unigram > MaxEnt-fpr-bigram	55	0.002724

C.2.2 Comparison of Hierarchical Models

Table C.9 to C.12 show the results of the significance test on micro/macro F1 for the hierarchical classification models.

Table C.9: Wilcoxon signed rank test for first level hierarchical classifiers with macro F1 as the measure

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-hier-bigram	3	0.1729
SVM-hier-unigram > MaxEnt-hier-bigram	28	0.007354
MaxEnt-hier-unigram > MaxEnt-hier-bigram	28	0.007354
MaxEnt-hier-unigram > SVM-hier-bigram	14	0.242

Table C.10: Wilcoxon signed rank test for second level hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-hier-bigram	15	0.01844
SVM-hier-unigram > MaxEnt-hier-bigram	28	0.007354
MaxEnt-hier-unigram > MaxEnt-hier-bigram	15	0.01844
MaxEnt-hier-unigram > SVM-hier-bigram	21	0.01527

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-hier-bigram	45	0.003706
SVM-hier-unigram > MaxEnt-hier-bigram	55	0.002367
MaxEnt-hier-unigram > MaxEnt-hier-bigram	45	0.003985
MaxEnt-hier-unigram > SVM-hier-bigram	36	0.00577

Table C.11: Wilcoxon signed rank test for third level hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure

Hypothesis	V-value	P-value
MaxEnt-hier-bigram < SVM-hier-unigram	0	0.002078
MaxEnt-hier-bigram < SVM-hier-bigram	0	0.004167
MaxEnt-hier-bigram < MaxEnt-hier-unigram	0	0.003004

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-hier-bigram	55	0.001762
SVM-hier-unigram > MaxEnt-hier-bigram	55	0.002078
MaxEnt-hier-unigram > MaxEnt-hier-bigram	36	0.005988
MaxEnt-hier-unigram > SVM-hier-bigram	21	0.01313

Table C.12: Wilcoxon signed rank test for fourth level hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-hier-bigram	28	0.007354
SVM-hier-unigram > MaxEnt-hier-bigram	45	0.004129
MaxEnt-hier-unigram > MaxEnt-hier-bigram	45	0.003029
MaxEnt-hier-unigram > SVM-hier-bigram	6	0.07446

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-hier-bigram	36	0.005933
SVM-hier-unigram > MaxEnt-hier-bigram	55	0.002724
MaxEnt-hier-unigram > MaxEnt-hier-bigram	45	0.004365
MaxEnt-hier-unigram > SVM-hier-bigram	36	0.00519

C.2.3 Comparison of Flat and Hierarchical Models

Table C.13 to C.16 show the significance test on micro/macro F1, comparing flat and hierarchical models from first level to fourth level of the SOC hierarchy.

Table C.13: Wilcoxon signed rank test for first level hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure

Hypothesis	V-value	P-value
SVM-fpr-bigram > SVM-hier-bigram	55	0.002353
MaxEnt-fpr-bigram > MaxEnt-hier-bigram	45	0.003004

Hypothesis	V-value	P-value
SVM-fpr-bigram > SVM-hier-bigram	45	0.003706
MaxEnt-fpr-bigram > MaxEnt-hier-bigram	55	0.002283

Table C.14: Wilcoxon signed rank test for second level hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure

Hypothesis	V-value	P-value
SVM-fpr-bigram > SVM-hier-bigram	45	0.003706
MaxEnt-fpr-bigram > MaxEnt-hier-bigram	45	0.003004

Hypothesis	V-value	P-value
SVM-fpr-bigram > SVM-hier-bigram	36	0.005988

Table C.15: Wilcoxon signed rank test for third level hierarchical classifiers with macro F1 as the measure

Hypothesis	V-value	P-value
MaxEnt-fpr-unigram < MaxEnt-hier-unigram	0	0.0678

Table C.16: Wilcoxon signed rank test for fourth level hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure

Hypothesis	V-value	P-value
MaxEnt-fpr-unigram < MaxEnt-hier-unigram	0	0.09828

Hypothesis	V-value	P-value
MaxEnt-fpr-unigram < MaxEnt-hier-unigram	0	0.05294

We also compare the performance of flat and hierarchical models on the portion of data that was hierarchically classified. Table C.17 to C.20 show the significance test on micro/macro F1 from the first level to the fourth level of the SOC hierarchy.

Table C.17: Wilcoxon signed rank test for first level flat and hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure. The models are implemented on the portion of data that was hierarchically classified

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-flat-unigram	55	0.0009521
MaxEnt-hier-unigram > MaxEnt-flat-unigram	55	0.002353

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-flat-unigram	45	0.003004
MaxEnt-hier-unigram > MaxEnt-flat-unigram	55	0.002118

Table C.18: Wilcoxon signed rank test for second level flat and hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure. The models are implemented on the portion of data that was hierarchically classified

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-flat-unigram	55	0.002572
MaxEnt-hier-unigram > MaxEnt-flat-unigram	55	0.002632

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-flat-unigram	55	0.002647
MaxEnt-hier-unigram > MaxEnt-flat-unigram	55	0.002498

Table C.19: Wilcoxon signed rank test for third level flat and hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure. The models are implemented on the portion of data that was hierarchically classified

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-flat-unigram	55	0.002283
MaxEnt-hier-unigram > MaxEnt-flat-unigram	55	0.002283

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-flat-unigram	45	0.001677
MaxEnt-hier-unigram > MaxEnt-flat-unigram	45	0.003706

Table C.20: Wilcoxon signed rank test for fourth level flat and hierarchical classifiers with micro F1 (left) and macro F1 (right) as the measure. The models are implemented on the portion of data that was hierarchically classified

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-flat-unigram	45	0.003004
MaxEnt-hier-unigram > MaxEnt-flat-unigram	55	0.001762

Hypothesis	V-value	P-value
SVM-hier-unigram > SVM-flat-unigram	10	0.04734