3-2021

Improving Text Classification with Semantic Information

Joshua H. White

IMPROVING TEXT CLASSIFICATION WITH SEMANTIC INFORMATION

THESIS

Joshua H. White, Capt, USAF

AFIT-ENG-MS-21-M-092

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-21-M-092

IMPROVING TEXT CLASSIFICATION WITH SEMANTIC INFORMATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Science

Joshua H. White, B.S.C.S.

Capt, USAF

March 26, 2021

AFIT-ENG-MS-21-M-092

IMPROVING TEXT CLASSIFICATION WITH SEMANTIC INFORMATION

THESIS

Joshua H. White, B.S.C.S.
Capt, USAF

Committee Membership:

Lt Col George E. Noel, Ph.D
Chair

Gilbert L. Peterson, Ph.D
Member

Lt Col David W. King, Ph.D
Member

AFIT-ENG-MS-21-M-092

Abstract

The Air Force contracts a variety of positions, from Information Technology to maintenance services. There is currently no automated way to verify that quotes for services are reasonably priced. Small training data sets and word sense ambiguity are challenges that such a tool would encounter, and additional semantic information could help. This thesis hypothesizes that leveraging a semantic network could improve text-based classification. This thesis uses information from ConceptNet to augment a Naive Bayes Classifier. The leveraged semantic information would add relevant words from the category domain to the model that did not appear in the training data. The experiment compares variations of a Naive Bayes Classifier leveraging semantic information, including an Ensemble Model, against classifiers that do not. Results show a significant performance increase in a smaller data set but not a larger one. Out of all models tested, an Ensemble Based Classifier performs the best on both data sets. The results show that ConceptNet does not add enough new or relevant information to affect classifier performance on large data sets.

Table of Contents

Page

List of Figures

List of Tables

IMPROVING TEXT CLASSIFICATION WITH SEMANTIC INFORMATION

I. Introduction

1.1 Problem Statement

The Air Force contracts various services ranging from Information Technology (IT) services to groundskeepers when it needs new services. Contractor proposals are compared against each other, but the Air Force also wants individual manpower quotes comparable to market salaries for equivalent positions. There is currently no automatic way for an organization to verify that the quotes are reasonably priced. A tool to automatically verify quoted prices would be useful to Air Force contracting personnel, but currently none exist. This tool could ingest commercially available or historical service data to produce accurate results.

There are several challenges with building an automated tool to predict salaries. The first is obtaining a data set for training the tool that reflects the area the Air Force is looking to hire in and encompasses the different job categories the tool is trying to predict. This tool needs a large enough training data set to make correct predictions. If a large enough training data set for the required professions is not available, then a method to add relevant words to the available training data set would be valuable.

The second challenge is words can have different meanings in the context of different professions. The difference of meaning could lead to a word being more critical for predicting a job belonging to one job versus another. For example, the word "set" in a Computer Science context could be referring to a Python data structure. But the word “set” can mean just a pair of things in another context, like a “set of tools” for a mechanic. The word “set” is just one example of a typical word having a different meaning between professions.

This thesis examines the classification step of creating a machine learning-based automated tool. The classification step takes the raw text for a job description and the job title and outputs the job category. This initial classification step is crucial because there is a one-to-one mapping of job categories to salary ranges. This mapping of job position to salary range can be found from historical data or directly from a contracting company. The next steps of the automated tool would examine the words in the category’s context to predict the job posting’s final salary. An example of the whole process would be the tool ingests the text data for a senior engineering position. The tool would first classify the job as an engineering job. Then the tool would determine the specific engineering job and output its salary. This thesis does not delve into the final salary prediction, only the model’s initial classification aspect of a job posting.

1.2 Hypothesis

This research hypothesizes that leveraging information from a semantic network will improve text-based classification for job postings. Specifically, this thesis will attempt to prove the hypothesis by leveraging information from a semantic network called ConceptNet to improve a Naive Bayes Classifier’s (NBC) predictions. ConceptNet is a graph where each node is a word, and relationships between words are the edges. The example “an apple is a fruit” is a pair of nodes and an edge, where “apple” and “fruit” are the nodes and “is a” is the edge.

The concept behind using ConceptNet to improve an NBC stems from how it makes predictions. The NBC maintains a multinomial distribution for each category in the data set. Think of the multinomial distribution as overlaying the words from the training set onto ConceptNet. The words for a training set would create a cluster on the graph that represents that category. Using the edges in ConceptNet, we can then expand the cluster one edge at a time and try to create a more extensive cluster. Using the new words in this larger cluster from ConceptNet, we can expand the words in a multinomial distribution for each category in the NBC. These additional words from ConceptNet are the additional semantic information we are leveraging in our model.

1.3 Approach

This thesis's approach is first, create the NBC that leverages semantic information. This thesis's classification models are all created in python and use the same two data sets for training and generating results. ConceptNet is a semantic network leveraged to create the categorical distributions for every category in the data sets. The NBC then utilizes these categorical distributions to make predictions.

The next step of this thesis's experiment is to create the comparative models using the same data sets as the NBC. The comparative models are a Random Forest Classifier, K-Nearest Neighbor based classifier, an NBC without semantic information, an ensemble model, and a Neural Network based classifier. The K-Nearest Neighbor and Random Forest Classifiers were both created with the Scikit-Learn python package. Keras and Tensorflow are the packages used to create the Neural Network based classifier.

1.4 Thesis Overview

This thesis document is arranged into five chapters. Chapter two presents a summary of relevant research dealing with resumes and job postings, classification methods, and semantic networks. Chapter three introduces the design and methodology of the various classification models used in the experimentation. The results are presented and analyzed in chapter four. Finally, chapter five concludes this thesis document and discusses future work opportunities in this domain.

II. Background and Related Work

2.1 Overview

There has been much research done involving resumes and job postings, from finding the best job for a given resume [1] or trying to predict the salary range given the text of a job advertisement [2]. This chapter will provide some background and knowledge, starting with previous research dealing with resumes and job postings. The following subsections will then cover other technical parts of the hypothesis, such as classification methods, statistical models, and knowledge graphs/semantic networks.

2.2 Previous Resume and Job Description Research

Resumes are required for job applications, and due to the sheer number of applicants, resumes are written in a range of formats. This diversity can adversely affect the information retrieval (IR) processes in data mining on resumes, but multiple studies have been conducted to improve the parsing and extraction process. One proposed system [3] uses a set of language processing techniques, part heuristic-based, and part pattern matching, for automatic resume management. The system is based on the fact that a company's HR Department will look for specific information, and the extraction process is tailored to that information. After processing, the resumes are placed into a database with a web-base front end for a company's HR to search through. Another resume IR algorithm was proposed that used a two-step process, called Text Block Classification and Resume Facts Identification, specifically designed to handle resumes of any hierarchical structure for later manipulation [4]. The first step takes the raw text and parses it into different blocks, or categories, of information designated by the researchers. The second step uses multiple classifiers to identify different attributes of information in the resumes.

Parsing resumes correctly is just the first step for systems wanting to match resumes to potential jobs for job-seeking candidates. One approach to resume matching was performed using a Deep Siamese Network, which is a pair of identical Convolutional Neural Networks (CNN), to match resumes to job descriptions over various domains [5]. The network was trained using document embeddings, generated with an algorithm called Doc2Vec, of resumes and job descriptions as input. Another job recommender system [6] approached the problem by first clustering users based on their website's activity. This activity consisted of click frequency, what they clicked on, job search frequency, and comment frequency to group users using clustering into one of three categories: proactive, passive, and moderate. Based on the category the user was classified as, the system used a different technique to recommend jobs. Another set of researchers created an algorithm that uses K-means++ to cluster users based on job title, then uses a mixture of Term Frequency-Inverse Document Frequency (TF-IDF) and part of speech weight to evaluate how the resumes will match up to a given job description [7]. Also, a graph-based approach [8] was created where, initially to create the graph, the job postings are the nodes and edges representing a content similarity score between pairs of job descriptions through a Deep Learning Matcher, a type of neural network. Then the systems create a vector out of a new user's resume and extract a subgraph based on the vector. Finally, PageRank, a ranking algorithm used by Google search, is used on the subgraph to return a ranked list of job recommendations to the user.

Researchers have also done work on techniques that ranks a pool of resumes of potential hires for organizations. One method of resume ranking [9] creates multiple ontology weighted graphs for a specific industry field, extracts the skill requirements from a job posting and job seekers skill items via pattern matching, and compares the graph weights to rank resumes. In this case, a specific field is divided into multiple sub-ontology graphs, and the researchers create the edge weights. A content-based method [10] in which the company creates a set of item features, or preferences, for a specific position, and then a score for those features are extracted from each resume. Then an algorithm designed around Minkowski distance ranks the resumes to their specific position.

2.3 Salary Prediction Research

While finding the right candidate for a job position is important, the ability to forecast the salary for a position has been a long researched problem. In the past, salary projections have been important for projecting employees' pension and was done with a function on inflation and merit [11]. More recently, the salary prediction has been studied for different reasons. Some companies have even gone so far as to make a business model of predicting a customer's potential salary based off their resume [12].

A machine learning and data science website called Kaggle held a competition to predict salaries given job advertisements from the United Kingdom. One research team published their findings [2]. The team used an assortment of supervised regression methods, including maximum-likelihood estimation, lasso regression, feed-forward neural networks, and random forests. After cross-validation and performance comparison, the research team found that the random forest model performed the best at predicting salaries. Another set of researchers continued this work from Jackman and Reid for predicting salaries. This group improved the preprocessing steps and then compared the random forest classification against decision tree classification to determine that the random forest classification more accurately predicted salaries [13].

2.4 Knowledge Graphs and Semantic Networks

ConceptNet is a semantic network designed to represent common sense knowledge of words, the concept of a word, and assertions of how words can relate to each other. The knowledge represented in ConceptNet comes from a variety of sources. Initially, the information came from The Open Mind Common Sense project, but later sources were added to include games with a purpose, information from Wiktionary (*en.wiktionary.org*) to support multiple languages, a lexical database called WordNet, semantic connections via DBPedia, and relational statements from Wikipedia's free text using ReVerb. Past versions of ConceptNet have been used in previous applications and projects such as analyzing the emotional content of a text, creating a dialog system for improving software specifications, and creating public information displays. ConceptNet Numberbatch is also a part of the project, and is a set of word embeddings created from ConceptNet, and can be used as input for many machine learning techniques including neural networks [14].

2.5 Naive Bayes Classifiers

The NBC is a common choice for text-based classification problems. A NBC has shown that it can outperform several text-based classifiers (such as decision tree, neural network, and support vector machine) in multiclass problems [15]. This research used an algorithm called Cfs Subset Evaluator and rank search to improve the feature selection when creating the NBC. Research on improving the classification of news articles by enriching the training set with data from Wikitology also shows performance increases [16]. This research used related titles of the news articles and related Wikipedia article title information to augment the training set. Trying to improve on the NBC for multiclass data sets is also an area of research. [17] shows that document length normalization is effective for text classification.

III. Methodology

This chapter provides a detailed description of the data sets, techniques, models, and evaluation employed in this experiment. The chapter presents both of the data sets involved in the research, discusses the comparative models of the experiment, and explores the model produced as part of this thesis. Finally, the model checking and comparison tests employed in this research are covered.

3.1 Objective

Most text classification models attempt to classify job descriptions by leveraging only words found in their training sets. There are semantic networks, like ConceptNet, that diagram the relationships between most known English words. This research aims to demonstrate that a model generated leveraging semantic information will more accurately predict a job posting category.

There are a few different reasons leveraging information from a semantic network may improve the results of a text-based classifier. Providing a classifier with synonyms and related words to those in the training set could give the classifier more information to make its prediction. Words can also have different meanings, or senses, in different professions. For example, the word shift can refer to a gear shift to a mechanic, a bit shift operation to a computer scientist, or a nurse's work shift. Another example is a plumber working with a pipe and a pastry chef piping icing. A word possessing different senses could change how a classifier treats it. If the classifier encounters the work more times in the training set's specific category, it could assign more importance to that word for that category.

This research will use an NBC and supplement the words in the training set with words from ConceptNet. This thesis uses the NBC as the model to supplement because of how it makes its predictions. For each category in the NBC, there is a multinomial distribution that contains each word's chance of belonging to that category. The NBC then uses these multinomial distributions to estimate the category for a job description. Words selected from ConceptNet will supplement each categorical distribution in one of two different ways. First, the word added from ConceptNet could be a word that was not previously in the categorical distribution. The second way ConceptNet supplements the NBC is by increasing the probabilities of words belonging to specific categories in the multinomial distributions. Finally, for each job description in the test set, Maximum Log-Likelihood is used to determine the job description's likely category. The NBC's performance is compared to several other models generated without semantic information. The comparison models are a K-Nearest Neighbor (KNN), Random Forest (RFC), and feed-forward Neural Network (NN) based classifiers.

3.2 Naive Bayes Classifier

The model used in this experiment to demonstrate the effect of leveraging semantic information is the NBC. NBCs are based on Bayes Theorem and assume that features are independent within a class. In this experiment, the NBC model's features are the number of times a word occurs in a category. The equation for a NBC is shown in eq. (1). Here the $\hat{y}$ is the predicted class, *argmax* is a function that returns the maximum value of a set of numbers, C_k represents a class in the data set, x_i is a feature from an entry in the data set, n is the number of features in a new feature vector to be classified, and k is a class in the data set.

$$\hat{y} = argmax\ p(C_k) \prod_{i=1}^{n} p(x_i|C_k), k \in \{1, ..., K\} \tag{1}$$

The $p(x_i|C_k)$ of the classifier is calculated for each word, or feature, in the training set. Not every word of the English language will have a probability for every class in the model. The true model parameters are latent variables. To estimate $p(x_i|C_k)$ in this thesis, we use word count. The fact that the data sets are labeled makes tallying the number of words straightforward for each class. The $p(x_i|C_k)$ for all words, x, are stored in a multinomial distribution. There is one multinomial distribution per class, henceforth referred to as a categorical. The NBC generates the categorical distributions leveraging both the training set keywords and semantic links defined in ConceptNet.

The following subsections will explain how the model leverages the semantic information and how the predicted class is selected.

3.2.1 Categorical Distribution Creation

As previously mentioned, the number of times each keyword appears in each class is known. The NBC requires a probability for each feature. To estimate each word's latent probabilities in every category, the model uses kernel density estimation (see equation 2). The kernel, K, used for this model is a Gaussian kernel, n is the number of features in the class, x is a feature, and h is a tunable smoothing parameter. This thesis uses a Gaussian kernel because it has demonstrated good performance in several other applications. The smoothing parameter, also known as the bandwidth, helps minimize spurious data artifacts. Undersmoothing results in a jagged distribution, while oversmoothing obscures the underlying distribution structure.

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2}$$

The additional semantic information extracted from ConceptNet will supplement the training set keywords. The added information will give a more extensive feature space to the categorical distributions for each class. Consider ConceptNet as a graph, where almost every word in the English language is a node. The edges of this graph are relations between words. Figure 1 shows an example of a few nodes and different edge types surrounding the node for ”job.”

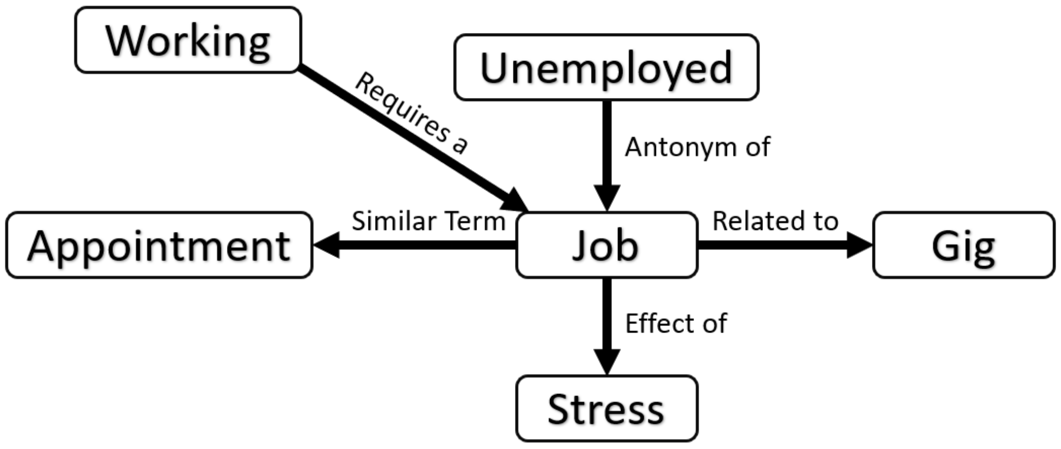


Figure 1: Example of Nodes and Edges from ConceptNet.

The additional semantic information added to the categorical distributions are words that share edges with the keywords from the training set. The idea is that creating a larger cluster of words related to each category will increase the NBC’s performance. Figure 2 is a representation of a subset of ConceptNet. Note that the next three figures is a fictional example of a subset of ConceptNet, and the number of keywords used is not to scale; the actual number of words in each distribution is in the results section.

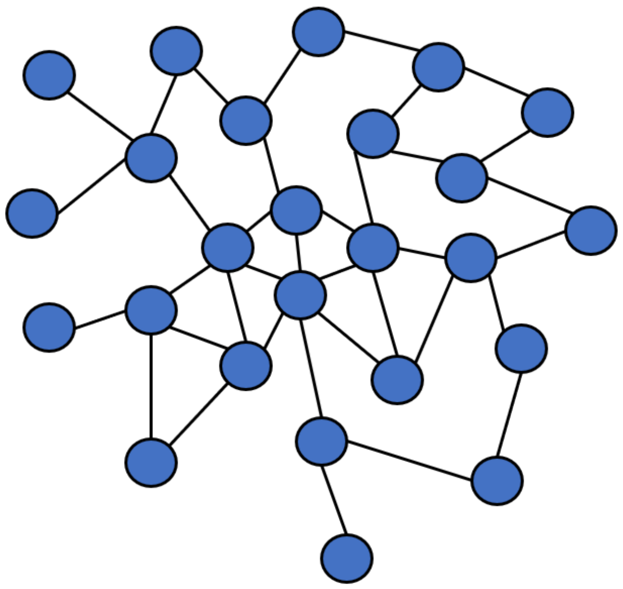


Figure 2: Visual Example of ConceptNet.

The categorical distribution starts with the keywords of each job description. The yellow nodes represent keywords in Figure 3.

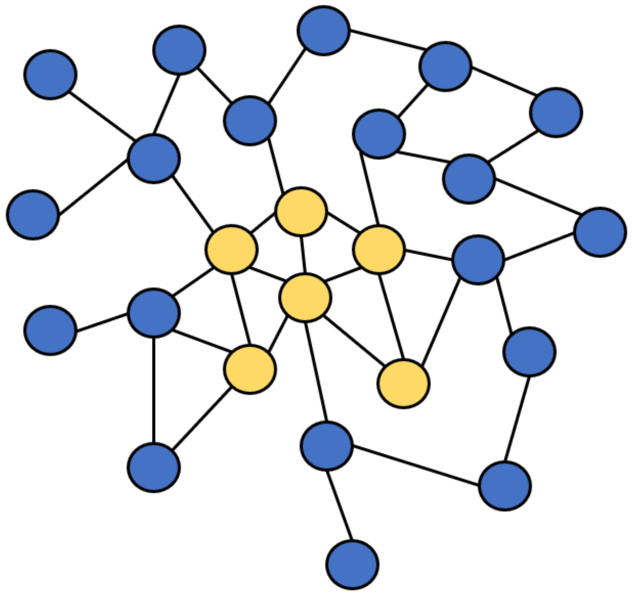


Figure 3: Visual Example of ConceptNet with Keywords.

The added semantic information are the nodes connected to the keywords in a categorical distribution. The green nodes in Figure 4 represent the nodes added to the categorical distribution of a job description extended by one edge. This extension happens multiple times for each job description in the training set.

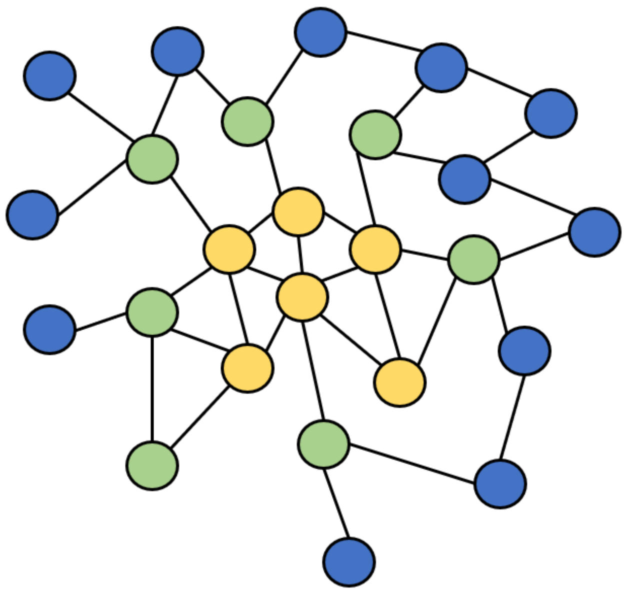


Figure 4: Visual Example of ConceptNet with Keywords and Tails.

ConceptNet also has an edge weight associated with each of the edges. Factors that affect the edge weight in ConceptNet are the edge's source and the number of sources. Per the ConceptNet documentation, "A typical weight is 1, but weights can be higher or lower. All weights are positive." This edge weight can be used as a criterion to filter out less useful edges to use in the model. A set number of edges, and the nodes connected to them, is extended out and added to the categorical distributions. This process repeats for every keyword in each category for every document in the training set. The algorithm that extends the nodes to create the categorical distribution will stop if it encounters a cycle.

Algorithm 1 shows how the generation of the categorical distribution for each category. The input is a list of the keywords for each job description in the training set.

The output is a data structure that contains each word in the category. Additionally, this data structure contains the number of times each word was a keyword, one edge out, two edges out, etc. This algorithm includes cycle checking for each keyword it extends.

Algorithm 1 Categorical Distribution Creation for one Category.

```
1: function GenCatDist($Category, NumEdges, MinEdgeWeight$)
2:     $Distribution \leftarrow \{\}$                    ▷ Create empty return dictionary
3:     for $Keywords$ in $Category$ do
4:         for $Word$ in $Keywords$ do
5:             $CycleCheck \leftarrow [\,]$              ▷ Create empty list for cycle checking
6:             append $Word$ to $CycleCheck$
7:             $CurrentEdgePosition \leftarrow 0$
8:             update $Distribution$ with $w$ and $CurrentEdgePosition$
9:             ExtendWord($Word, CurrentEdgePosition + 1$)
10:        end for
11:    end for
12:    return $Distribution$
13: end function
14:
15: function ExtendWord($Word, CurrentEdgePosition$)
16:    $EdgeList \leftarrow$ retrieve all edges over $MinEdgeWeight$ from ConceptNet
17:    for $Edge$ in $EdgeList$ do
18:        $EdgeWeight \leftarrow Edge.EdgeWeight$
19:        $ConnectedWord \leftarrow Edge.ConnectedWord$
20:        $NextLayer \leftarrow [\,]$                  ▷ Empty list to hold next layer of words
21:        if $EdgeWeight > MinEdgeWeight$ then
22:            if $ConnectedWord$ not in $CycleCheck$ then
23:                append $ConnectedWord$ to $CycleCheck$
24:                append $ConnectedWord$ to $NextLayer$
25:            end if
26:        end if
27:        for $w$ in $NextLayer$ do
28:            update $Distribution$ with $w$ and $CurrentEdgePosition$
29:            if $CurrentEdgePosition + 1 > NumEdges$ then
30:                ExtendWord($w, CurrentEdgePosition + 1$)
31:            end if
32:        end for
33:    end for
34:    return
35: end function
```

The NBC uses the output of Algorithm 1 to create the multinomial distribution for each category. The next step is to generate the input value for each word to the Kernel Density Estimator. Each word has a value that starts at zero. Every time a word is a keyword, the area of one standard deviation under a normal curve centered at zero is added to that word's value. The yellow node in 5 represents a keyword and shows the area under the curve that is added. Every time a word is one edge out from a keyword, the area under the curve from .5 to 1.5, or one standard deviation, is added to that word's value. The green node in 5 represents a word one edge out from a keyword. The value added to each word is a static value for the NBC model, and is tunable. The process repeats for the number of edges extended, which is a tunable parameter in the model. The input to the Kernel Density Estimator is the value for each word to create each word's probabilities in the multinomial distribution. The probabilities are normalized for each class.

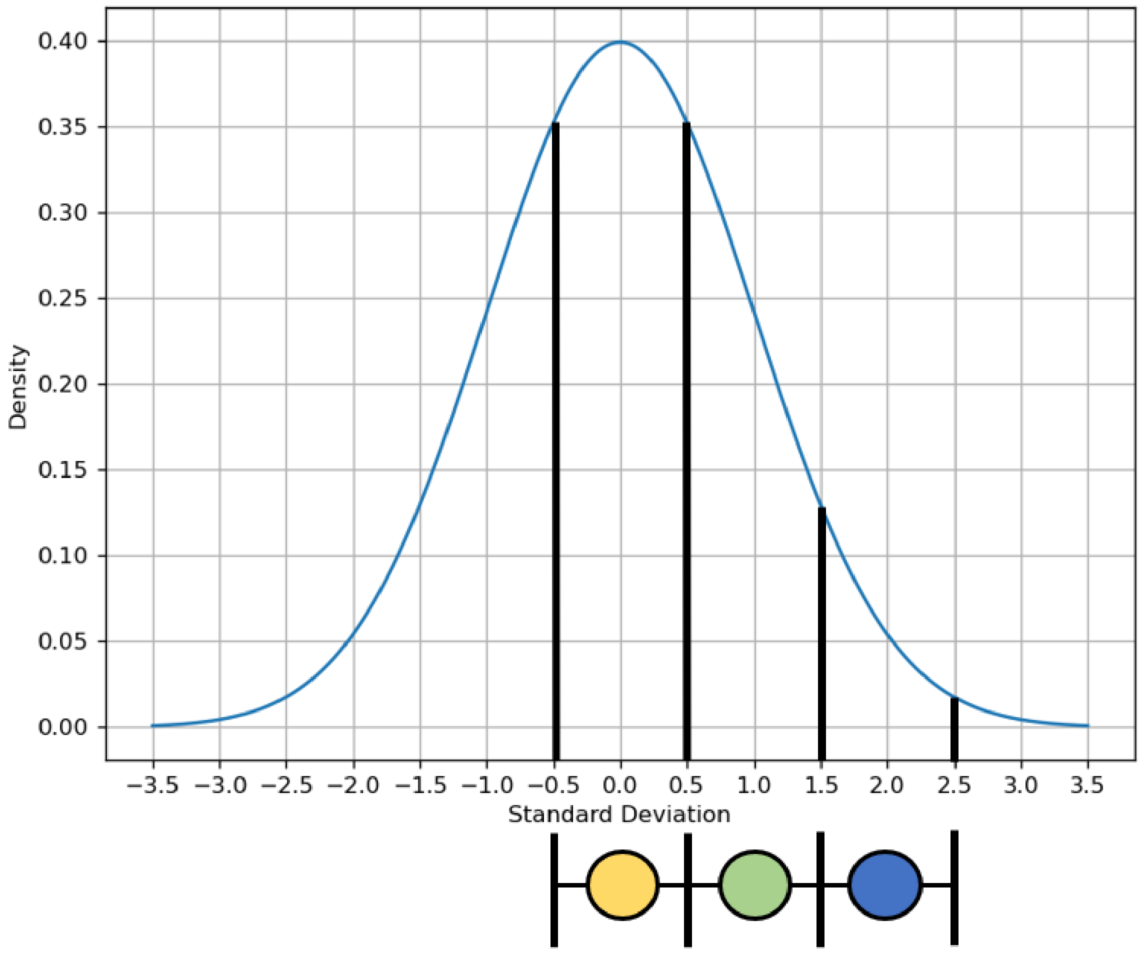


Figure 5: Visual Example of Node Probability.

A Dynamic Naive Bayes Classifier was made for this experiment to examine how a dynamic value instead of a static value would affect model performance. The dynamic value is affected by the edge weight that connects the two words. A higher edge weight will assign a higher value to the connected node's value. The parameters for the rest of the Dynamic Naive Bayes Classifier are the same as the standard NBC. This model also leverages semantic information from ConceptNet. The code for this model can be found in appendix A.

3.2.2 Final Class Selection

Once the model generates the categorical distributions using the training data, the model can predict an unknown job description's category. At this point, the latent probabilities for each word in the categorical distributions are tiny. To get the joint probability density of a job description requires calculating the product of small probabilities. The issue with this approach is computers lack the precision to handle such small numbers adequately. To avoid this problem, we use the Maximum Log-Likelihood, equation 3, instead. The category with the largest value from equation 3 is the NBC model's output class.

$$\hat{y} = argmax \sum_{i=1}^{n} \log(p(x_i|C_k)), k \in \{1, ..., K\} \tag{3}$$

The whole point of leveraging ConceptNet was to ensure the categorical distributions contain as many words as possible that pertain to its category. However, there will be instances of a word in the test set not existing in a categorical distribution for a class. When calculating the Maximum Log-Likelihood, the model needs a value for $p(x_i|C_k)$ when a word in the test set is not in a categorical distribution. A technique called Laplace Smoothing, or additive smoothing, is used to solve the lack of $p(x_i|C_k)$, as shown in eq. (4). Laplace Smoothing provides the model a value to use for $p(x_i|C_k)$ when it encounters a word in the test set that is not in a categorical distribution. In this equation x is the word, α is the smoothing parameter, N is the number of trials in the distribution, and d is the number of features in the distribution.

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d}, (i = 1, ..., d) \tag{4}$$

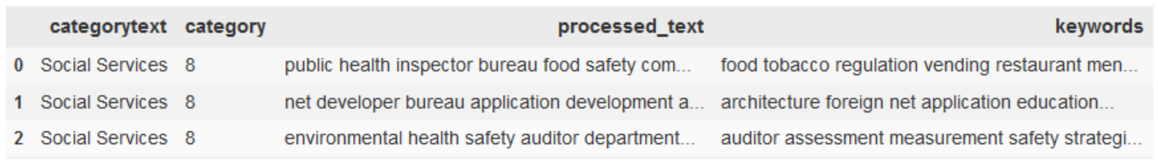
At this point, all possible parameters for the NBC models have been explained. Table 1 shows the values actually used for all parameters used in each of the NBC models for both data sets.

Table 1: Parameter Values used for the Naive Bayes Classifier.

Parameter	***NYC Data Set***	***Kaggle Data Set***
keywords	23	20
min edge weight	3	4
KDE bandwidth	1.8	0.7
standard deviation width	1	2.5
Laplace Lambda	0.001	0.0578

3.3 Data Sets

The experiment in this thesis uses two different data sets. The first is a government job posting data set gathered in New York City on 22 December 2019 [18]. This data set was obtained as a comma-separated value file and manipulated via Python. The data has 1613 entries with 28 unique features, though the model only uses the job category, business title, and job description features. There are a total of 12 classes of job postings available in this dataset, shown in Table 2. The un-processed data set allows entry to be a combination of the available categories. Entries can have more than one category because the input allowed for multiple inputs for the category field. During the experiment, we selected the first category listed as the truth category for that entry.



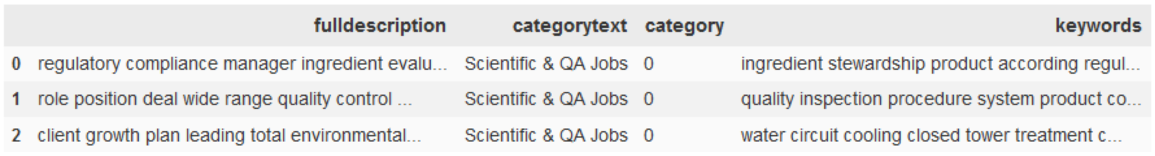
	categorytext	category	processed_text	keywords
0	Social Services	8	public health inspector bureau food safety com...	food tobacco regulation vending restaurant men...
1	Social Services	8	net developer bureau application development a...	architecture foreign net application education...
2	Social Services	8	environmental health safety auditor department...	auditor assessment measurement safety strategi...

Figure 6: Processed NYC Data Set.

Table 2: NYC Data Set: Total Category Sizes.

Category Name	*Total Category Size*
0. Administration & Human Resources	178
1. Building Operations & Maintenance	113
2. Clerical & Administrative Support	15
3. Communications & Intergovernmental Affairs	31
4. Constituent Services & Community Programs	113
5. Engineering, Architecture, & Planning	348
6. Finance, Accounting, & Procurement	152
7. Health	152
8. Information Technology & Telecommunications	229
9. Legal Affairs	142
10. Policy, Research & Analysis	83
11. Public Safety, Inspections, & Enforcement	103

The second data set comes from a Kaggle contest held in 2013. The raw data set consists of job postings provided by a job search engine company called Adzuna. All job postings in this data set originate from the United Kingdom. Figure 7 is a snippet of the Kaggle Data Set after processing. The Kaggle Data Set used in this thesis is a subset of the entire data set. This thesis uses a subset to ensure the categories were unique enough, to remove a catch-all category the data set had, and ensure the categories were similar in size. Table 13 shows the categories and number of entries in each category. This thesis's models only use the job title, text description, and category features from this data set.



	fulldescription	categorytext	category	keywords
0	regulatory compliance manager ingredient evalu...	Scientific & QA Jobs	0	ingredient stewardship product according regul...
1	role position deal wide range quality control ...	Scientific & QA Jobs	0	quality inspection procedure system product co...
2	client growth plan leading total environmental...	Scientific & QA Jobs	0	water circuit cooling closed tower treatment c...

Figure 7: Processed Kaggle Data Set Example.

Table 3: Kaggle Data Set: Total Category Sizes.

Category Name	*Total Category Size*
0. Scientific & QA	2489
1. PR, Advertising & Marketing	8854
2. Legal	3939
3. HR & Recruitment	7713
4. Charity & Voluntary	2332
5. Social Work	3455
6. Creative & Design	1605
7. Energy, Oil, & Gas	2255
8. Travel	3126
9. Manufacturing	3765
Total Entries	39533

3.3.1 Preprocessing

The text for each job posting is preprocessed before being used in the models for this experiment. Both of the data sets used in this experiment received the same preprocessing steps, and each model used in the experiment uses these steps. The preprocessing for this thesis is different than the preprocessing done in the related works. First, the preprocessing script merges each entry title and full description text. These feature names are different between the data sets, but the contents are the same. The following steps occur on this combined text: HTML tag removal, punctuation removal, tokenization, stop word removal, lemmatization, removing duplicate entries, and removing non-English words. The preprocessing also removes any words not contained in the NLTK English word dictionary. The processed text is then output to a new comma-separated value file for further manipulation.

The experiment is a multiclass classification problem that does not contain any way to produce new data entries for validation. Additionally, the number of entries in the classes of both data sets is imbalanced. Stratified K-Fold cross-validation is employed to overcome those two issues. There will be five folds for each trial of the experiment. Python scripts will create these K-Fold subsets of the data sets before use in any of the models. A random state parameter is also provided in the python script to ensure the experiment results are reproducible.

3.3.2 Keyword Extraction

The raw input for both data sets incorporates a description of the job and the job title. The description of the job is an unstructured block of text. The preprocessing cleans up this text, but there is still an unequal amount of words for each job description. For the NBC model in this experiment, the number of input words needs to be uniform. The uniformity ensures each document in the training set affects the model the same amount. The model employs a term frequency-inverse document frequency (TF-IDF) technique to extract the n-most significant words from a document. In this thesis, a document is one job posting.

Given a word w, a corpus D, and a document contained in the corpus $d \in D$, we can generate the TF-IDF value,

$$w_d = f_{w,d} * \log(|D|/f_{w,D}) \tag{5}$$

for all words in a document. The $f_{w,d}$ is the number of occurrences in which w appears in the document d, and $f_{w,D}$ is the number of occurrences in which w appears in the corpus D.

The model in this experiment uses a TF-IDF implementation from the Scikit-learn python package. A TF-IDF value is created for every word in a document. The words are then ranked in descending order. The top n words (where n is a tunable parameter) are then selected as keywords for that document. This process is done to the training and test sets separately, so the test set keywords are not influenced by the training set. This keyword extraction is repeated for every fold in both the training and validation sets.

3.4 Comparison of Classification Models

Three different classification models were created to use as comparisons for the model in this experiment. The comparison classification models are a KNN, RFC, and a Feed Forward Neural Network classification model. The same preprocessing steps performed on the raw text for the data sets was used for these models. Additionally, stratified cross validation was also used. None of the semantic information from ConceptNet that was used in the Bayes Classifier was used in the training of any of these comparison models.

3.4.1 K-Nearest Neighbor Classification

The K-Nearest Neighbor model used in this experiment is the Scikit-Learn implementation called KNeighborsClassifier [19]. The model takes in TF-IDF vectors of each of the job postings. There is only one parameter in this experiment that is tuned: n_neighbors, which is the number of neighbors to use to generate the output of the model. The parameter tuning is done with two of Scikit-Learns cross validation optimizers, RandomizedSearchCV and GridSearchCV. The final k value used in the experiment for the NYC data set is 11 and 12 for the Kaggle data set.

3.4.2 Random Forest Classification

Random forest [20] based models have been shown to perform well in regression tasks for job postings [2]. The Scikit-Learn implementation of an RFC, the RandomForestClassifier object, was used in this experiment [19]. All words in the job posting, after preprocessing, were used as possible features in the decision trees of the model. There are multiple ways to represent the data set text as input for our model.

For this experiment TF-IDF vectors are used. Table 4 shows the model parameters that were tuned for this experiment. Table 5 shows the values used to generate the results for each of the data sets. Parameter tuning for this model was performed with two different Scikit-Learn tuners, RandomizedSearchCV and GridSearchCV. The randomized search is used first followed by the grid search. All code used to generate this model, to include parameter tuning, can be found in appendix A.

Table 4: Overview of the Tunable Parameters for the Random Forest Classifier.

Parameter Name	***Description***
n_estimators	Number of trees in the forest.
max_depth	Maximum depth of the tree.
min_samples_split	Minimum number of samples required to split an internal node.
min_samples_leaf	Minimum number of samples required to be at a leaf node.
bootstrap	Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

Table 5: Parameter Values used for the Random Forest Classifier.

Parameter	***NYC Data Set***	***Kaggle Data Set***
n_estimators	800	800
max_depth	50	100
min_samples_split	5	5
min_samples_leaf	1	1
bootstrap	False	False

3.4.3 Neural Network Classification

Neural network based classifiers have been used in various publications using different types of layers [2] [5]. The networks for this experiment were implented in Keras [21] with TensorFlow [22]. Multiple configurations of neural networks were tested for this experiment: feed forward networks with one hidden layer, multiple hidden layers, and a convolutional layer were tested. Additionally, three different types of input were tested: pretrained word embeddings, generating word embeddings, and a sparse matrix input. The final configuration for this experiment is a sparse matrix input with one hidden layer of 48 nodes. All code used to generate this model can be found in appendix A.

3.4.4 Ensemble Model Classifier

An Ensemble Model is the final comparison model. This model is a simple majority vote model made up of the NBC, Random Forest Classifier, and Feed Forward Neural Network. Each of the three models receives the same input, they each make their predictions, and the final prediction is the majority vote of the individual outputs. The NBC does leverage semantic information. All code used to generate this model can be found in appendix A.

IV. Results

4.1 Introduction

Model performance metrics will be presented as micro-average scores or F1-scores due to the class imbalance in the data sets. In this thesis, getting the correct results of all job postings and classes are equally important. There is no type of job positions that the Air Force wants to prioritize over the other. Macro-averaging would treat smaller classes' accuracy the same as larger classes, unfairly biasing the resulting score.

The following subsections present the results of both data sets with various model configurations. In each subsection, the results of the comparative models generated without semantic information will be given first, followed by the results of the models generated using semantic information. Section 4.2 has the data set 1, the NYC data set, results for all models. Section 4.3 has the second data set, the Kaggle data set, results for all models. Section 4.4 provides a comparison of results from the different data sets generated for this thesis.

4.2 NYC Data Set Results

This section presents the results for data set 1, the New York City job postings. The model results are split into two subsections: those that do not contain semantic information and those that do. Then the results of the two will be compared and discussed in a final subsection.

4.2.1 NYC Data Set Results - Models With No Semantic Information

First, this subsection discusses the F1-scores of the models with no semantic information. The models that do no include semantic information are the comparison models described in Chapter 3. Additionally, the results of an NBC created with the same steps outlined in Chapter 3 but without semantic information added is included for both data sets. The tables of F1-scores will also contain 95% confidence intervals of those scores.

Table 6: NYC Data Set: Model's Without Semantic Information F1-Scores & Confidence Intervals.

Model Name	*F1-Score*	*95% CI*
K-Nearest Neighbor	66.9	[66.4, 67.5]
Random Forest Classifier	74.5	[73.9, 75.1]
Feed-Forward Neural Network	75.5	[75.0, 75.9]
Naive Bayes Classifier	52.6	[52.0, 53.3]

As seen in Table 6 the FFNN and RFC classifiers perform the best with F1-scores of 75.5 and 74.5. The NBC not leveraging semantic information in its categorical distributions performed significantly worse with an F1-score of 52.6.

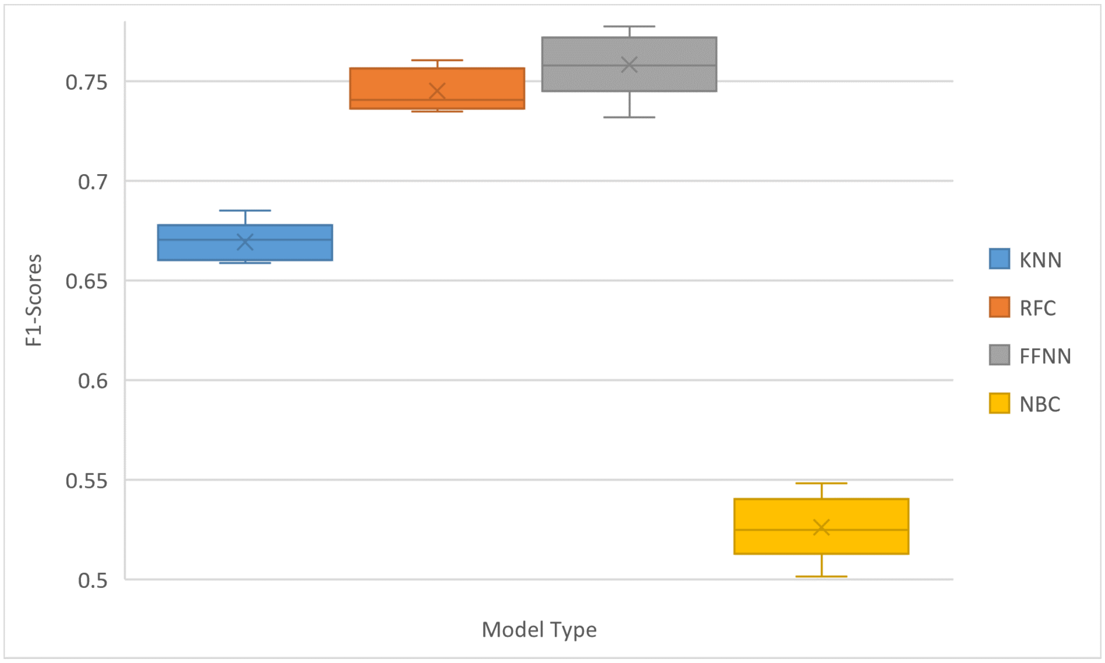


Figure 8: NYC Data Set: No Semantic Information F1-Score Box and Whisker Plot.

Figure 8 shows the F1-scores for each of the models in a box and whisker plot. The tight spread of F1-scores shows that the results for all of the models are consistent over the different test sets.

Table 7: NYC Data Set: Test Set Category Sizes.

Category Name	***Category Test Size***
0. Administration & Human Resources	38
1. Building Operations & Maintenance	25
2. Clerical & Administrative Support	3
3. Communications & Intergovernmental Affairs	7
4. Constituent Services & Community Programs	25
5. Engineering, Architecture, & Planning	72
6. Finance, Accounting, & Procurement	32
7. Health	32
8. Information Technology & Telecommunications	49
9. Legal Affairs	30
10. Policy, Research & Analysis	19
11. Public Safety, Inspections, & Enforcement	23

Table 7 shows each category in the NYC Data Set and the size of its test sets.

Table 8: Data Set 1: F1-Score by Category for Models without Semantic Information.

Class	***Support***	***KNN***	***RFC***	***FFNN***	***NBC***
0	38	51.2	59.3	61.8	36.5
1	25	67.4	79.8	75.9	45.0
2	3	0	0	1.7	0
3	7	14.4	19.4	26.4	6.1
4	25	52.1	59.4	64.8	38.0
5	72	78.7	83.2	86.8	72.1
6	32	65.9	75.0	77.2	59.4
7	32	74.2	79.4	75.7	64.7
8	49	76.6	83.1	87.4	64.7
9	30	69.0	79.9	81.1	58.6
10	19	38.1	49.7	52.1	34.9
11	23	53.3	67.5	66.4	54.5

Note: KNN: K-Nearest Neighbor ; RFC: Random Forest Classifier ; FFNN: Feed-Forward Neural Network ; NBC: Naive Bayes Classifier

Table 8 shows the average F1-score by category for the models with no semantic information. One thing to note is that the number of job postings for the classes' training and test sets are proportional. For example, the smallest class in the training set is also the smallest class in the test set. The NBC model produces lower F1-scores across the categories, aligning with its lower overall F1-score.

The smaller categories tend to have lower F1-scores. The lower score happens because the training set has fewer words to build the models. The RFC model performs better because it uses the same word multiple times in multiple decision trees. Even if one decision tree makes an incorrect prediction for a category, hundreds of other trees in the model could still lead to a correct prediction. The NBC model only has the words from the training set to build its categorical distributions. This smaller data set leads to a worse performance from the NBC.

4.2.2 NYC Data Set Results - Models that Leverage Semantic Information

This subsection presents the results of the models that leverage semantic information. Table 9 shows the F1-scores and 95% confidence intervals for the models leveraging semantic information. Figure 9 is a box and whisker plot of the F1-scores for these models.

Table 9: NYC Data Set: F1-Scores & Confidence Intervals for Models Leveraging Semantic Information.

Model Name	***F1-Score***	***95% CI***
Naive Bayes Classifier	69.1	[68.2, 70.1]
Dynamic Naive Bayes Classifier	68.4	[67.4, 69.5]
Ensemble Model Classifier	75.5	[74.9, 76.2]

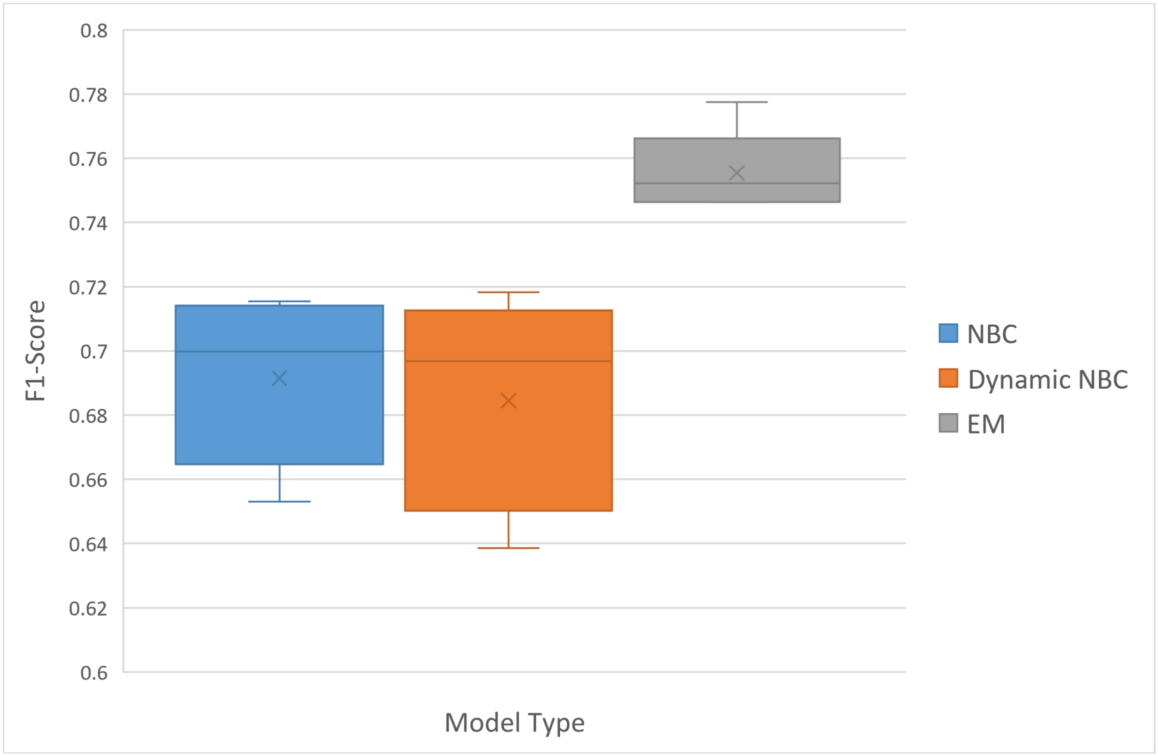


Figure 9: NYC Data Set: F1-Score Box and Whisker Plot for Models with Semantic Information.

Table 9 shows that the EM performs significantly better than the two NBC models. Figure 9 again shows a small F1-score range for each model, showing that these models are consistent over the different training/test folds for this data set.

Table 10: NYC Data Set: F1-Scores by Class for Models Leveraging Semantic Information.

Class	*Support*	*NBC*	*DNBC*	*EM*
0	38	48.9	47.5	60.3
1	25	73.1	71.2	78.6
2	3	0.0	7.8	0.0
3	7	33.6	29.8	27.2
4	25	54.7	55.8	65.7
5	72	84.1	83.4	84.2
6	32	68.5	67.2	76.3
7	32	74.0	72.0	78.9
8	49	82.2	82.1	86.3
9	30	73.9	73.9	80.0
10	19	47.3	44.4	49.2
11	23	65.1	69.3	69.3

Note: NBC: Naive Bayes Classifier ; DNBC: Dynamic Naive Bayes Classifier ;
EMF1: Ensemble Model

Table 10 shows the average F1-score by category for the models leveraging semantic information. The F1-scores for each category are similar for each of the models. The largest difference is in category 0 with a spread of 11.4%. The F1-scores in this data set do not follow a uniform ratio of class size and larger F1-scores. The smallest category has the worst F1-score due to its size. The remainder of the categories do not display a linear relationship between class size and the F1-score. This non-linear relationship could be due to the small size of this data set. There are not enough words in the models, and some words are better at determining a job description's class than others, leading to the relationship seen in this table. Appendix B contains all of the confusion matrices for these models.

4.2.3 NYC Data Set Results Summary

This subsection will provide an overview of the F1-scores for each model, then an analysis of the results. As seen in Table 11 the EM, FFNN, and RFC are the top-performing models for the first data set. The EM does leverage semantic information but does not perform significantly better than the models that do not. The NBC that leverages semantic information does outperform the KNN model. There is also a significant difference between the NBC model leveraging semantic information and those that do not. The only difference in these models is that one supplements its categorical distribution with words from ConceptNet, and the other does not.

Table 11: NYC Data Set: All Models F1-Scores & 95% Confidence Intervals.

Model Name	***F1-Score***	***95% CI***
Ensemble Model Classifier	75.5	[74.9, 76.2]
Feed-Forward Neural Network	75.5	[75.0, 75.9]
Random Forest Classifier	74.5	[73.9, 75.1]
Naive Bayes Classifier with Semantic Information	69.1	[68.2, 70.1]
Dynamic Naive Bayes Classifier	68.4	[67.4, 69.5]
K-Nearest Neighbor	66.9	[66.4, 67.5]
Naive Bayes Classifier without Semantic Information	52.6	[52.0, 53.3]

4.3 Kaggle Data Set Results

This section presents results using a subset of the United Kingdom job posting drawn from the Kaggle data set. First, the model results are divided into two categories: those that do not contain semantic information and those that do. Next the results of the two are compared and discussed in the last subsection.

4.3.1 Kaggle Data Set Results - Models With No Semantic Information

This subsection discusses the F1-scores of the models with no semantic information. The tables of F1-scores will also contain 95% confidence intervals of those scores.

Table 12: Kaggle Data Set: Models With No Semantic Information F1-Scores & Confidence Intervals.

Model Name	***F1-Score***	***95% CI***
K-Nearest Neighbor	81.7	[81.6, 81.8]
Random Forest Classifier	83.3	[83.2, 83.4]
Feed-Forward Neural Network	83.1	[83.1, 83.2]
Naive Bayes Classifier	79.3	[79.2, 79.4]

As seen in Table 12 the RFC and FFNN models perform the best with F1-scores of 83.3 and 83.1, which are not statistically different. The NBC performed the worst out of the models with no semantic information with an F1-score of 79.3.

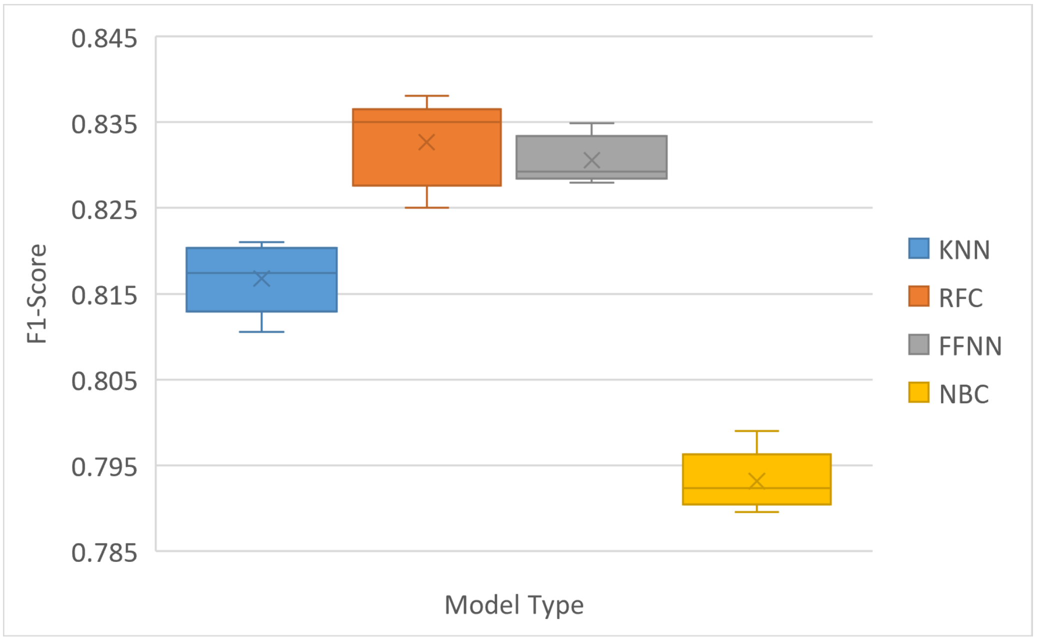


Figure 10: Kaggle Data Set: No Semantic Information F1-Score Box and Whisker Plot.

Figure 10 shows the F1-scores for each of the models in a box and whisker plot. Only the F1-scores of the RFC have a spread of more than 1%, and its' spread is less than 1.5%. The tight spread of F1-scores shows that the results for all of the models are consistent over the different test sets. Table 13 shows a listing of each category's names and sizes in the Kaggle Data set.

Table 13: Kaggle Data Set: Category Test Size.

Category Name	***Category Test Size***
0. Scientific & QA	501
1. PR, Advertising & Marketing	1774
2. Legal	791
3. HR & Recruitment	1545
4. Charity & Voluntary	468
5. Social Work	691
6. Creative & Design	321
7. Energy, Oil, & Gas	451
8. Travel	626
9. Manufacturing	753

Table 14: Kaggle Data Set: No Semantic Information F1-Score Table.

Class	***Support***	***KNN***	***RFC***	***FFNN***	***NBC***
0	501	72.5	75.1	74.7	72.0
1	1774	86.0	86.4	86.9	82.6
2	791	92.8	94.2	93.4	92.1
3	1545	86.8	87.0	88.3	83.9
4	468	66.0	68.8	68.2	66.1
5	691	83.4	85.5	84.3	83.1
6	321	34.7	29.3	45.0	40.4
7	451	73.1	78.4	76.9	73.0
8	626	88.4	90.0	89.1	86.8
9	753	75.9	79.5	79.5	76.4

Note: KNNR: K-Nearest Neighbor ; RFC: Random Forest Classifier ; FFNN: Feed-Forward Neural Network ; NBC: Naive Bayes Classifier

Table 14 shows each class's average F1-score for the models with no semantic information. One thing to note is that the number of job postings for the classes' training and test sets are proportional. For example, the smallest class in the training set is also the smallest class in the test set. Table 14 shows the smallest class, 'Creative Design,' has the worst F1-score for all four models. The larger classes all have higher F1-scores, specifically 'PR, Advertising, Marketing,' 'HR Recruitment,' and 'Legal.' The larger classes have a higher recall because the models benefit from having more words in the training set. This also explains why the 'Creative Design' class has the lowest F1-score.

Appendix B contains all of the confusion matrices for each of these trials. The order of the figures will be the same as presented in the table: KNN, RFC, FFNN, and finally NBC.

4.3.2 Kaggle Data Set Results - Models that Leverage Semantic Information

This subsection presents results of the models leveraging semantic information. Table 15 contains the F1-scores and the 95% confidence intervals for those scores. Figure 11 is a box and whisker plot of the F1-scores.

Table 15: Kaggle Data Set: Models Leveraging Semantic Information F1-Scores.

Model Name	***F1-Score***	***95% CI***
Naive Bayes Classifier	79.5	[79.5, 79.6]
Dynamic Naive Bayes Classifier	79.5	[79.4, 79.6]
Ensemble Model Classifier	84.7	[84.6, 84.8]

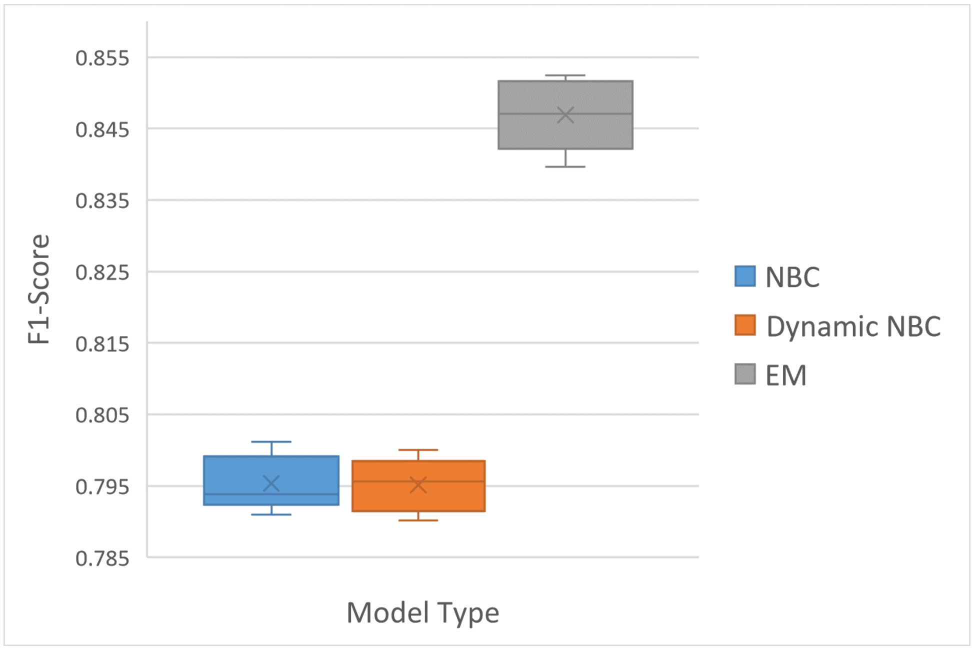


Figure 11: Kaggle Data Set: Box and Whisker Plot of Models leveraging Semantic Information.

As seen in Table 15 the EM outperforms the other models with an accuracy score of 84.7 by a statistically significant amount. The box and whisker plot in Figure 11 also demonstrates that each of these models' accuracy scores is very close to the mean. Again, this demonstrates that the models are consistent over different test sets for this data set.

Table 16: Kaggle Data Set: Models Leveraging Semantic Information F1-Scores.

Class	*Support*	*NBC*	*DNBC*	*EM*
0	501	72.1	72.3	77.2
1	1774	82.8	82.8	87.5
2	791	92.1	92.1	94.9
3	1545	84.0	84.0	88.3
4	468	66.5	66.6	71.8
5	691	83.2	83.3	86.3
6	321	40.3	40.5	42.4
7	451	73.1	73.0	80.7
8	626	87.0	86.9	90.9
9	753	76.3	76.3	81.4

Note: NBC: Naive Bayes Classifier ; DNBC: Dynamic Naive Bayes Classifier ; EM: Ensemble Model

Table 16 lists the F1-scores by category for the models leveraging semantic information. These results show the continuation of smaller classes having a lower F1-score compared to higher classes. Again, this trend is due to the larger classes having more words in the training set, leading to a higher F1-score. Appendix B contains all of the confusion matrices for these models.

4.3.3 Kaggle Data Set Results Summary

This subsection will provide an overview of the F1-scores for each model, then an analysis of the results. As seen in Table 17 the EM has the highest F1-score of

84.7, which is a model generated with semantic information. However, the models generated with no semantic information are close in accuracy score behind this model: specifically the RFC, FFNN, and KNN models. Finally, all variations of the NBC models performed the worst.

Table 17: Kaggle Data Set: All Models F1-Scores & 95% Confidence Intervals.

Model Name	***F1-Score***	***95% CI***
Ensemble Model Classifier	84.7	[84.6, 84.8]
Random Forest Classifier	83.3	[83.2, 83.4]
Feed-Forward Neural Network	83.1	[83.1, 83.2]
K-Nearest Neighbor	81.7	[81.6, 81.8]
Naive Bayes Classifier with Semantic Information	79.5	[79.5, 79.6]
Dynamic Naive Bayes Classifier	79.5	[79.4, 79.6]
Naive Bayes Classifier without Semantic Information	79.3	[79.2, 79.4]

The NBC models leveraging semantic information did not perform significantly better than the NBC model that did not. This result goes against the hypothesis that models generated leveraging semantic information will predict a job posting class more accurately than a model that does not. The following subsection analyzes the semantic information added to the NBC model.

4.4 Data Set Comparison

The best performing models in both data sets were the EM, FFNN, and RFCs. While the EM does leverage semantic information, there is no significant difference in these models' performance. This lack of difference in the best performing models provides evidence against the hypothesis of this thesis. The performance of the NBC models varies between the two data sets. The size of the data sets affects how much the semantic information affected the NBC models. The NBC leveraged more words when the training set was smaller, significantly improving the NBC's performance in data set 1. The training set in data set 2 was much more extensive, and leveraging ConceptNet did not significantly improve the performance results.

Table 18 shows each category's word statistics in the NYC Data Set. The table lists the number of words used in each category. It then lists the number of words leveraged from ConceptNet that were not keywords–i.e., words that do not appear in the training data. The next column expresses this statistic as a percent of total words. Finally, the number of keywords in each category is listed. Table 18 shows that semantic information makes up a significant amount of each category. The addition of semantic information to the NBC improves the model's F1-score by 16.5% for this data set. This performance improvement does support the hypothesis of this paper. However, the NBC model does not outperform other models not leveraging semantic information.

Table 18: NYC Data Set: Semantic Information Leveraged in NBC Models.

Class	***Total Words***	***Added Non-Keywords***	***Added$_{\%}$***	***Keywords***
0	2901	1416	48.8	1485
1	2119	1168	55.1	951
2	463	258	55.7	205
3	871	497	57.1	374
4	2234	1194	53.4	1040
5	2908	1186	40.8	1722
6	2332	1157	49.6	1175
7	2586	1371	53.0	1215
8	2929	1402	47.9	1527
9	2127	1063	50.0	1064
10	1766	980	55.5	786
11	1836	1004	54.7	832

We will then analyze the number of words added to the categorical distributions that are not keywords in the Kaggle Data Set. Table 19 provides a breakdown of the number of words in each class's categorical distribution. The table contains the total number of words in each class, the number of words added from ConceptNet that are not keywords, and the percentage of the added non-keywords over the total number of words. The last column contains the number of times a keyword was added to its categorical distribution.

Table 19: Kaggle Data Set: Semantic Information in NBC Model.

Class	***Total Words***	***Added Non-Keywords***	***Added$_{\%}$***	***Keywords***
0	6065	77	1.27	5988
1	7453	61	0.82	7392
2	4982	68	1.36	4914
3	6809	58	0.85	6751
4	4782	80	1.67	4702
5	4291	88	2.05	4203
6	5105	74	1.45	5031
7	4914	68	1.38	4846
8	5428	66	1.22	5362
9	5861	66	1.13	5795

Table 19 shows that the number of non-keywords added to each categorical distribution is a small fraction of the total number of words in the distribution. Keywords are reinforced in the categorical distributions each time they are encountered in a job description and during edge extension. This reinforces the values of keywords more instead of breaking out of the cluster of keywords and adding new non-keywords to the categorical distribution. Imagine the categorical distribution as a cluster of nodes on ConceptNet. When stepping multiple edges out from keywords, we are just traveling back into the cluster instead of expanding it. The NYC Data Set has much fewer keywords, so instead of traveling back into the cluster, it expands.

Table 20 contains the number of non-keywords used in any prediction in the test set. The table also includes the percentage of correctly predicted job postings that contain an added non-keyword.

Table 20: Kaggle Data Set: Added Semantic Information Actually Used.

Class	***Added Non-Keywords***	***Used Non-Keywords***	***Predicted***$_{\%}$
0	77	31	85.9
1	61	19	85.9
2	68	38	81.8
3	58	20	85.5
4	80	38	87.2
5	88	50	82.5
6	74	33	89.7
7	68	38	85.1
8	66	25	77.9
9	66	32	84.3

The goal of leveraging ConceptNet was to add enough new words to the categorical distributions to improve the model's F1-score. Rather than adding more non-keywords to the categorical distribution, current keywords are getting a more significant presence. This data set is more extensive and contains more keywords than data set 1 did before semantic information is added to the categorical distributions. Then ConceptNet does not augment the categorical distributions enough to see a performance increase in the NBC models. ConceptNet is too generalized to add enough words to affect performance when extending the tails of the categorical distributions for this larger data set.

V. Conclusion

5.1 Summary

This thesis aims to test the hypothesis that leveraging a semantic network can improve text-based classification. The experiment tested the hypothesis by creating a NBC leveraging the semantic information from ConceptNet to improve its performance. To test the NBC's performance, the model was compared to models that did not supplement their training set with any words from ConceptNet. The comparison models, implemented in Python, are a Random Forest classifier, K-Nearest Neighbor classifier, Neural Network classifier, and a NBC without semantic information. The results section compares the average F1-score of each model and average F1-score by category for each model and then takes an in-depth look at the value-added from the semantic network.

5.2 Research Findings

Among the top-performing text-based classifiers, this thesis found no statistically significant difference between a text-based classifier that leverages semantic information and one that does not. This evidence goes against the hypothesis that a text-based classifier leveraging semantic information will perform better than those that do not. Leveraging semantic information does improve the results of a NBC for smaller data sets. The NBC leveraged more words when the training set was smaller, significantly improving the NBC's performance in the NYC Data Set. The training set in the Kaggle Data Set was much more extensive, and leveraging ConceptNet did not significantly improve the performance results. The Ensemble Model, Neural Network-based classifiers, and Random Forest classifiers performed the best of the models tested.

5.3 Significance of Research

With limited data in the training set, leveraging semantic information did improve the NBC's results. However, when working with larger training sets, leveraging semantic information did not improve the NBC's performance. Further research exploring an Ensemble Model Classifier, Random Forest Classifier, or a Neural Network based classifier leveraging semantic information could produce performance improvements.

5.4 Future Work

There are several directions to take in predicting salaries given a job posting or resume. The results of this thesis only apply to text-based classification. The results do support future research in the following areas:

1. Extend the models to predict a salary range for a job posting. Future studies could implement a two-step ensemble model utilizing this research. The first step of the model would decide the category of the input job posting or resume. The second step would be a linear regression model specifically for that type of job that outputs a salary range.

2. Use a NBC leveraging semantic information to rank a pool of resumes given specific requirements or keywords. Instead of outputting the category of a document, modify the Naive Bayes model to give a score based on a categorical distribution. The pool of resumes would then be ranked, given their scores.

3. Explore leveraging semantic information in other types of models not covered in this thesis. For example, using the added words from a semantic network in a long short-term memory recurrent neural network (LSTM) [23].

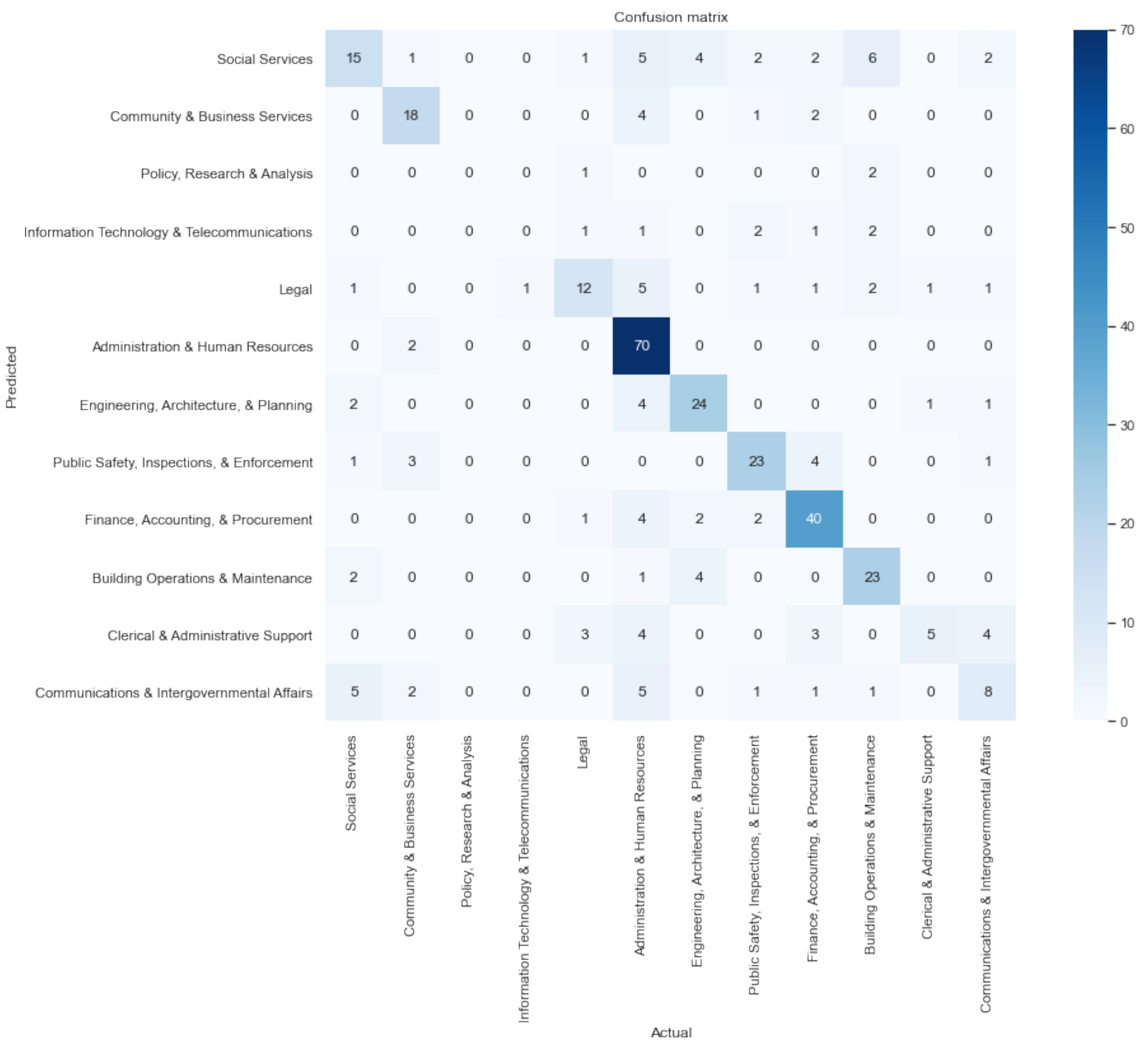
Appendix A. Code used in Thesis

To obtain access to the source code for this thesis please contact the author at joshua.h.white.51@gmail.com.

Appendix B. Confusion Matrices

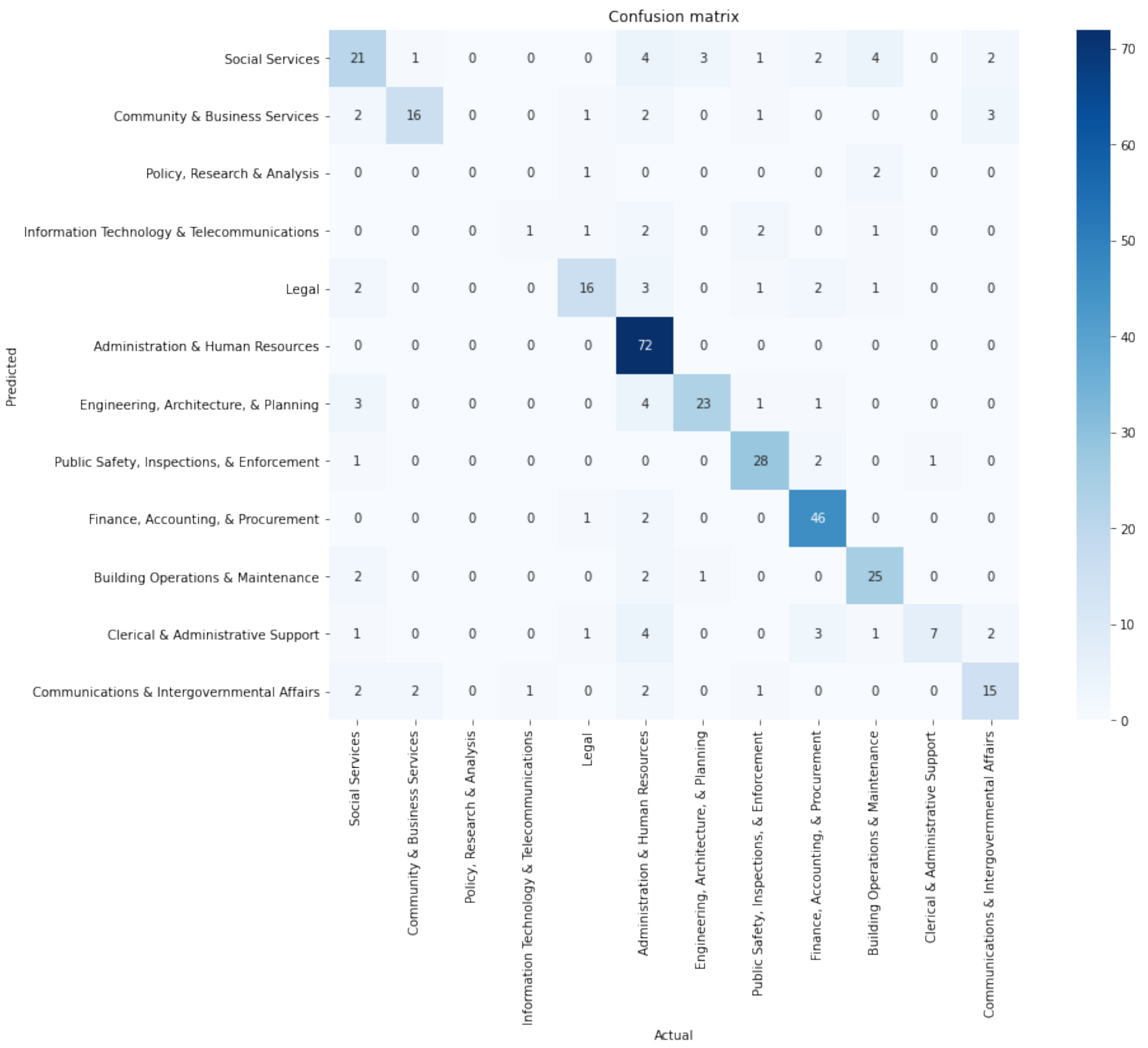
Below are the confusion matrices for the models in this thesis. The matrices for the NYC Data Set are presented first followed by the matrices for the Kaggle Data Set.

2.1 NYC Data Set Confusion Matrices



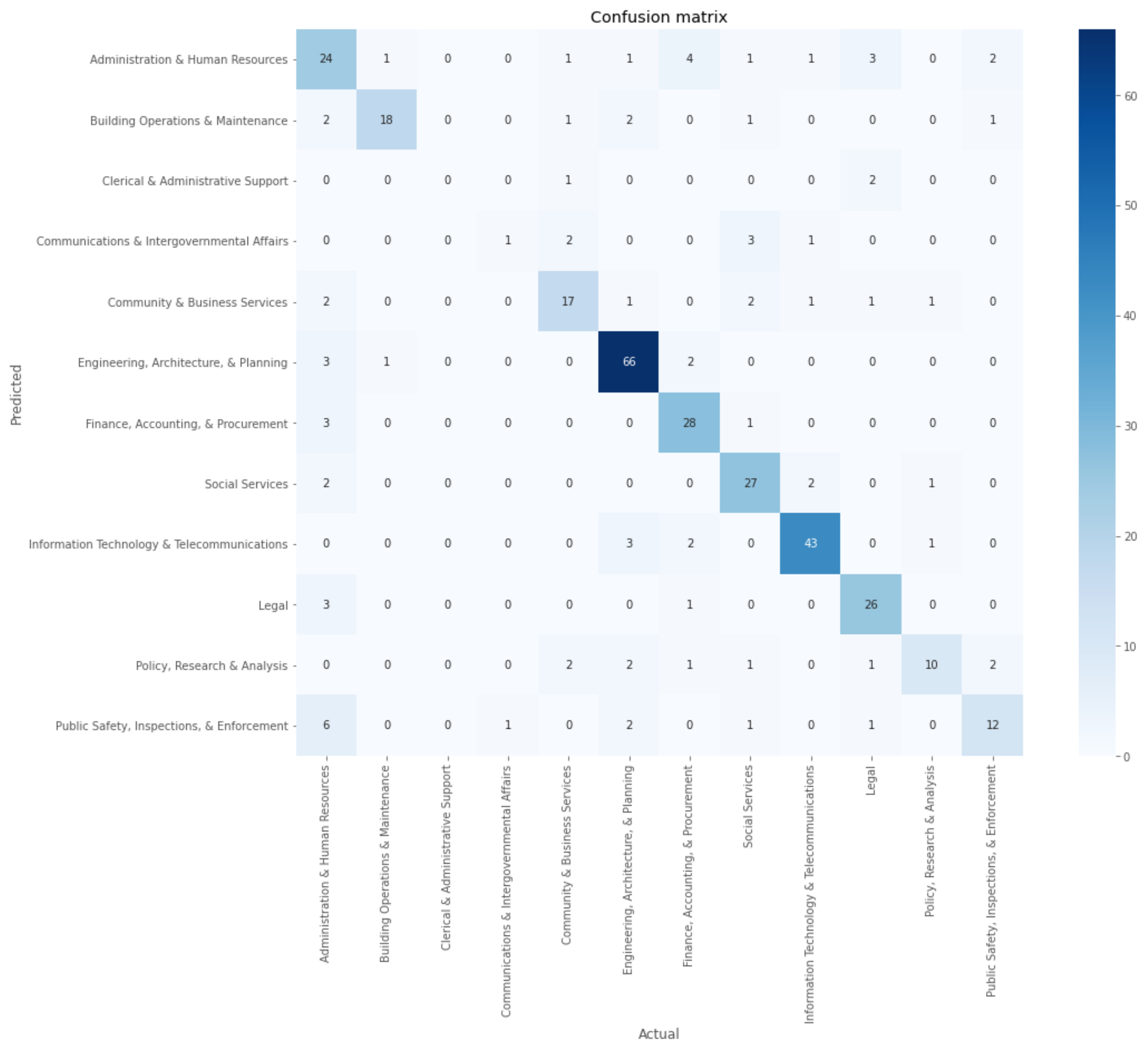
Predicted \ Actual	Social Services	Community & Business Services	Policy, Research & Analysis	Information Technology & Telecommunications	Legal	Administration & Human Resources	Engineering, Architecture, & Planning	Public Safety, Inspections, & Enforcement	Finance, Accounting, & Procurement	Building Operations & Maintenance	Clerical & Administrative Support	Communications & Intergovernmental Affairs
Social Services	15	1	0	0	1	5	4	2	2	6	0	2
Community & Business Services	0	18	0	0	0	4	0	1	2	0	0	0
Policy, Research & Analysis	0	0	0	0	1	0	0	0	0	2	0	0
Information Technology & Telecommunications	0	0	0	0	1	1	0	2	1	2	0	0
Legal	1	0	0	1	12	5	0	1	1	2	1	1
Administration & Human Resources	0	2	0	0	0	70	0	0	0	0	0	0
Engineering, Architecture, & Planning	2	0	0	0	0	4	24	0	0	0	1	1
Public Safety, Inspections, & Enforcement	1	3	0	0	0	0	0	23	4	0	0	1
Finance, Accounting, & Procurement	0	0	0	0	1	4	2	2	40	0	0	0
Building Operations & Maintenance	2	0	0	0	0	1	4	0	0	23	0	0
Clerical & Administrative Support	0	0	0	0	3	4	0	0	3	0	5	4
Communications & Intergovernmental Affairs	5	2	0	0	0	5	0	1	1	1	0	8

Figure 12: Confusion Matrix for the NYC Data Set K-Nearest Neighbor Classifier



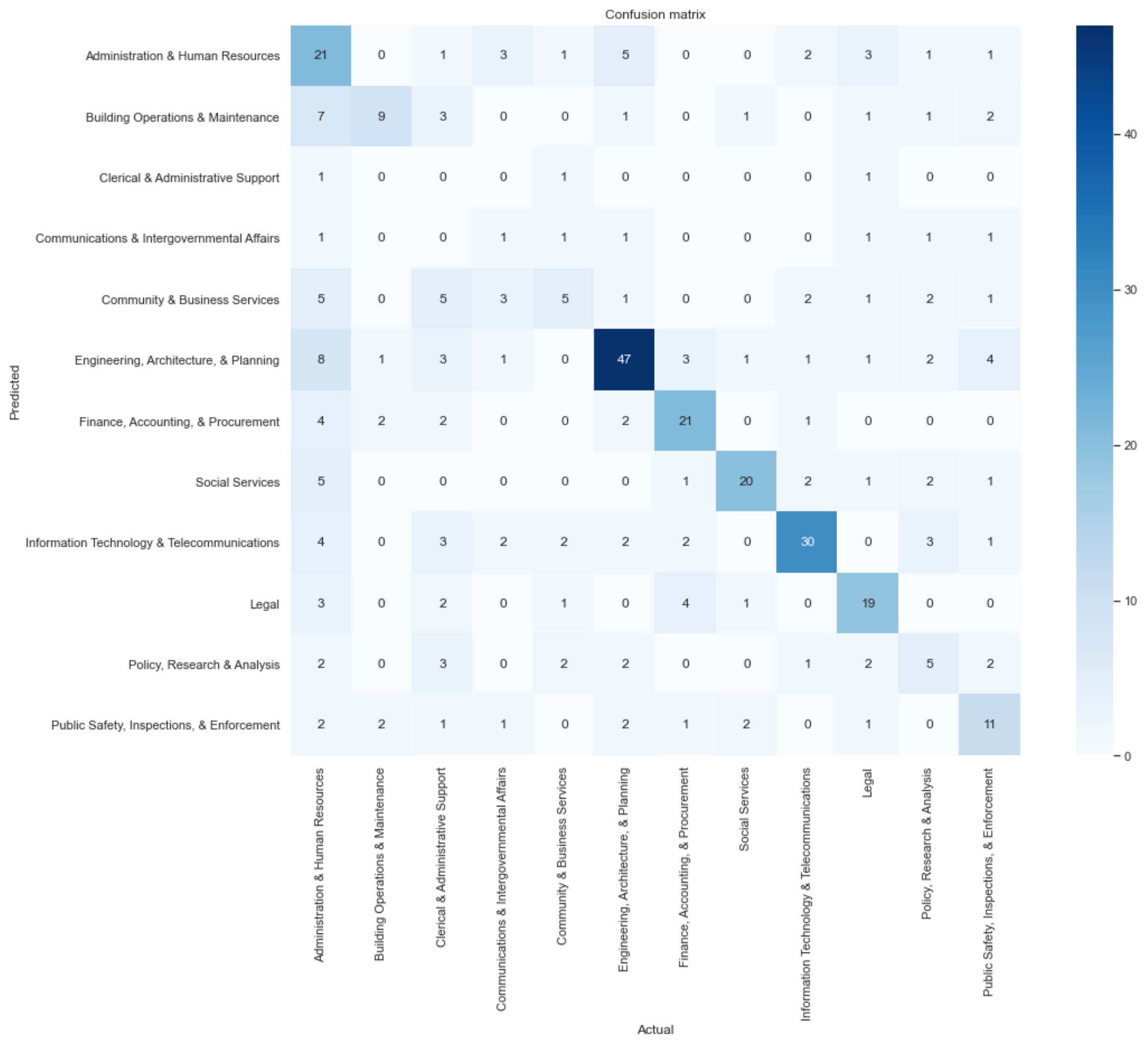
	Social Services	Community & Business Services	Policy, Research & Analysis	Information Technology & Telecommunications	Legal	Administration & Human Resources	Engineering, Architecture, & Planning	Public Safety, Inspections, & Enforcement	Finance, Accounting, & Procurement	Building Operations & Maintenance	Clerical & Administrative Support	Communications & Intergovernmental Affairs
Social Services	21	1	0	0	0	4	3	1	2	4	0	2
Community & Business Services	2	16	0	0	1	2	0	1	0	0	0	3
Policy, Research & Analysis	0	0	0	0	1	0	0	0	0	2	0	0
Information Technology & Telecommunications	0	0	0	1	1	2	0	2	0	1	0	0
Legal	2	0	0	0	16	3	0	1	2	1	0	0
Administration & Human Resources	0	0	0	0	0	72	0	0	0	0	0	0
Engineering, Architecture, & Planning	3	0	0	0	0	4	23	1	1	0	0	0
Public Safety, Inspections, & Enforcement	1	0	0	0	0	0	0	28	2	0	1	0
Finance, Accounting, & Procurement	0	0	0	0	1	2	0	0	46	0	0	0
Building Operations & Maintenance	2	0	0	0	0	2	1	0	0	25	0	0
Clerical & Administrative Support	1	0	0	0	1	4	0	0	3	1	7	2
Communications & Intergovernmental Affairs	2	2	0	1	0	2	0	1	0	0	0	15

Figure 13: Confusion Matrix for the NYC Data Set Random Forest Classifier



Predicted \ Actual	Administration & Human Resources	Building Operations & Maintenance	Clerical & Administrative Support	Communications & Intergovernmental Affairs	Community & Business Services	Engineering, Architecture, & Planning	Finance, Accounting, & Procurement	Social Services	Information Technology & Telecommunications	Legal	Policy, Research & Analysis	Public Safety, Inspections, & Enforcement
Administration & Human Resources	24	1	0	0	1	1	4	1	1	3	0	2
Building Operations & Maintenance	2	18	0	0	1	2	0	1	0	0	0	1
Clerical & Administrative Support	0	0	0	0	1	0	0	0	0	2	0	0
Communications & Intergovernmental Affairs	0	0	0	1	2	0	0	3	1	0	0	0
Community & Business Services	2	0	0	0	17	1	0	2	1	1	1	0
Engineering, Architecture, & Planning	3	1	0	0	0	66	2	0	0	0	0	0
Finance, Accounting, & Procurement	3	0	0	0	0	0	28	1	0	0	0	0
Social Services	2	0	0	0	0	0	0	27	2	0	1	0
Information Technology & Telecommunications	0	0	0	0	0	3	2	0	43	0	1	0
Legal	3	0	0	0	0	0	1	0	0	26	0	0
Policy, Research & Analysis	0	0	0	0	2	2	1	1	0	1	10	2
Public Safety, Inspections, & Enforcement	6	0	0	1	0	2	0	1	0	1	0	12

Figure 14: Confusion Matrix for the NYC Data Set Feed-Forward Neural Network Classifier



Predicted \ Actual	Administration & Human Resources	Building Operations & Maintenance	Clerical & Administrative Support	Communications & Intergovernmental Affairs	Community & Business Services	Engineering, Architecture, & Planning	Finance, Accounting, & Procurement	Social Services	Information Technology & Telecommunications	Legal	Policy, Research & Analysis	Public Safety, Inspections, & Enforcement
Administration & Human Resources	21	0	1	3	1	5	0	0	2	3	1	1
Building Operations & Maintenance	7	9	3	0	0	1	0	1	0	1	1	2
Clerical & Administrative Support	1	0	0	0	1	0	0	0	0	1	0	0
Communications & Intergovernmental Affairs	1	0	0	1	1	1	0	0	0	1	1	1
Community & Business Services	5	0	5	3	5	1	0	0	2	1	2	1
Engineering, Architecture, & Planning	8	1	3	1	0	47	3	1	1	1	2	4
Finance, Accounting, & Procurement	4	2	2	0	0	2	21	0	1	0	0	0
Social Services	5	0	0	0	0	0	1	20	2	1	2	1
Information Technology & Telecommunications	4	0	3	2	2	2	2	0	30	0	3	1
Legal	3	0	2	0	1	0	4	1	0	19	0	0
Policy, Research & Analysis	2	0	3	0	2	2	0	0	1	2	5	2
Public Safety, Inspections, & Enforcement	2	2	1	1	0	2	1	2	0	1	0	11

Figure 15: Confusion Matrix for the NYC Data Set Naive Bayes Classifier without Semantic Information

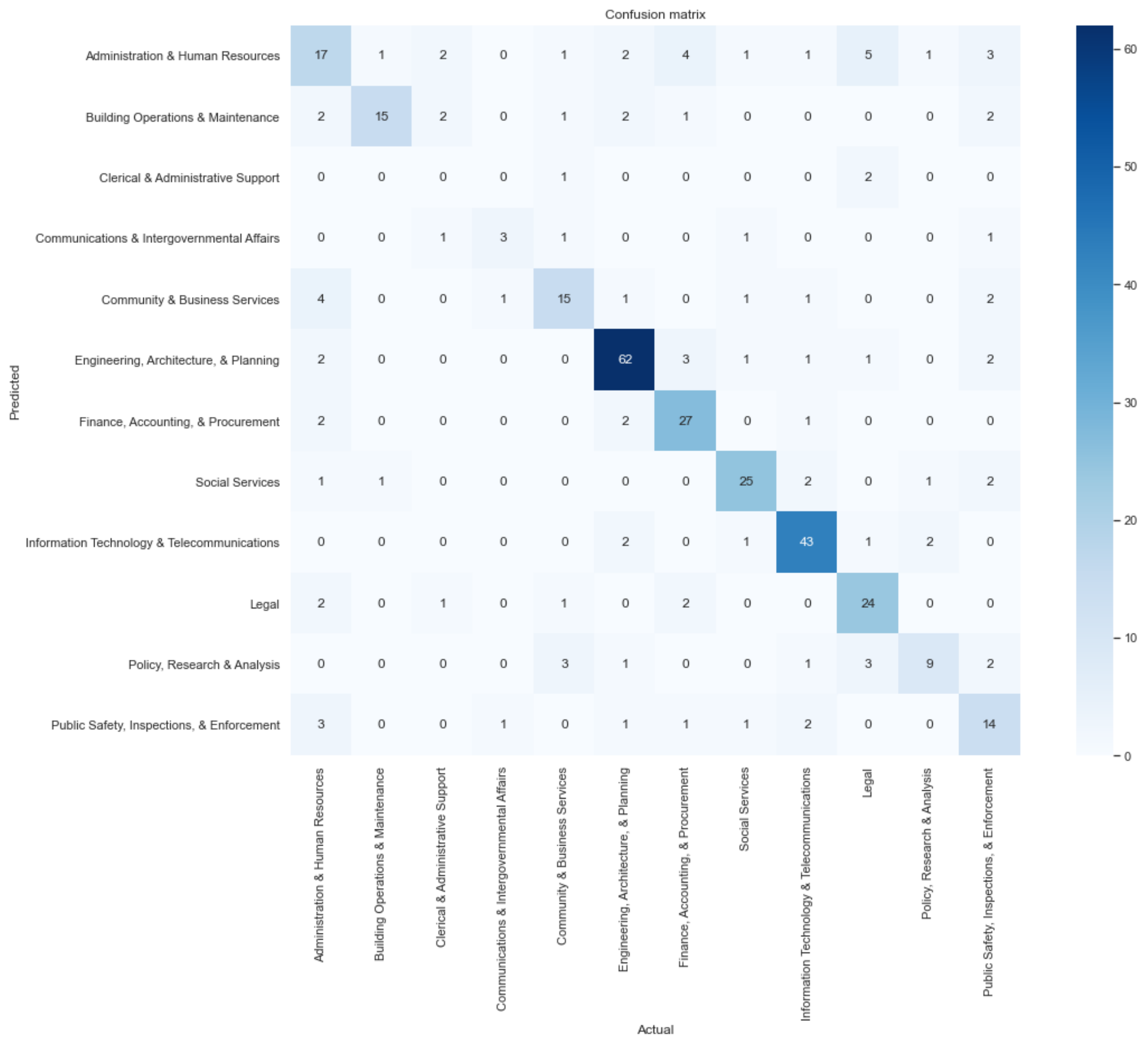


Figure 16: Confusion Matrix for the NYC Data Set Naive Bayes Classifier with Semantic Information

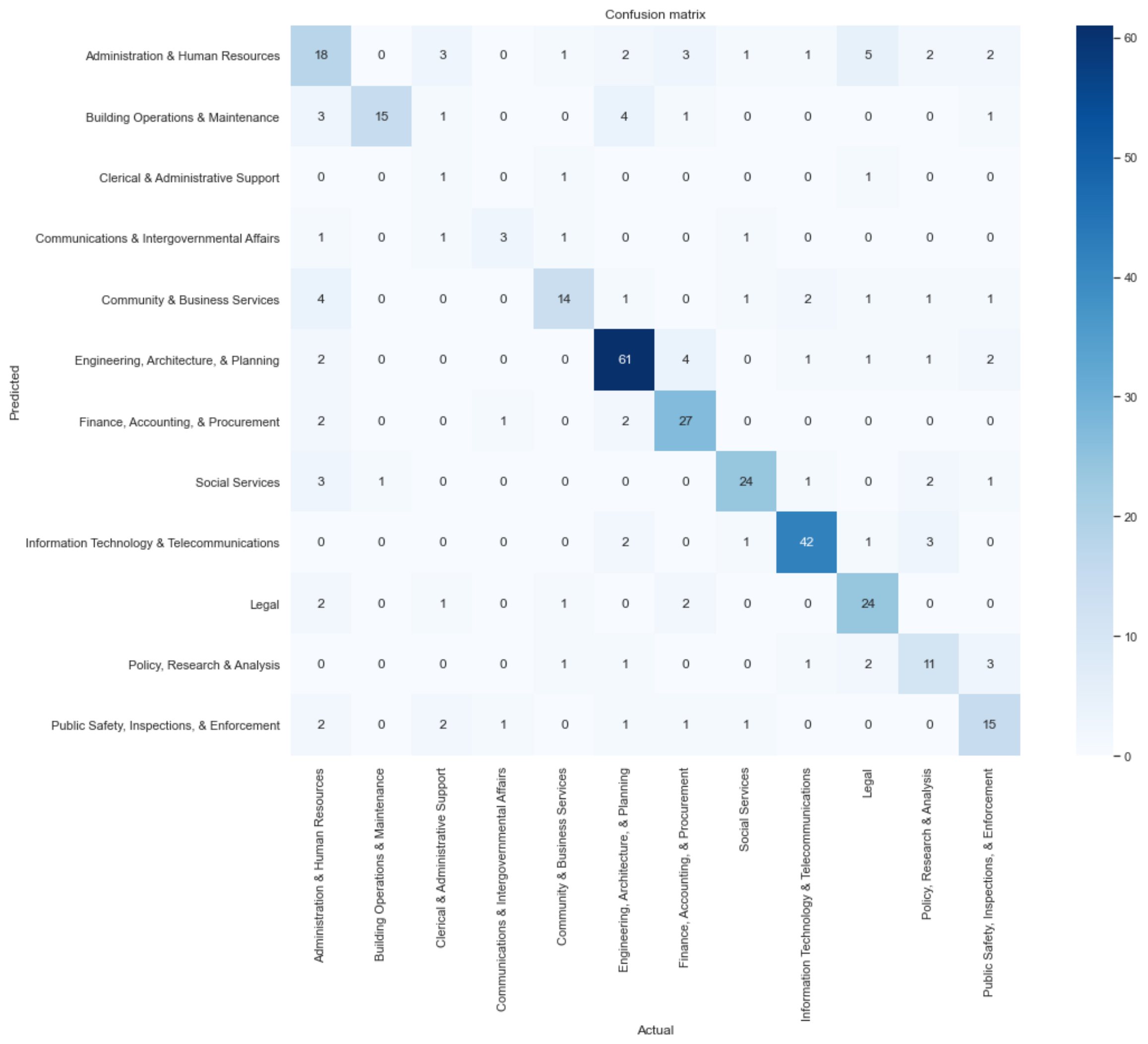


Figure 17: Confusion Matrix for the NYC Data Set Dynamic Naive Bayes Classifier

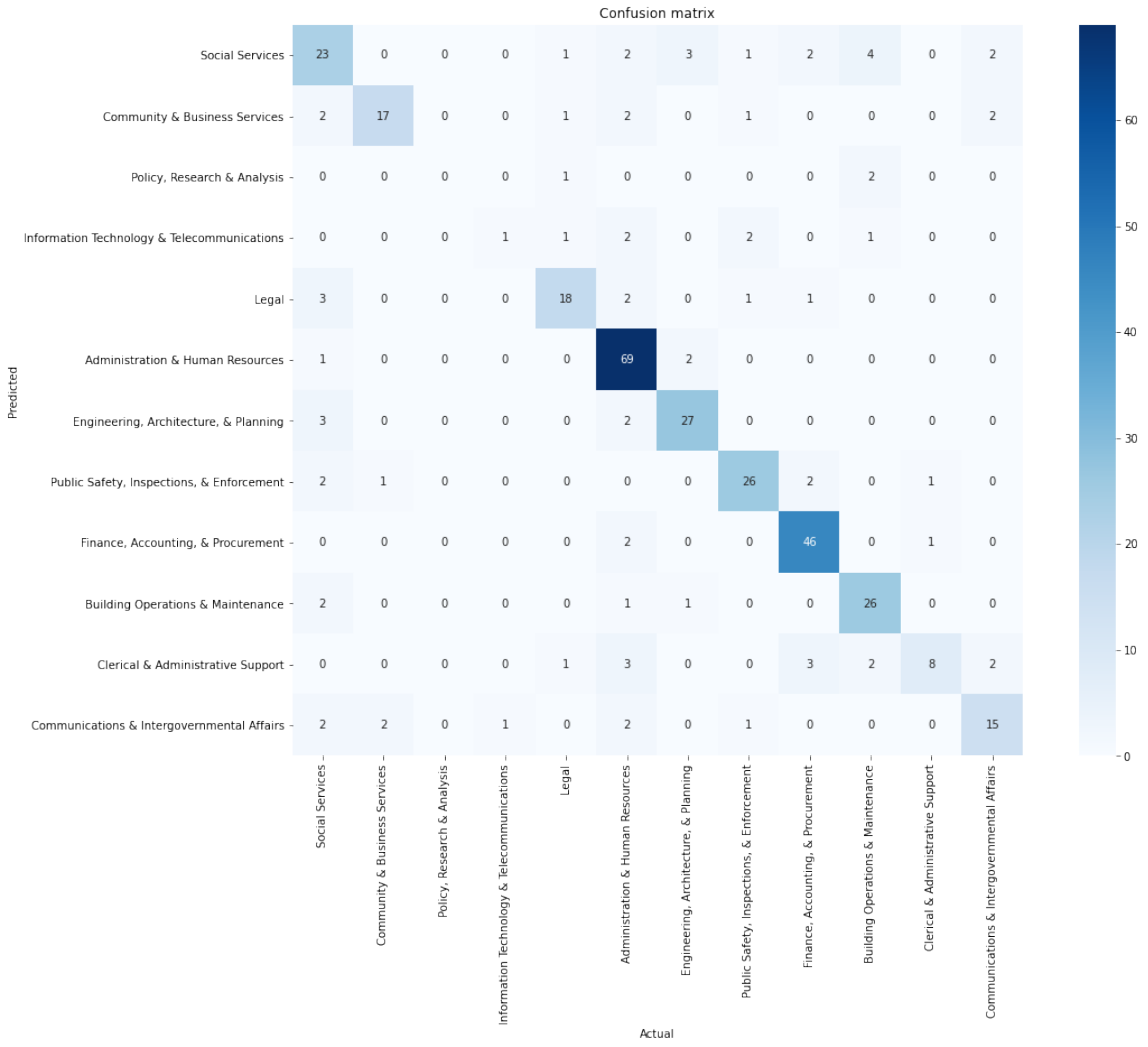
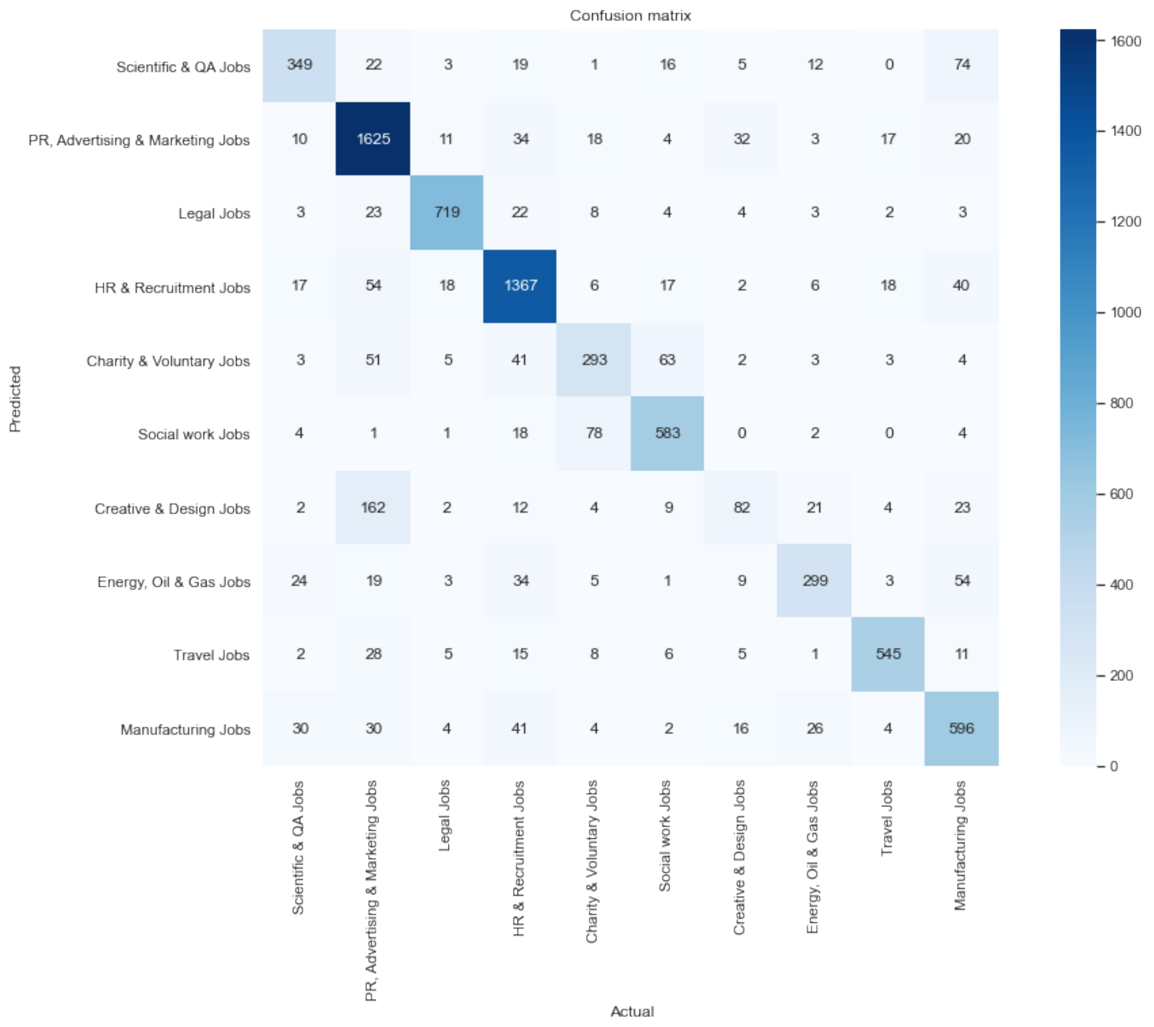


Figure 18: Confusion Matrix for the NYC Data Set Ensemble MOdel

2.2 Kaggle Data Set Confusion Matrices



Predicted \ Actual	Scientific & QA Jobs	PR, Advertising & Marketing Jobs	Legal Jobs	HR & Recruitment Jobs	Charity & Voluntary Jobs	Social work Jobs	Creative & Design Jobs	Energy, Oil & Gas Jobs	Travel Jobs	Manufacturing Jobs
Scientific & QA Jobs	349	22	3	19	1	16	5	12	0	74
PR, Advertising & Marketing Jobs	10	1625	11	34	18	4	32	3	17	20
Legal Jobs	3	23	719	22	8	4	4	3	2	3
HR & Recruitment Jobs	17	54	18	1367	6	17	2	6	18	40
Charity & Voluntary Jobs	3	51	5	41	293	63	2	3	3	4
Social work Jobs	4	1	1	18	78	583	0	2	0	4
Creative & Design Jobs	2	162	2	12	4	9	82	21	4	23
Energy, Oil & Gas Jobs	24	19	3	34	5	1	9	299	3	54
Travel Jobs	2	28	5	15	8	6	5	1	545	11
Manufacturing Jobs	30	30	4	41	4	2	16	26	4	596

Figure 19: Confusion Matrix for the Kaggle Data Set K-Nearest Neighbor Classifier

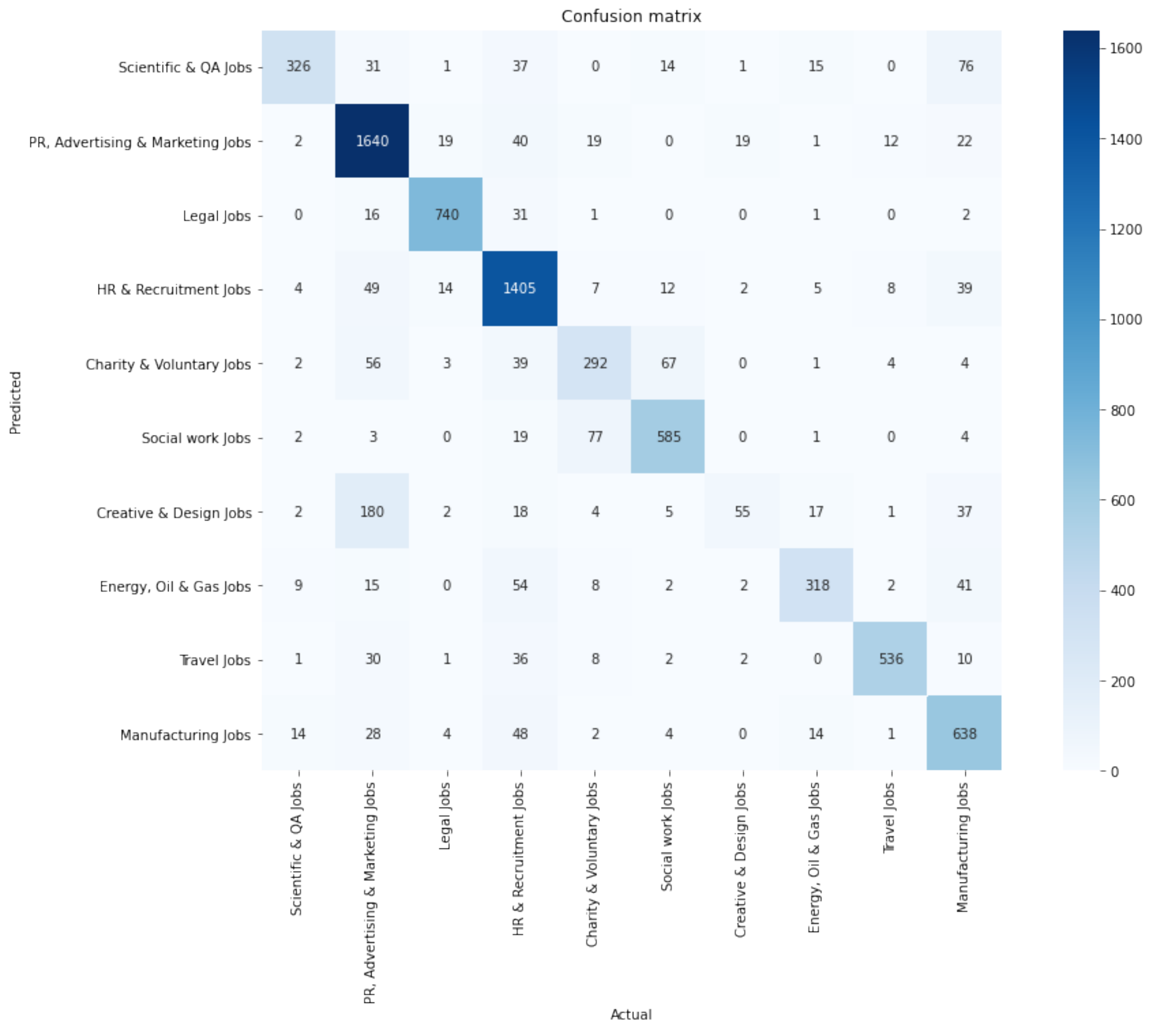
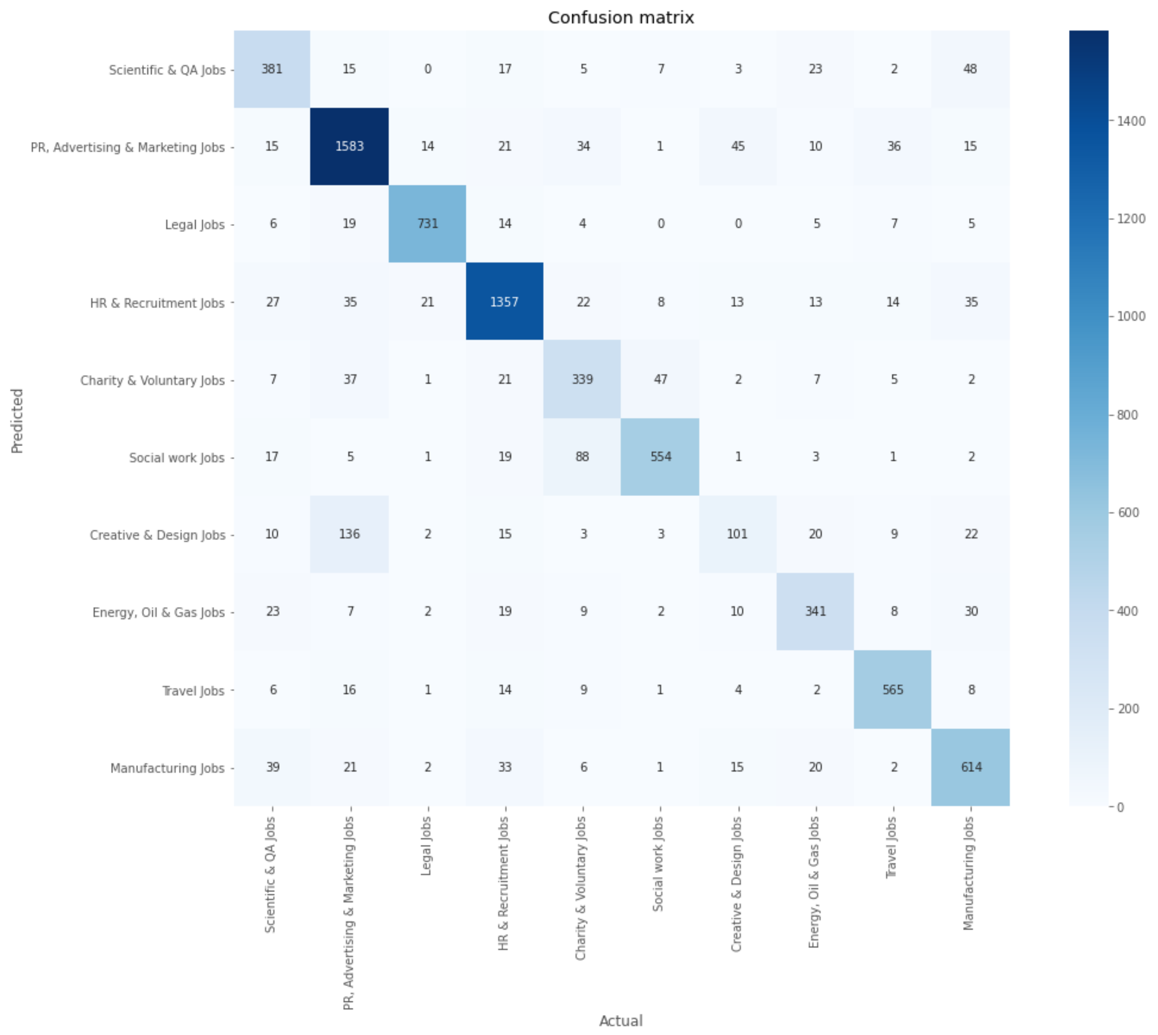
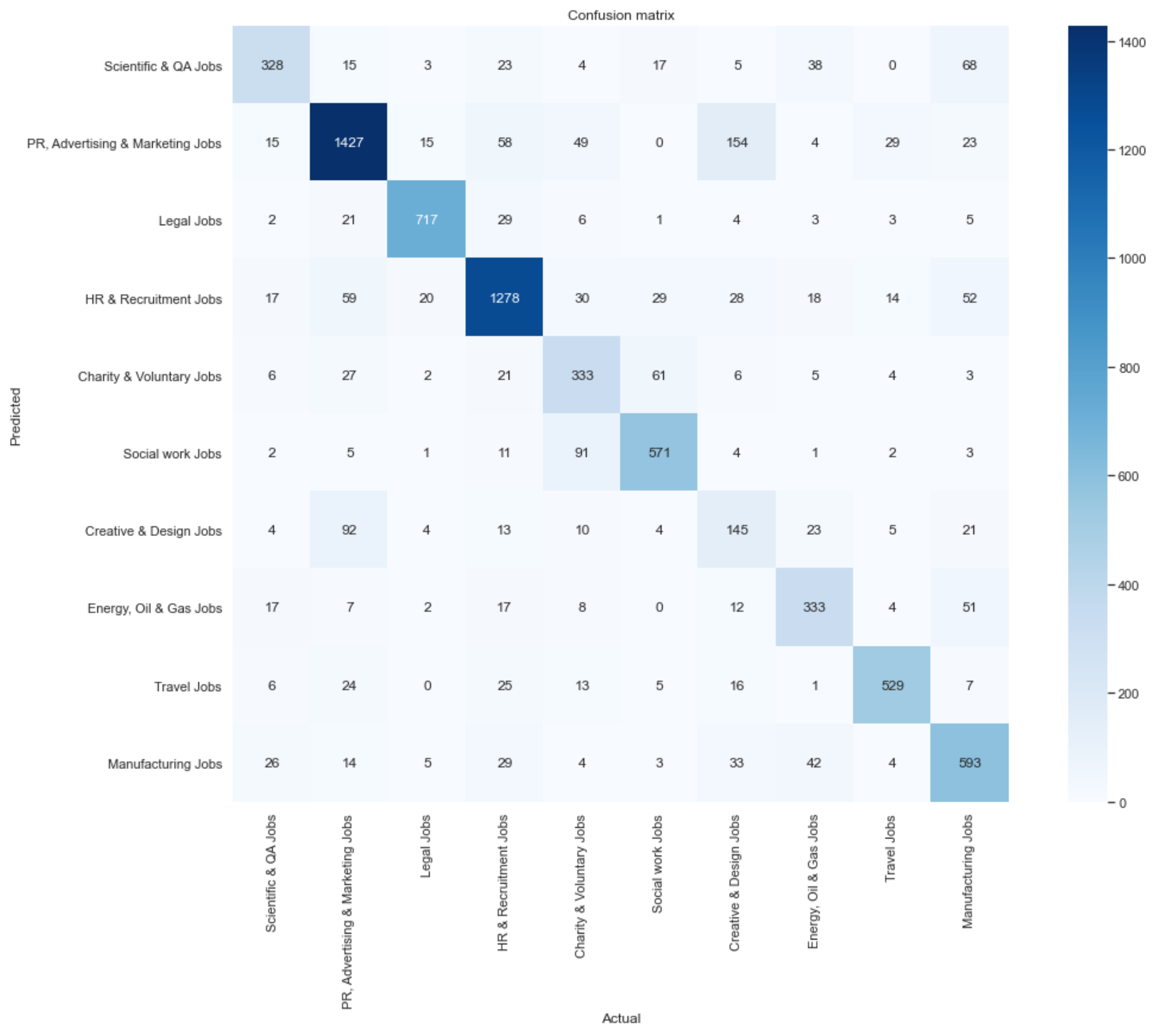


Figure 20: Confusion Matrix for the Kaggle Data Set Random Forest Classifier



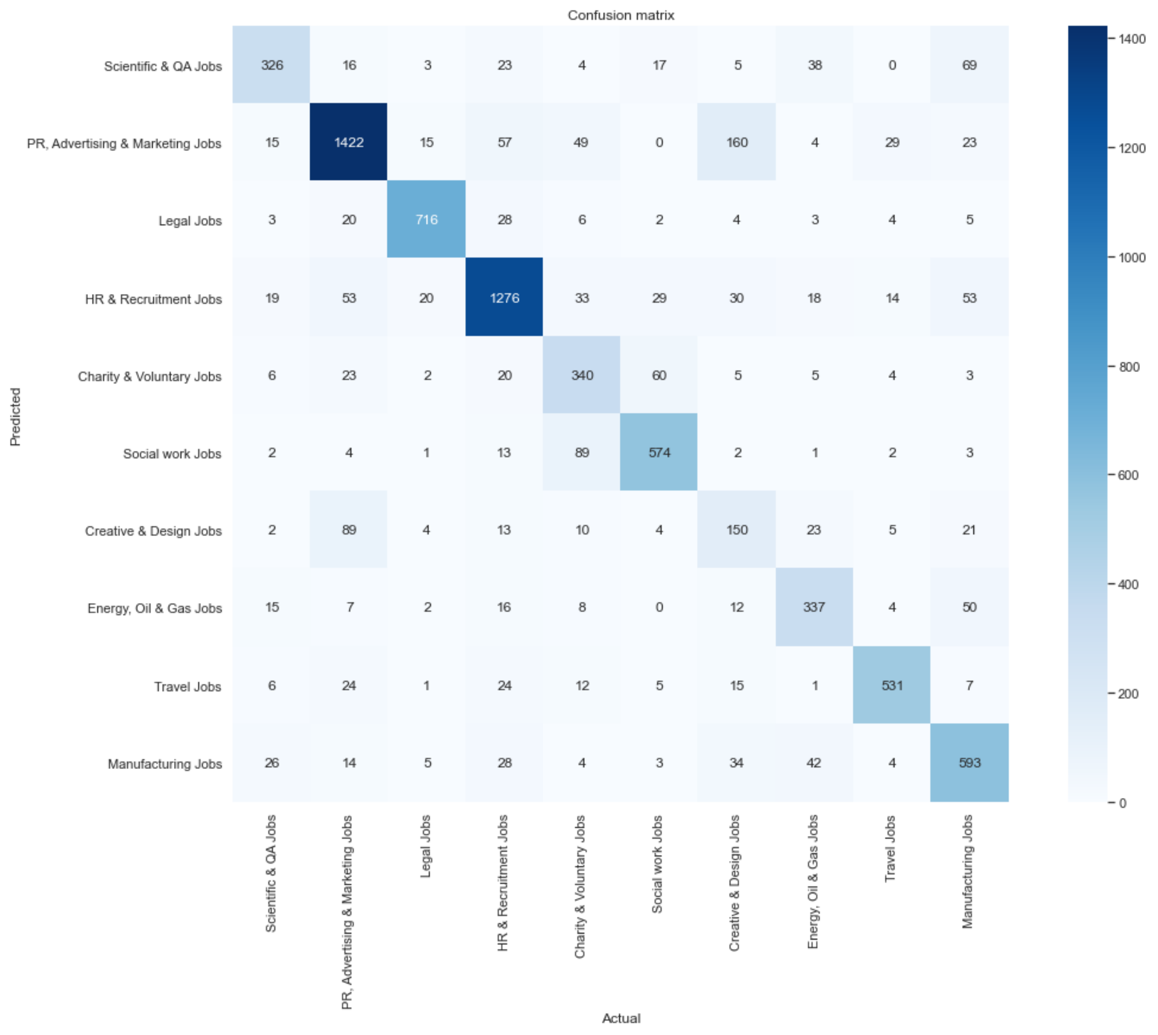
Predicted \ Actual	Scientific & QA Jobs	PR, Advertising & Marketing Jobs	Legal Jobs	HR & Recruitment Jobs	Charity & Voluntary Jobs	Social work Jobs	Creative & Design Jobs	Energy, Oil & Gas Jobs	Travel Jobs	Manufacturing Jobs
Scientific & QA Jobs	381	15	0	17	5	7	3	23	2	48
PR, Advertising & Marketing Jobs	15	1583	14	21	34	1	45	10	36	15
Legal Jobs	6	19	731	14	4	0	0	5	7	5
HR & Recruitment Jobs	27	35	21	1357	22	8	13	13	14	35
Charity & Voluntary Jobs	7	37	1	21	339	47	2	7	5	2
Social work Jobs	17	5	1	19	88	554	1	3	1	2
Creative & Design Jobs	10	136	2	15	3	3	101	20	9	22
Energy, Oil & Gas Jobs	23	7	2	19	9	2	10	341	8	30
Travel Jobs	6	16	1	14	9	1	4	2	565	8
Manufacturing Jobs	39	21	2	33	6	1	15	20	2	614

Figure 21: Confusion Matrix for the Kaggle Data Set Feed-Forward Neural Network Classifier



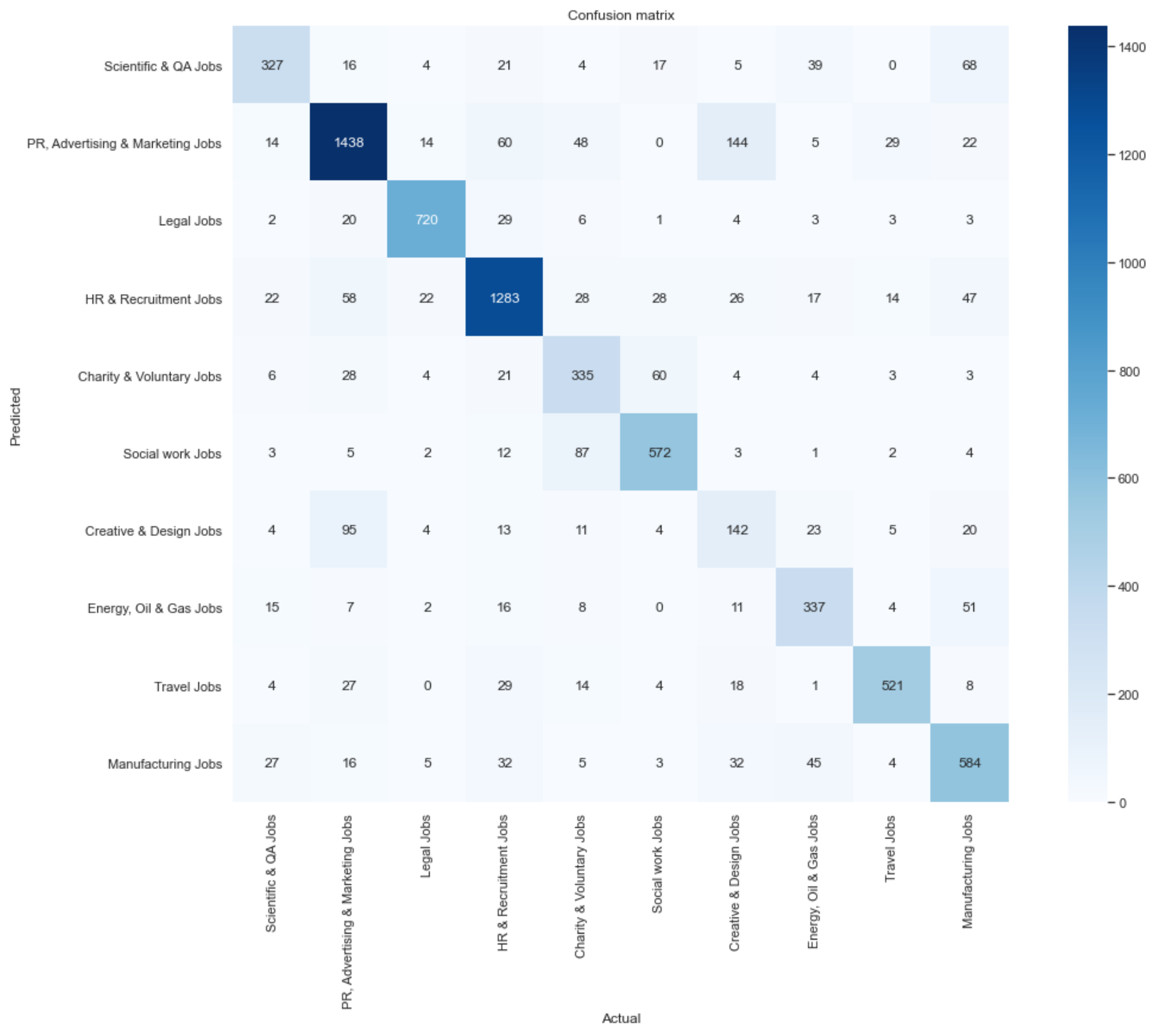
Predicted \ Actual	Scientific & QA Jobs	PR, Advertising & Marketing Jobs	Legal Jobs	HR & Recruitment Jobs	Charity & Voluntary Jobs	Social work Jobs	Creative & Design Jobs	Energy, Oil & Gas Jobs	Travel Jobs	Manufacturing Jobs
Scientific & QA Jobs	328	15	3	23	4	17	5	38	0	68
PR, Advertising & Marketing Jobs	15	1427	15	58	49	0	154	4	29	23
Legal Jobs	2	21	717	29	6	1	4	3	3	5
HR & Recruitment Jobs	17	59	20	1278	30	29	28	18	14	52
Charity & Voluntary Jobs	6	27	2	21	333	61	6	5	4	3
Social work Jobs	2	5	1	11	91	571	4	1	2	3
Creative & Design Jobs	4	92	4	13	10	4	145	23	5	21
Energy, Oil & Gas Jobs	17	7	2	17	8	0	12	333	4	51
Travel Jobs	6	24	0	25	13	5	16	1	529	7
Manufacturing Jobs	26	14	5	29	4	3	33	42	4	593

Figure 22: Confusion Matrix for the Kaggle Data Set Naive Bayes Classifier without Semantic Information



Predicted \ Actual	Scientific & QA Jobs	PR, Advertising & Marketing Jobs	Legal Jobs	HR & Recruitment Jobs	Charity & Voluntary Jobs	Social work Jobs	Creative & Design Jobs	Energy, Oil & Gas Jobs	Travel Jobs	Manufacturing Jobs
Scientific & QA Jobs	326	16	3	23	4	17	5	38	0	69
PR, Advertising & Marketing Jobs	15	1422	15	57	49	0	160	4	29	23
Legal Jobs	3	20	716	28	6	2	4	3	4	5
HR & Recruitment Jobs	19	53	20	1276	33	29	30	18	14	53
Charity & Voluntary Jobs	6	23	2	20	340	60	5	5	4	3
Social work Jobs	2	4	1	13	89	574	2	1	2	3
Creative & Design Jobs	2	89	4	13	10	4	150	23	5	21
Energy, Oil & Gas Jobs	15	7	2	16	8	0	12	337	4	50
Travel Jobs	6	24	1	24	12	5	15	1	531	7
Manufacturing Jobs	26	14	5	28	4	3	34	42	4	593

Figure 23: Confusion Matrix for the Kaggle Data Set Naive Bayes Classifier with Semantic Information



Predicted \ Actual	Scientific & QA Jobs	PR, Advertising & Marketing Jobs	Legal Jobs	HR & Recruitment Jobs	Charity & Voluntary Jobs	Social work Jobs	Creative & Design Jobs	Energy, Oil & Gas Jobs	Travel Jobs	Manufacturing Jobs
Scientific & QA Jobs	327	16	4	21	4	17	5	39	0	68
PR, Advertising & Marketing Jobs	14	1438	14	60	48	0	144	5	29	22
Legal Jobs	2	20	720	29	6	1	4	3	3	3
HR & Recruitment Jobs	22	58	22	1283	28	28	26	17	14	47
Charity & Voluntary Jobs	6	28	4	21	335	60	4	4	3	3
Social work Jobs	3	5	2	12	87	572	3	1	2	4
Creative & Design Jobs	4	95	4	13	11	4	142	23	5	20
Energy, Oil & Gas Jobs	15	7	2	16	8	0	11	337	4	51
Travel Jobs	4	27	0	29	14	4	18	1	521	8
Manufacturing Jobs	27	16	5	32	5	3	32	45	4	584

Figure 24: Confusion Matrix for the Kaggle Data Set Dynamic Naive Bayes Classifier

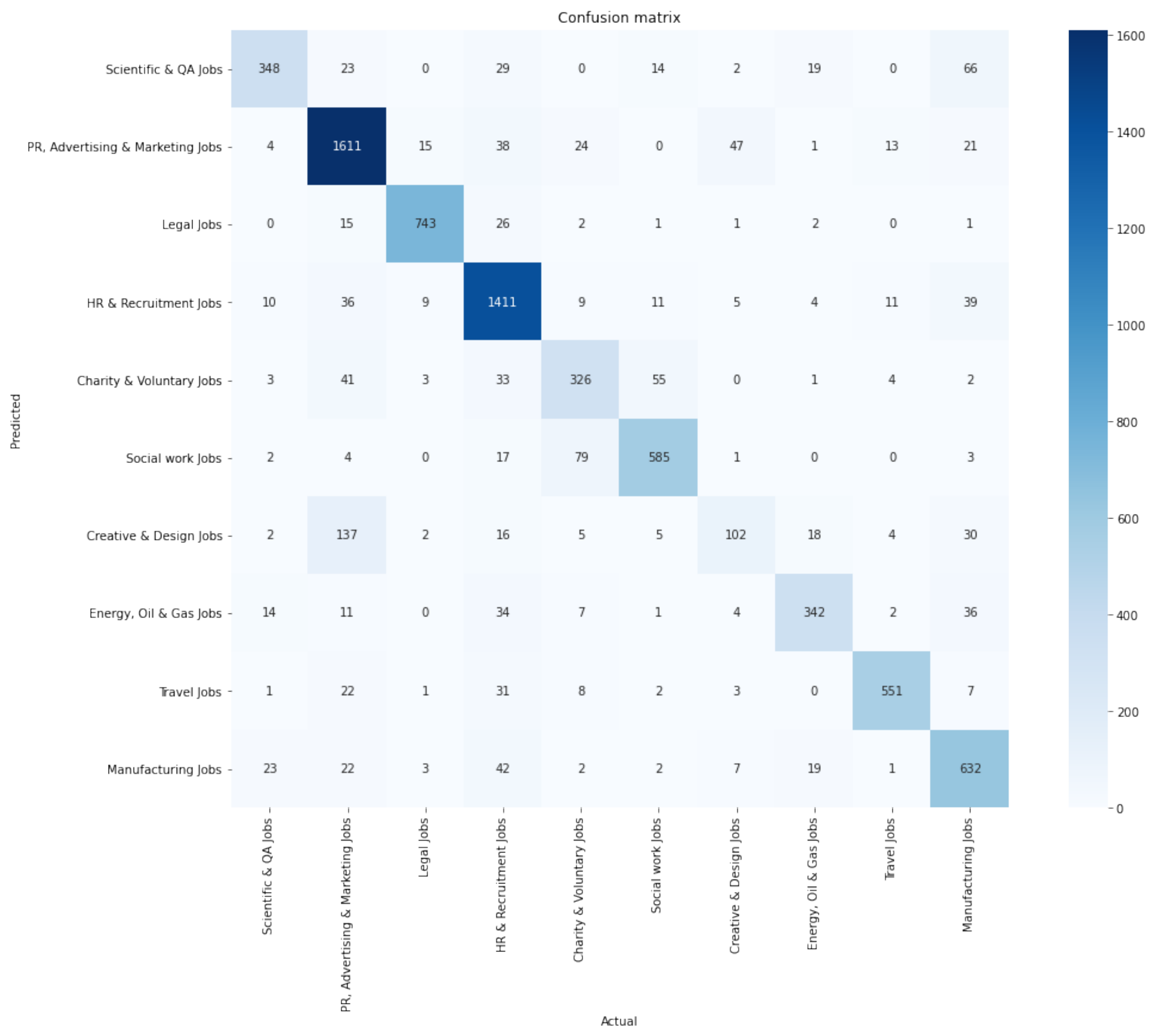


Figure 25: Confusion Matrix for the Kaggle Data Set Ensemble MOdel

Bibliography

1. Shaha T. Al-Otaibi. A survey of job recommender systems. *International Journal of the Physical Sciences*, 7(29), jul 2012.

2. Shaun Jackman and Graham Reid. *Predicting Job Salaries from Text Descriptions*. PhD thesis, University of British Columbia, 2013.

3. Sunil Kumar Kopparapu. Automatic extraction of usable information from unstructured resumes to aid search. *Proceedings of the 2010 IEEE International Conference on Progress in Informatics and Computing, PIC 2010*, 1:99–103, 2010.

4. Jie Chen, Chunxia Zhang, and Zhendong Niu. A Two-Step Resume Information Extraction Algorithm. *Mathematical Problems in Engineering*, 2018, 2018.

5. Saket Maheshwary and Hemant Misra. Matching Resumes to Jobs via Deep Siamese Network. pages 87–88. Association for Computing Machinery (ACM), 2018.

6. Ying Hong Wang and Whai En Chen. A Job Recommender System Based on User Clustering. *Journal of Computers*, 8(8):1960–1967, 2013.

7. Liting Duan, Xiaolin Gui, Mingan Wei, and You Wu. A resume recommendation algorithm based on k-means++ and part-of-speech TF-IDF. In *ACM International Conference Proceeding Series*. Association for Computing Machinery, oct 2019.

8. Walid Shalaby, Bahaa Eddin Alaila, Mohammed Korayem, Layla Pournajaf, Khalifeh Aljadda, Shannon Quinn, and Wlodek Zadrozny. Help me find a job: A graph-based approach for job recommendation at scale. *Proceedings - 2017 IEEE*

International Conference on Big Data, Big Data 2017, 2018-Janua:1544–1553, 2017.

9. Lv Hexin and Zhu Bin. Skill ontology-based semantic model and its matching algorithm. *2006 7th International Conference on Computer-Aided Industrial Design and Conceptual Design, CAIDC*, pages 12–15, 2006.

10. Nikolaos D. Almalis, George A. Tsihrintzis, and Nikolaos Karagiannis. A content based approach for recommending personnel for job positions. *IISA 2014 - 5th International Conference on Information, Intelligence, Systems and Applications*, pages 45–49, 2014.

11. Jacques F. Carriere and Kevin J. Shand. New Salary Functions for Pension Valuations. *North American Actuarial Journal*, 2(3):18–26, 1998.

12. Adzuna. Value My Resume, 2020.

13. Sananda Dutta, Airiddha Halder, and Kousik Dasgupta. Design of a novel prediction engine for predicting suitable salary for a job. *Proceedings - 2018 4th IEEE International Conference on Research in Computational Intelligence and Communication Networks, ICRCICN 2018*, pages 275–279, 2018.

14. Catherine Havasi Robert Speer. Representing General Relational Knowledge in ConceptNet5. Technical report, 2012.

15. S. L. Ting, W. H. Ip, and Albert H.C. Tsang. Is Naïve bayes a good classifier for document classification? *International Journal of Software Engineering and its Applications*, 5(3):37–46, 2011.

16. Sundus Hassan, Muhammad Rafi, and Muhammad Shahid Shaikh. Comparing SVM and Naïve Bayes classifiers for text categorization with Wikitology as

knowledge enrichment. *Proceedings of the 14th IEEE International Multitopic Conference 2011, INMIC 2011*, pages 31–34, 2011.

17. Sang Bum Kim, Hae Chang Rim, Dong Suk Yook, and Heui Seok Lim. Effective methods for improving naive Bayes text classifiers. *PRICAI 2002: Trends in Artificial Intelligence*, 2417:414–423, 2002.

18. Kaggle. Kaggle: New York City Current Job Postings, 2019.

19. Fabian Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, and D Cournapeau. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

20. Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

21. François Chollet and Others. Keras. \url{https://keras.io}, 2015.

22. M Abadi, A Agarwal, Paul˜Barham, Eugene˜Brevdo, Zhifeng˜Chen, Craig˜Citro, Greg˜S.˜Corrado, Andy˜Davis, Jeffrey˜Dean, Matthieu˜Devin, Sanjay˜Ghemawat, Ian˜Goodfellow, Andrew˜Harp, Geoffrey˜Irving, Michael˜Isard, Yangqing Jia, Rafal˜Jozefowicz, Lukasz˜Kaiser, Manjunath˜Kudlur, Josh˜Levenberg, Dandelion˜Mané, Rajat˜Monga, Sherry˜Moore, Derek˜Murray, Chris˜Olah, Mike˜Schuster, Jonathon˜Shlens, Benoit˜Steiner, Ilya˜Sutskever, Kunal˜Talwar, Paul˜Tucker, Vincent˜Vanhoucke, Vijay˜Vasudevan, Fernanda˜Viégas, Oriol˜Vinyals, Pete˜Warden, Martin˜Wattenberg, Martin˜Wicke, Yuan˜Yu, and Xiaoqiang˜Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.

23. Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. A C-LSTM Neural Network for Text Classification. 2015.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 25–03–2021	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2019 — Mar 2021

4. TITLE AND SUBTITLE Improving Text Classification with Semantic Information	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Joshua H. White, Capt, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-MS-21-M-092
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) 711 HPW/RH 2610 Seventh Street, Bldg. 441 WPAFB OH 45433-7765 POC: Aashae Eberle	**10. SPONSOR/MONITOR'S ACRONYM(S)** 711 HPW/RH
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

The Air Force contracts a variety of positions, from Information Technology to maintenance services. There is currently no automated way to verify that quotes for services are reasonably priced. Small training data sets and word sense ambiguity are challenges that such a tool would encounter, and additional semantic information could help. This thesis hypothesizes that leveraging a semantic network could improve text-based classification. The leveraged semantic information would add relevant words from the category domain to the model that did not appear in the training data. This thesis uses information from ConceptNet to augment a Naive Bayes Classifier. The experiment compares variations of a Naive Bayes Classifier leveraging semantic information, including an Ensemble Model, against classifiers that do not. Results show a significant performance increase in a smaller data set but not a larger one. The results show that ConceptNet does not add enough new or relevant information to affect classifier performance on large data sets. Out of all models tested, an Ensemble Based Classifier performs the best on both data sets.

15. SUBJECT TERMS

Text-Based Classification, Job Description, Document Classification, ConceptNet, Naive Bayes Classifier, Semantic Network

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Lt Col George E. Noel, AFIT/ENG
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	79	(937) 255-3636, ext 4613; george.noel@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18