Analyzing LinkedIn Profiles Using Machine Learning

By
Benjamin Andrew and Alexander Thiel

Abstract

Understanding the necessary skills required to work in an industry is a difficult task with many potential uses. By being able to predict the industry of a person based on their skills, professional social networks could make searching better with automated tagging, advertisers can target more carefully, and students/candidates can better find a career path that fits their skillset. The aim in this project is to apply deep learning to the world of professional networking. Deep Learning is a type of machine learning that has recently been making breakthroughs in the analysis of complex datasets that previously were not of much use. Initially the goal was to apply deep learning to the skills-to-company relationship, but a lack of quality data required a change to the skills-to-industry relationship. To accomplish the new goal, a database of LinkedIn profiles that are part of various industries was gathered and processed. From this dataset a model was created to take a list of skills and output an industry that people with those skills work in. Such a model has value in the insights that it forms allowing candidates to: determine what industry fits a skillset, identify key skills for industries, and locate which industries possible candidates may best fit in. Various models were trained and tested on a skill to industry dataset. The model was able to learn similarities between industries, and predict the most likely industries for each profiles skillset.

Table of Contents

Chapter I: Introduction

This section will cover the topic of this research project, the motives in developing this technology, and the existing work in the field. Furthermore, the terminology involved in this problem domain will be defined for the rest of the paper.

A. Topic

LinkedIn is a large social network where people can post information about their professional lives and skills. Recruiting agencies will use this site as a method to find potential employees for their companies based upon the information that each person lists about themselves. Previously, recruiters might manually search LinkedIn and other sites to find possible candidates or use programs to automatically search for chosen qualities. Recently, however, some research has been done to see if machine learning can improve the process by removing the requirement for humans to choose what qualities their recruiters or programs are searching for in possible candidates. Once the models have identified potential candidates, human recruiters can handle other important hiring factors like personality, culture, and the final say.

Machine learning is the process of using computers to search for a mathematical function to approximate the relationship between input data and a result. This is done by creating some standard model that data is fed into which has a corresponding objective function. The computer will try to minimize the objective function by changing aspects of that standard model. By repeating this minimization the model will hopefully, "train," to become more accurate at whatever task it is given. There are many different types of machine learning models, but recently the largest achievements have been made using variants of neural networks: specifically deep neural networks. These neural network models roughly emulate the mechanics of biological neurons by taking an input, applying some weight to it, and sending it to another neuron. Deep neural networks are simply neural networks with many, "layers," of neurons between the input and output of the neural network. This increase in layers allows for deep neural networks to learn very complex relationships between the inputs and outputs.

This project is an analysis of the application of machine learning to attributes on linkedin profiles, for the purpose of predicting the industry that individual profiles work in.

B. Motivation

This research project emerged from the observation that recruiting is often difficult for companies to do in a day and age where there are billions of people around the world, all of them potential hires. Instead of having resumes *always* be looked at by a human eye, having machine learning analyse resumes or profile data from a person could help whittle down the mountains of potential candidates to just a few hundred, who can then be contacted by recruiters.

The idea behind applying machine learning to a problem is that there is some mathematical relationship between the input and output that may be too complex to derive analytically, or by hand. Thus machine learning becomes a powerful tool for scenarios where there are not apparent indicators for determining the outcome, or clear and direct relationships. Recruiting specifically is a good candidate because of this specific quality of machine learning. While there are many indicators that recruiters currently use to determine how "fit" a candidate is for a company, machine learning will be able to identify more indicators, combinations of indicators, and complex models to determine if a candidate is a good fit for their industry.

This has an added benefit of being able to identify good candidates working in many different industries that have overlapping skills that recruiters may not realize. Similarly such a tool could be used by a candidate to determine if they are accurately portraying themselves as a valuable member of a certain industry.

C. Literature Review

Machine learning being applied to recruitment is not a new concept. A considerable amount of research has been applied to how machine learning can be applied to recruiting. Also, many novel applications of machine learning have been successfully implemented.

In the recruiting world a large emphasis has been placed on on the value of employees, and how to identify them using machine learning to analyze resumes[1]. This can be a viable option for ranking possible candidates for recruiters to look through[2][3][4]. These methods use supervised and unsupervised machine learning to complete their tasks[5].

Some successful implementations of machine learning in this area can be found in the form of recommendation engines. Career and academic recommendation engines have been developed to suggest what jobs to search for[6] or what courses to take in an academic career[7][8]. Notably, the job recommenders use skills as an important feature to create recommendations.

Neural networks specifically have had success in the area of recruitment because of their ability to learn complex behavior[9]. The following papers have implemented neural networks to predict whether applicants are good for a specific job. The input here is simply a list of applicants and their resumes and the output is whether the employee is hired. A setup like this can be used for large companies, to determine how likely a candidate is to get the job (essentially a ranking)[10].

Activation is a key part of neural networks. Activation functions take a numerical input, apply a nonlinear function to the number, and return the number. They are important in neural networks because they allow them to solve nonlinear problems. One previous system for job matching used activation functions in their respective network[11], and since this experiment uses neural network as well, it was decided to include an activation function. Based on this research, cross-entropy was chosen for the cost and the Rectified Linear Unit (ReLU) activation function[12] was chosen for all layers except the output layer which uses a Softmax activation function.

The Adam Optimizer is a very common optimization algorithm for neural networks that is integrated in tensorflow (the neural network library to be used.) It is a form of gradient descent that adjusts weights between neurons with every training step[13]. Adam Optimizer is also good useful for analog networks (i.e. an output that is not just 1 or 0 but some number in between)[14].

Another point that's worth mentioning is the fact that the dataset used in this experiment is not stationary. A stationary dataset means that the data does not *need* relabeling over time. For example, a picture of a cat being labeled as a cat will always be true. However, the necessary skills to work in an industry will change over time. While there is research in the area of incremental learning over time[15], self organizing networks that change as the data changes[16], and even networks that expand in scope while the data changes[17], we did not include these in the scope of this research project. Instead, the focus was more on whether or not it was possible to map from the skill domain to industry domain with a useful accuracy.

D. Terminology

For clarity, some of the terminology that will be used in future explanations has been defined to reduce the impact of ambiguous or conflicting definitions from other sources. Some terms may not be defined here and will used as is agreed upon by standard definitions.

Profile
A profile in this case is a LinkedIn page containing data about a person. This may include many things like: name, location, current employer, current job, past experience, skills, industry, number of connections, and many more aspects.

Candidate
A candidate is determined to be a person who holds a LinkedIn Profile. They may have any combination of the above listed qualities to be a candidate, but not all candidates may be evaluated. More complete profiles may provide better (more accurate) relationships between input and output data.

Classification Model
A classification model is a machine learning program that takes inputs and produces as outputs the “class” or “label” that it believes has the highest probability of being correct.

Minibatch Training
Minibatch training is assumed to be when the model takes a group of training examples that is smaller than the entire set of training examples, trains on the data provided by that group of training examples, and then moves to the next group of training examples. This is repeated until the entire set of training examples has been examined. Minibatch training may be performed many times on a single model.

Chapter II: Methodology

This section expands upon the method used by explaining the research method and entire data pipeline. A visualization of this general process is shown in Figure 1. This chapter details the individual elements of the data gathering, preprocessing, how that preprocessed data is input into a network, and how the network output is turned back into human readable data. Section A will include the tools used to accomplish all tasks. Section B documents the process of collecting data. Section C covers the data processing that occurs just before the learning process.

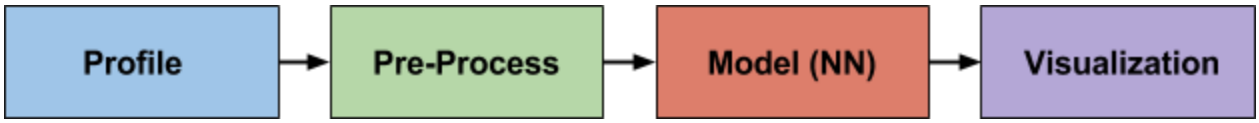

Figure 1: This figure shows the data flow from a single linkedin profile, preprocessing, entry to the neural network, and visualization of the network output.

A. Tools

A large portion of this project was software development. Various existing software libraries were used to create the data gathering, processing, visualization, and machine learning aspects of this project. Shown below are the high level software components that were used.

1. Python

All of the data pipeline was written using the Python 3+ programming language. Python was chosen for several reasons. The primary reason is that Python is the most widely used programming language in the field of machine learning with around 60% of machine learning developers and data scientists using it non exclusively [2]. This popularity is especially desirable because of the resulting number of supported libraries and references. Other lesser reasons for choosing this language are: familiarity, ease of use (rapid prototyping), a wide range of built in tools for data handling and analysis, and a large open source community.

The following is a list of Python modules used in this project:

Reading and Parsing HTML

- Beautifulsoup4
- Lxml

Natural Language Processing

- NLTK

Data Processing and Visualization

- matplotlib
- Scipy

- Sklearn
- Numpy

Machine Learning Framework

- Tensorflow

Website Scraping

- Selenium
- Fakeuseragent

2. Selenium

Selenium is a browser control tool is usually used for testing websites on multiple browsers programmatically. We used Selenium for controlling a Google Chrome browser in order to gather LinkedIn profiles (covered in section B).

3. Github

Github is a website dedicated to hosting software and code, with version history available. We used github to host our remote repository, which can be found here: https://github.com/apockill/ResumeML

B. Data Collection

Data collection is deemed to be the part of the project that was dedicated to all aspects of collecting data. Using a combination of the methods listed below, roughly 900,000 LinkedIn profiles were gathered. As a note, all information gathered was publicly available information, none of which required logging in to view a profile. Figure 2, shown below, is the overall data collection process. This section covers how data was gathered programmatically plus its technical aspects and the use of a pre-collected dataset.

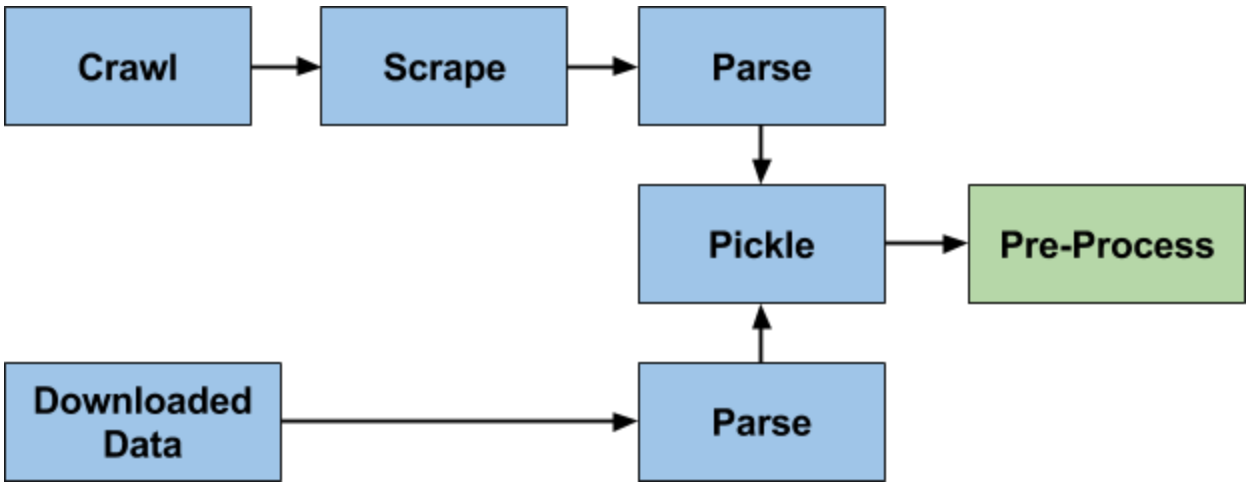

Figure 2: How data flows within the Data Collection block from Figure 1.

1. Crawling

Crawling is the method of searching (or in some cases mapping) a website by following hyperlinks. Along the way information can be recorded. Using a parameterized Google search to only return results from LinkedIn, a list of relevant links are provided that have already been crawled by the Google bot. With this method, additional "soft" search parameters could be implemented to help influence what type of profiles would be at the top (an example found in Figure 3). Next, a script was created using Selenium and its Python interface to use the created parameterized search and iteratively go to each returned Google result, check if the link directed a profile, check if the profile had already been visited, and scrape the profile by downloading the profile's HTML source code. Note that the profiles that were crawled were *not* a random sample of the average linkedin profile, but were heavily focused on the search terms used. For example, a large number of software engineering people were scrapped because our search terms tended to have 'software' in them. This is not a problem for the results however, because the training process of a neural network is agnostic to the source of the data (training is discussed in Chapter III).

a) Scraping

Given a website, scraping is the process of gathering information from the site. There two mechanisms used for scraping websites: general and specific. Specific scraping is gathering only desired information information from the site. General scraping, is the process of collecting all available data from a website. General scraping was implemented for this project by instructing Selenium to download the HTML source for a profile to a directory and to give the HTML source a unique name. This allowed efficient data collection before the specific types of data necessary was unknown. Additionally it should be noted that general scraping does not increase the load on the servers hosting a website as all information is already sent to the machine accessing the site.

b) Parsing

Once the HTML source had been acquired, information needed to be parsed from the source to isolate the useful information about a candidate from the formatting and undesired data. A program was created, using the BeautifulSoup Python module, to locate all relevant data and store it as strings, numbers, etc in memory. Figure 3 shows the type of searches that the webcrawler was using to find fresh profiles to scrape.

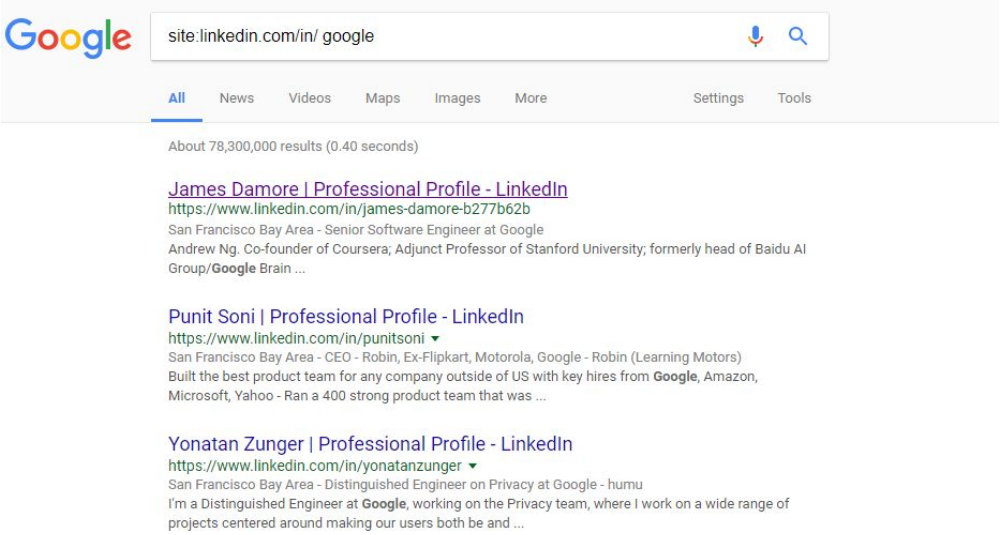

Figure 3: An example of a parameterized Google search that the Selenium browser might encounter with a special focus on Google employees.

2. Pre-Collected Dataset

Another method of obtaining data was searching to find if any sources had already gathered a large number of LinkedIn profiles in some form. After searching through many sources an existing dataset was found available for download that used the JSON format to store all information presented for a very large number of profiles. The information gathered in this dataset matched the information that had been gathered from the HTML profiles in the web crawling efforts.

a) Parsing

In this case the dataset sourced was pre-parsed in the JSON format. This means that the parsing program simply needed to be instructed to read desired information from the JSON format which Python supports.

3. Pickling

With such a large number of profiles all containing several pieces of necessary information, the program for parsing all profiles began to take upwards of 2 minutes to load all the desired information into computer memory. To reduce loading times pickling was introduced. Pickling is a method of saving python objects to storage and loading them at a later time in another script. Using pickling we were able to load every profile from both HTML and JSON formats, parse all of the attributes, and then pickle the parsed python objects. This reduced load times by 75%, speeding up testing iterations.

C. Data Pre-Processing

Much of the work related to creating an effective machine learning model comes before any machine learning takes place. It is necessary to pre-process or "clean" data that is being input to the model. In low level cases this can mean removing data that is not necessary, or in a more complex case it can mean normalizing different ranges of numeric values. There are many different methods applied clean data being fed into machine learning models, and the specific tasks change depending on the type of input data. In this section, the data cleaning tactics used in this project are explained which include: the effects of thresholding data, removing punctuation, splitting phrases into words, lemmatization, stemming, alphabetization, the effects of these steps, and how all steps culminate into a lexicon for training. Figure 4 is an overview of the data cleaning process and its effect on an example skill.

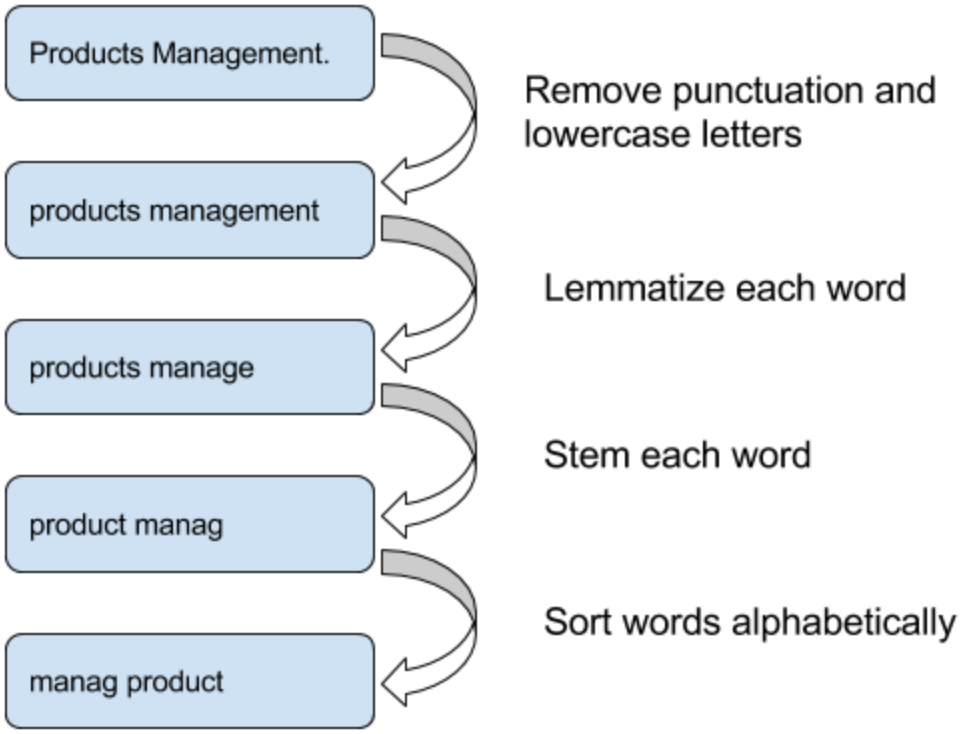

Figure 4: An overview of the preprocessing techniques that are applied to every "Skill" and "Industry" string before any operation.

1. Threshold Data

All members of LinkedIn fill in their profiles to different extents. The varying levels of completeness of profiles was investigated before starting the machine learning model. Figure 5 below shows how many individual skills people tend to write out on their profiles.

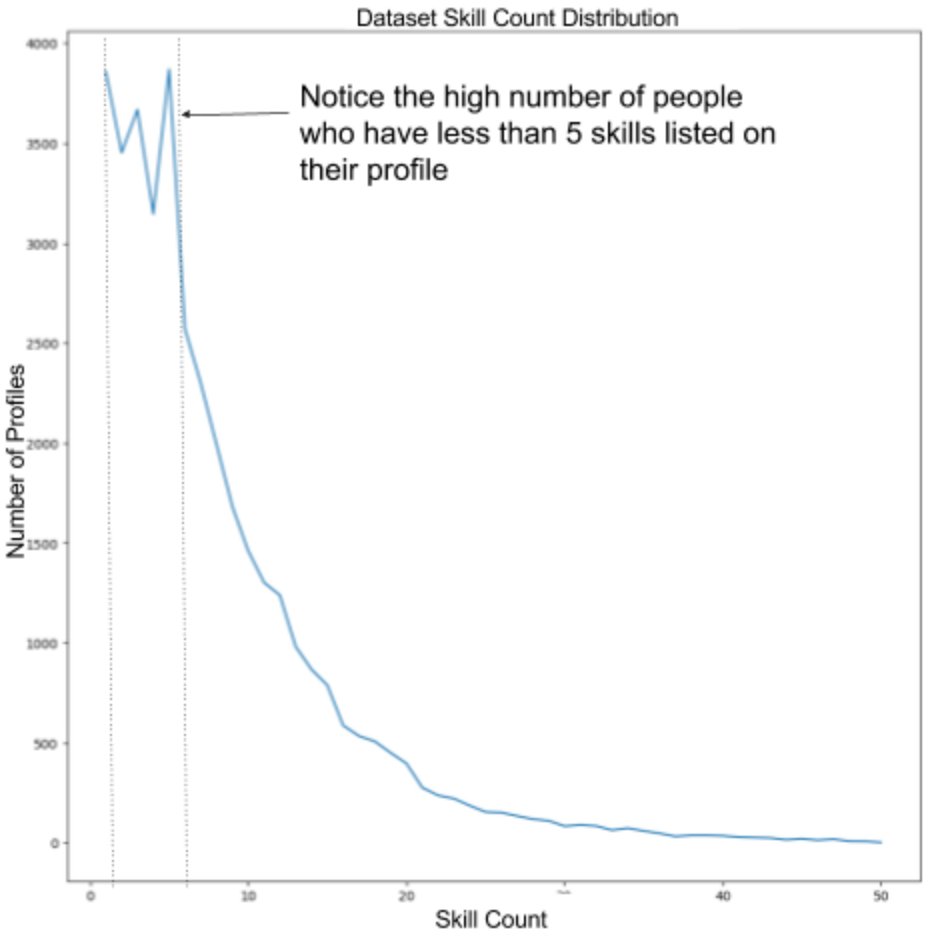

Figure 5: A distribution of profiles vs the number of listed skills they contain.

Notice the high number of people who have less than 5 skills on their profile. This means that our machine learning model must be able to guess someone's industry based on a sparse amount of data. People with 10+ skills are essentially giving the model more clues to guess by. Because of this, training was tested on different dataset configurations. Certain configurations set the minimum number of skills to greater than 3, meaning that the model would be given at least 3 skills, or clues, to base the prediction on. We noticed that raising the minimum helped improve the model's prediction accuracy.

2. Remove Punctuation

Remove punctuation from skills. This is a simple step in the sanitization process. Some profiles have punctuation in their skills and it helps to condense similar skills if punctuation is removed. This also includes lowercasing the letters in the skill. Although uncommon, a simple step like this can help prevent issues with input data.

3. Split Words

To prepare for the next steps stemming and lemmatization, all skills were split into individual words. After stemming and lemmatization, the words will be recombined back into a single skill string.

4. Lemmatization

Lemmatization is a more complex process than stemming. It takes a word as an input and returns that word's lemma. Depends heavily on the part of speech (i.e. noun, verb, adjective, etc.) to produce a correct lemma. For example, "run" ⇒ "run", "running" ⇒ "run", and "ran" ⇒ "run". This means that skills using different variations of the same word will be counted as the same skill.

5. Stemming

Stemming is a method of reducing a word to its stem. A stem is not necessarily a word, but it is root of many different words with different affixes. Thus many types of similar words will be reduced to the same stem. This can be done in many ways. Some algorithms will remove affixes (sacred has its -ed affix removed and becomes scar). There are many different algorithms available to use. For example, "run" ⇒ "run", "running" ⇒ "run", and "ran" ⇒ "ran".

6. Alphabetizing

Since skills tend to be one to three words long, the order of the words in the skills were alphabetized. This meant that skills like "Managing Products" and "Product Managing" would be counted as the same skill, since the lemmatized and stemmed words are then sorted alphabetically.

7. Comparison of Different Preprocessing Steps

The goal of removing punctuation ⇒ lemmatization ⇒ stemming ⇒ alphabetizing for each skill is to reduce the number of unique skills in the entire dataset. Table 1 below shows the number of unique skills with different preprocessing steps. It should be noted that due to time constraints, no testing was done to ensure that stemming after lemmatization is a necessary step, and that it yields results that accurately keep the meaning of the data the same.

Table 1: A comparison of different preprocessing technique combinations and their effects on the total unique skill count

Delete Punctuation	**Lemmatize**	**Stem**	**Alphabetize**	**Unique Skills after Preprocessing**
✘	✘	✘	✘	79,260
✔	✘	✘	✘	70,035
✔	✔	✘	✔	68,760
✔	✘	✔	✔	66,545
✔	✔	✔	✘	67,060
✔	✔	✔	✔	66,504

8. Lexicon Creation

In this step, all skills were gathered from the dataset. A "lexicon" was created here by gathering and sanitizing all of the possible entries, counting them, and adding entries that have a threshold amount of appearances throughout the dataset. A lexicon is just a list of phrases in an array, and is used to convert from an array of human-readable phrases to an array of numbers that can be fed into a model (or output from a model).

Specifically two lexicons are created: one for skills (input lexicon) and one for industry (output lexicon). If an entry is evaluated and it has not been evaluated before then it is added to the lexicon. The result is an array that will be used to create input and output vectors for the machine learning model. An example of using a lexicon would be to convert a set of skills to an array of numbers that can be input into the model. Another example would be to take the output array of numbers from the model and get the actual human-readable text output using an output lexicon.

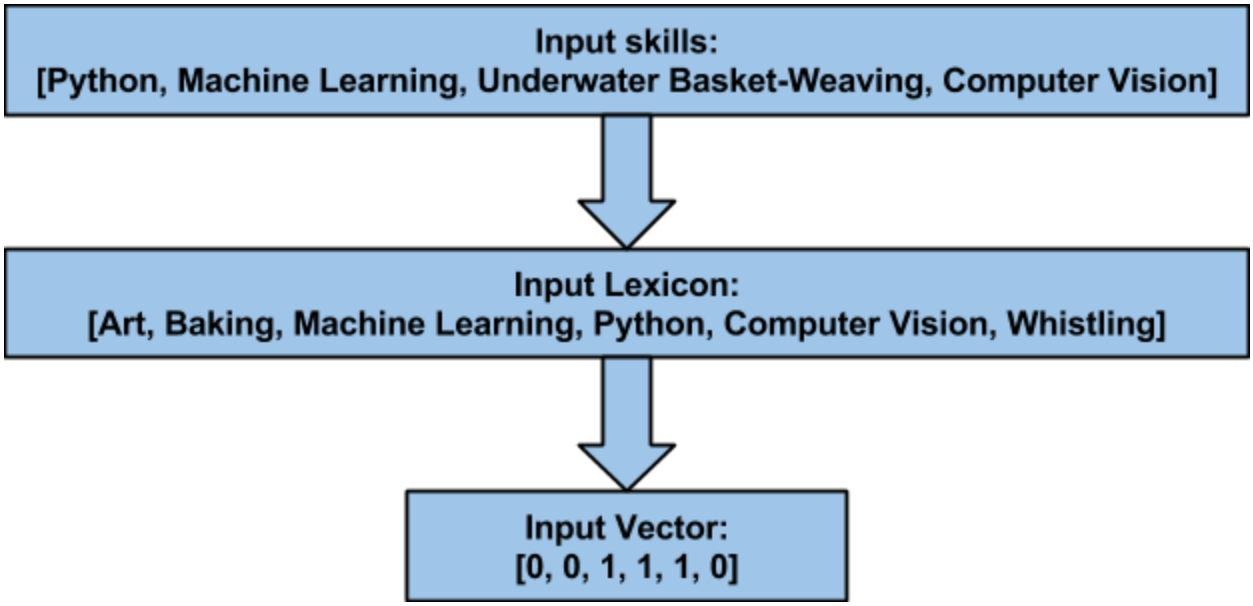

Figure 6: An example of using an input lexicon to create an input vector for the machine learning model from the skills contained in a profile.

The input and output lexicons were limited to only contain entries that appeared a certain number of times in the overall dataset. For example, if a particular skill appears less than 100 times in the entire dataset, it might not be included in the input lexicon. If a particular industry only appears 2000 times in the entire dataset, it might not be included in the output lexicon. Doing this reduces the dimensionality of the problem, and the amount of data necessary to train the model. On the other hand, it limits the amount of inputs, or clues, the network can take in, and it reduces the number of outputs that it can predict. Various input/output threshold configurations were tested, which can be seen later in Table 2.

Chapter III: Classification Model

This section will cover the high-level structure of the classification model that was used for predicting industries based on skills, and go over the flow of data from a user's profile through the network, how the output is translated from numbers to a readable industry prediction, and finally the methods of how the model is trained from scratch to solve for these relationships.

A. Architecture

The model used for prediction is a five layer deep neural network. There is an input layer, three hidden layers, and an output layer. The reason that multiple hidden layers are chosen versus a single hidden layer is that multilayer networks are known to generalize data better, and are capable of recognizing XOR combinations in data [1].

Various network hidden layer neuron counts were tested. Numbers ranging from 500 to 4500 neurons per hidden layer were tested, on each dataset, to find an optimal size of network. More information about specific network sizes that were tested and the results can be found in Chapter IV.

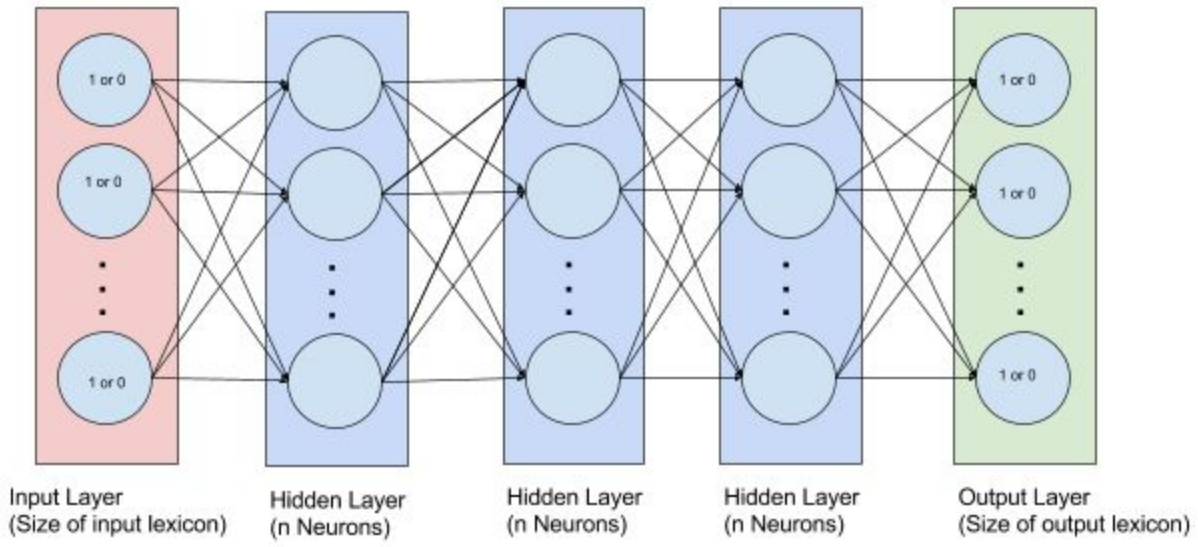

Figure 7: A diagram showing the architecture of the classification neural network used in this paper

Each layer, excluding the output layer, uses the standard neuron function given in Eq. (1), where W is the 'weight' of the neuron, b is the offset, x is the neuron input value, and y is the neuron output value.

$$y = Wx + b \tag{1}$$

The output value y is fed into a ReLU activation function. An activation function is a function that the output of a neuron gets passed through. Activation functions take a variety of forms, but

ReLU is simple: if the output of a neuron is less than 0, it changes the output to 0. If it is greater than 0, it keeps it at its current value. The activated values for one layer are then fed into the next layer.

B. Inputs and Outputs

The model is fed the skills of a user, and it output the predicted industry that the user worked in. Figure 8 shows what the data flow might look like for a given user, and input/output lexicon. Notice that the order of the input lexicon is what defines the order of the input array to the network, and the output array of the network also corresponds to the order of the output lexicon. The lexicons act like a hashmap from array to human readable labels.

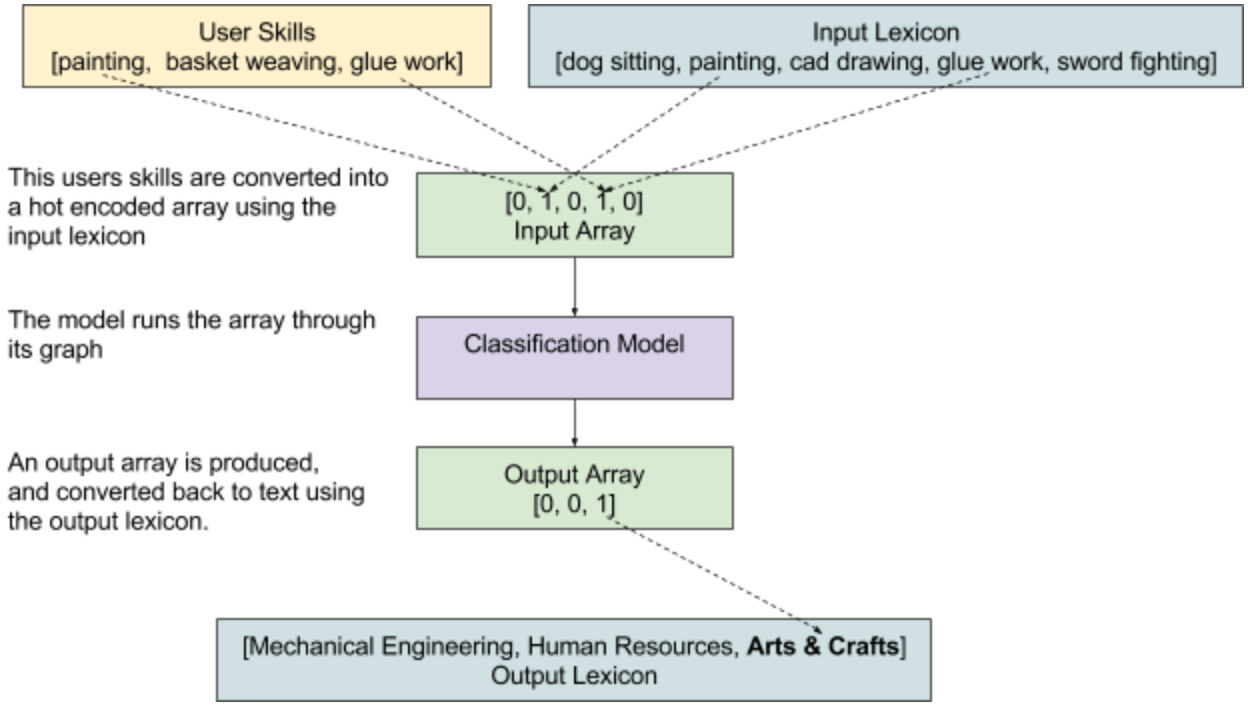

Figure 8: A diagram showing an example input/output to the neural network

C. Training

Training a classification model is the process of applying an optimization algorithm to the fresh model. A neural network can have millions of possible numerical parameters that start off completely randomized, and through the process of training they are solved such that the network accuracy increases beyond random guesses. The following subsections will cover the particular optimization algorithm that was used, the hyperparameters for structure of the network, and the visualizations of the training as it happens over time.

1. Backpropagation

Backpropagation is a common method of training a neural network where the system output is compared to the desired output, and the system variables are adjusted until the difference

between the two is minimized [20]. During training the model is shown many examples the input/output pairs which is then used during backpropagation optimization to change the network parameters so that the network outputs correctly for each input in the training set.

2. Training Parameters

Tensorflow has different built-in methods for training, and for this project the Adam Optimizer was used with a (typical) learning rate of 0.001. As a note other learning rates were attempted early on when results were not being recorded because it was during the process of getting the network to train. During this time the learning rate was settled. This optimizer was minimizing the cost function that consisted of the soft max of the cross entropy of the final layer. All networks were trained to 1000 epochs (or iterations over all training examples)

3. Training Results

Tensorboard is tool made for tensorflow models that is utilized for various neural network related visualization tasks. For this paper, tensorboard was used to visualize the training of each model, and to gauge the accuracy of the model over time as it trained.

Figure 9: A graph of 5 models performance on a particular dataset configuration during training.

These graphs are helpful for comparing the performance of different model configurations on the same dataset. In Figure 9 it's clear that 4000 neurons/hidden layer is the optimal configuration for dataset 3.

Chapter IV: Experiments

In the following section the various datasets that were created will be described, along with a selection of experiments that were performed on those datasets. The accuracy of the best performing model for each dataset will be shown. In the next chapter, results will be analyzed.

A. Datasets

Four datasets were created throughout the research project. Each dataset was created based on results of experiments performed on the previous dataset. The creation of a dataset is done based on three criterion: minimum skills per profile, the skill lexicon threshold, and the industry lexicon threshold.

Minimum Skills Per Profile
The minimum amount of skills that a profile must have listed in order to be included in the dataset. Increasing the minimum number of skills per profile in the dataset helps the network have more information to learn from, but lowers the overall size of the dataset, which can hurt training.

Skill/Industry Lexicon Threshold
The minimum amount of times that a particular skill or industry must appear in the overall dataset in order to be included as an input or output to the network. Raising this threshold will reduce the dimensionality of the network which helps accuracy, but it will also lower the overall size of the dataset, which hurts training.

Take note in Table 2 that as the criteria for dataset creation is increased, the total examples in the dataset lowers dramatically. At the same time, the dimensionality of the input and output also lowers.

Table 2: A table showing the names, configurations, and attributes of each dataset that was trained and tested on

	Creation Parameters			**Dataset Characteristics**		
Name	**Min. Skills Per Profile**	**Skill Lexicon Threshold**	**Industry Lexicon Threshold**	**Total Examples**	**Input Lexicon Size**	**Output Lexicon Size**
dataset_1	1	100	2500	48,078	1124	58
dataset_2	1	100	3500	42,899	1124	42
dataset_3	3	125	3500	34,006	944	42
dataset_4	6	125	3500	21,922	944	42

B. Tests

The results of the models are defined as the percent accuracy in predicting the industry of a particular profile based on that profiles skills. A total of nineteen networks were trained and tested; the experiments were performed on four different dataset configurations, and for each dataset configuration many different model configurations were trained and tested.

The test set for every dataset was fixed at 10,000 entries of the dataset, randomized at creation. The "accuracy" of each model is judged by (correct predictions) / (total predictions). To view the results and parameters of all nineteen tests, look at figure A-1 in the appendix. Shown below are the best performing models that were found through trial and error.

In Table 3,note how the accuracy of the models increases from dataset_1 to dataset_4. This is because each dataset was constrained in scope, using the criteria shown in the datasets section. By constraining the scope of the problem (number of industries, number of inputs) the model is better able to map these two data types. Also note that despite trying many different hidden layer sizes, 4000 neurons remained consistently the best size for the model, despite changes in the dataset.

Table 3: The best performing model configurations on each dataset

Dataset Used	**Experiment ID**	**Neurons/Hidden Layer**	**Accuracy (0 to 1)**
dataset_1	6	3500	0.3425
dataset_2	11	4000	0.375
dataset_3	16	4000	0.4127
dataset_4	18	4000	0.4252

Chapter V: Analysis

In this section, the effects of various hyperparameters on the performance of different models will be discussed, and high level visualizations will be analyzed to draw insights on how the neural network is learning the data. Various datasets were created, and multiple experiments were run on each dataset.

A. Hyperparameters

As explained earlier, hyperparameters are variables that the experimenter can change about the model structure or the data that enters the model. In this subsection the effects of neuron count per hidden layer on the models accuracy will be discussed, as well as the effects of quality of the data that is fed into it.

1. Effects of Neuron Count on Accuracy

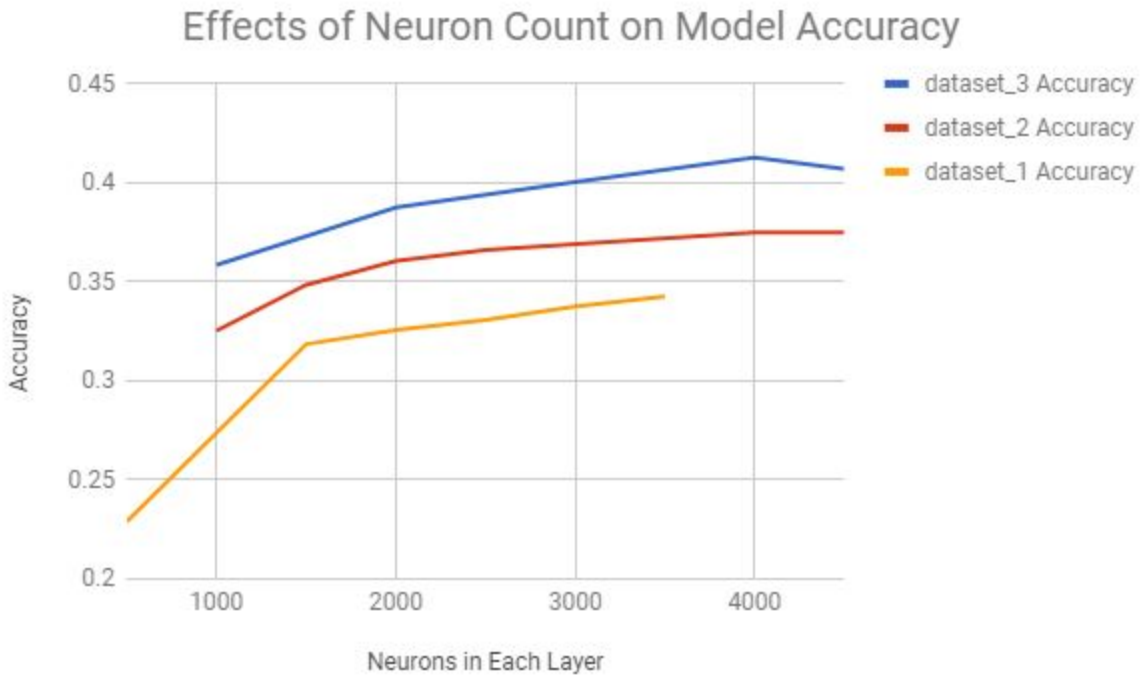

Figure 10: All model tests graphed based on the neurons in the hidden layer compared to the model accuracy after training

It's important to note that the accuracy of the models increased according to how constrained the dataset was. For example, note that as shown in Table 3, dataset 1 had a higher input dimensionality of 1124 possible input skills, and a higher output dimensionality of 58 possible outputs. Figure 10 shows that after constraining the criteria for dataset creation to limit the number of possible inputs and outputs, higher accuracy scores can be reached across the board,

irrelevant of hidden layer neural count. Despite the smaller size of dataset_4 compared to dataset_1, the accuracy is still significantly higher. This implies that the size of inputs and outputs matter much more than the amount of data available for training.

2. Effects of Number of Inputs on Accuracy

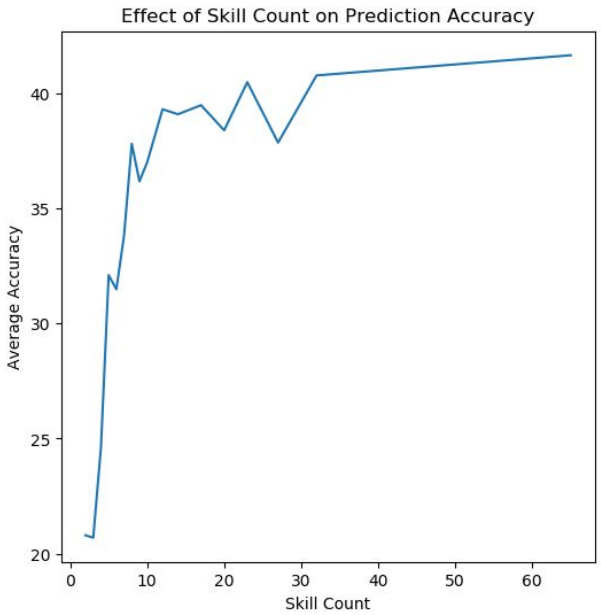

Figure 11: A graph showcasing the effects of how many skills a user profile has on the accuracy of the model

Shown in the figure above is an interesting relationship that was found: users with more skills listed on their profiles will have a higher chance of being accurately labeled by the model. However, there are diminishing returns. Adding more than 10 skills to your profile will highly increase the classification accuracy, but adding more from there will not help as much.

B. Performance Visualizations

Machine learning models are oftentimes referred to as “black boxes” because of how difficult it is to see whether or not the model is learning the concepts behind the data, or simply memorizing. There are clever ways of visualizing the output of a network that can help with this problem by showing in a clear way how the network is conceptualizing the data. This subsection will cover two such techniques: confusion matrices, and T-SNE plots.

1. Confusion Matrix

The first implemented performance visualization was the confusion matrix, an example of which can be seen below as Figure 11. In this method predicted labels are plotted on the horizontal axis and true labels on the vertical axis. Each element in this graph is given a number from 0-1 (represented as a dark color for higher values) relating to the probability that a certain true label is given a predicted label. Thus, a diagonal line of dark boxes would correlate to an accurate network.

The main benefit of the confusion matrix is seeing which labels are likely to be confused. Looking at Figure 11, it can be seen that law practice is likely to be confused with legal service (A). This confusion matrix might indicate that more data is necessary, or that some industries just have large amounts of overlap. Medical practice is also confused with hospital healthcare (B), and computer networking is heavily confused across the board with Information Technology (C). These observations are helpful because they show that when the network fails, it's not because of a failure to learn the concepts behind the data, but rather a failure in the data to be complete enough for perfect classification each time.

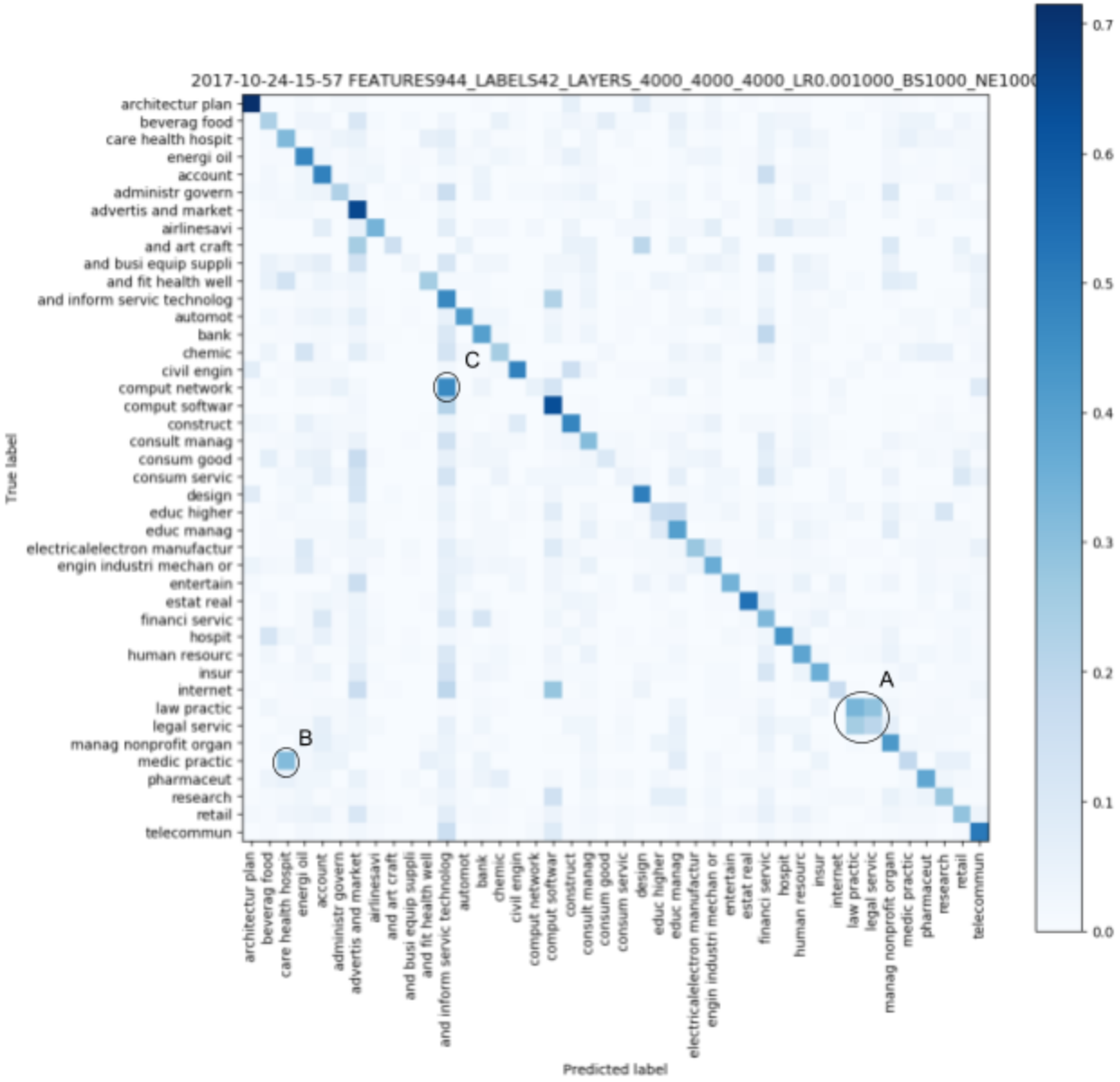

Figure 11: A confusion matrix plot. Darker colors indicate a higher chance a true label is a given a predicted label. Dark boxes along the diagonal indicate skills are accurately predicted as themselves.

2. t-SNE

Another useful visualization that shows the behavior of the neural network is a t-SNE graph. t-SNE is an embedding algorithm that attempts visualize higher dimensional data by mapping it to 2D or 3D space [3]. As shown below in Figure 12, even with an accuracy of 42% the network learned some interesting similarities between industries. For example, the industries Civil Engineering and Construction are very similar as these professions are very similar. Similar logical groupings of labels are consistently seen across the t-SNE graph. A t-SNE graph with a full legend can be seen as Figure A-2. Note that Law Practice and Legal Service are very close together, and that in confusion matrix (Figure X) it was shown that the two are often confused for each other.

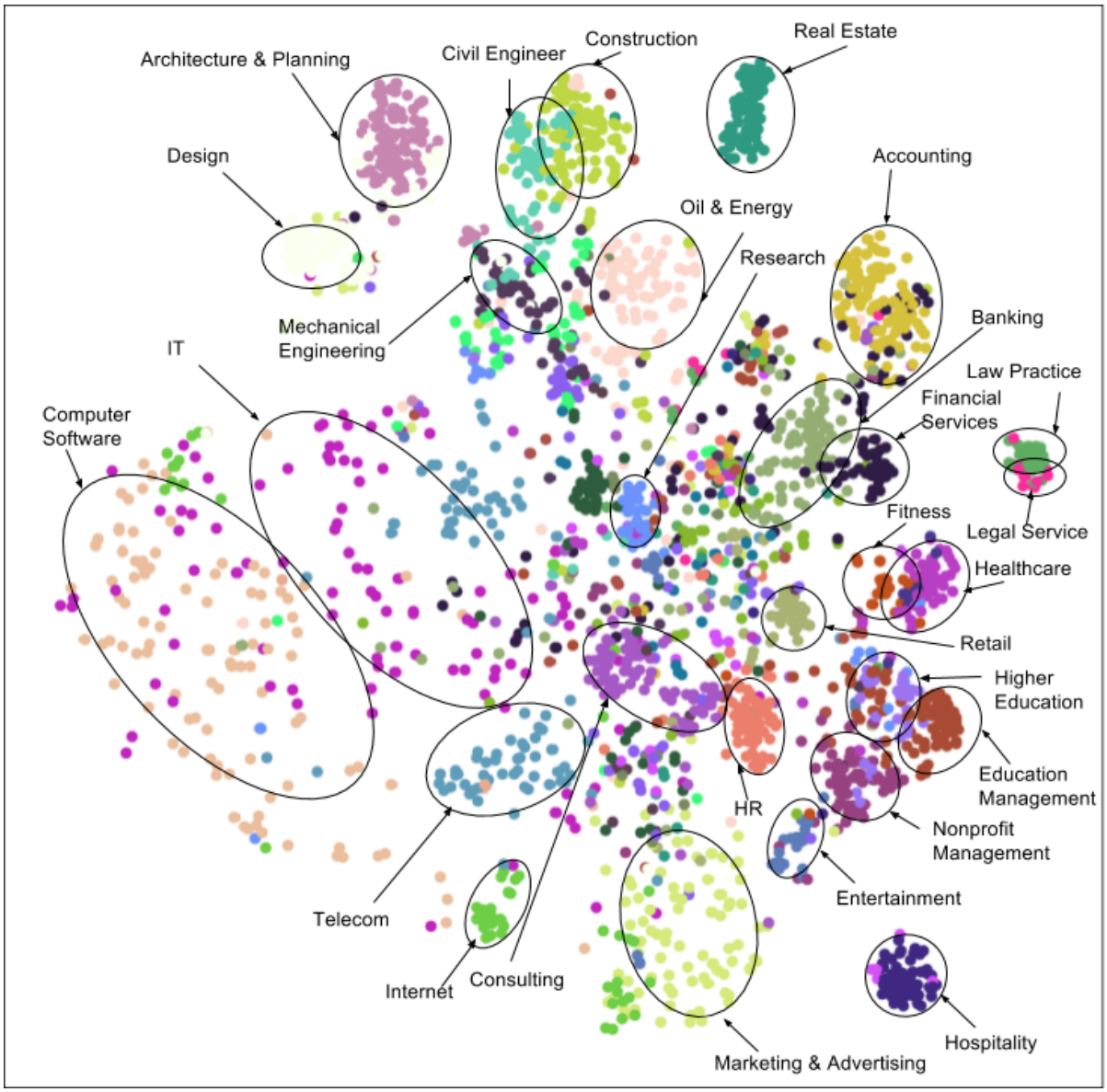

Figure 12: A 2D T-SNE graph plotting all correctly labeled industries. Each color is a separate label. The most cohesive labels have been named.

C. Discussion

"Crawling" LinkedIn in the way this project describes is not recommended as after a period of time login blocks are put in place to prevent the crawler from gathering more data on profiles. This slows the data collection process significantly, nearly to the point of impracticality. Recommended other methods are using pre-collected data sets and/or applying to LinkedIn for the ability to use their data.

The data provided in this project indicates that using deep neural networks to analyze LinkedIn profiles is possible and helpful. A peak accuracy of 43% was achieved with 42 labels that consistently overlap. Possible methods to improve upon these results are by gathering more training examples with an emphasis on profiles with more information.

Along with the achieved accuracy, the model understood similarities between industries. Specifically predicting an industry based upon the skills that a user has input on their profile has been proven to be far better than random choice. It provides many useful relationships, which makes this model a possible recommendation engine, labeling tool, and a helpful tool for identifying what industries to look within for candidates with specific skills.

D. Conclusion

This research project resulted in the creation of a neural network capable of mapping a person's skills to the industry that they work in. The optimal parameters for this data mapping were found, and it was shown that the neural network developed some understanding of the mapping between an input of skills and an output of industry. It demonstrated an understanding of combinations of skills, similarities of industries, and more. This understanding is helpful for those in the recruiting career as it can make finding candidates with proper skills easier, for students and others trying to find a class or career plan that fits their skills, and for LinkedIn to create a industry recommender system that could lead to an easier process for their users to complete their profiles. Proving that mapping between the skill and industry domains leads to the question of: what other domains can be mapped to? Given more data, is it possible to map from skills to current company that a person works at? Another topic that is worth continuing research is the idea that the necessary skills for a particular industry change over time. Since this mapping is not stationary, a model would need to be developed with the capability to adjust to new data over time.

Works Cited

[1] N. Hatch and J. Dyer, "Human capital and learning as a source of sustainable competitive advantage", 2017. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/smj.421/full. [Accessed: 02- Nov- 2017].

[2] H. Escalante and I. Guyon, "Design of an explainable machine learning challenge for video interviews - IEEE Conference Publication", *Ieeexplore.ieee.org*, 2017. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7966320/authors?part=1. [Accessed: 02- Nov- 2017].

[3] T. Zimmermann, L. Kotschenreuther and K. Schmidt, "Data-driven HR - R\'esum\'e Analysis Based on Natural Language Processing and Machine Learning", *Arxiv.org*, 2017. [Online]. Available: https://arxiv.org/abs/1606.05611. [Accessed: 02- Nov- 2017].

[4] V. Menon and H. Rahulnath, "A novel approach to evaluate and rank candidates in a recruitment process by estimating emotional intelligence through social media data - IEEE Conference Publication", *Ieeexplore.ieee.org*, 2017. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7854061/authors. [Accessed: 02- Nov- 2017].

[5] M. Verma, "Cluster based Ranking Index for Enhancing Recruitment Process using Text Mining and Machine Learning", *Pdfs.semanticscholar.org*, 2017. [Online]. Available: https://pdfs.semanticscholar.org/f3f8/23cf20f93b337693024ee9d4f7bceea40123.pdf. [Accessed: 02- Nov- 2017].

[6] A. Sood and R. Bhatia, "Ensemble Recommendation for Employability Using Cattell's Sixteen Personality Factor Model", 2017. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-30927-9_3. [Accessed: 02- Nov- 2017].

[7] B. Patel and V. Kakuste, "CaPaR: A Career Path Recommendation Framework - IEEE Conference Publication", *Ieeexplore.ieee.org*, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7944917/. [Accessed: 02- Nov- 2017].

[8] C. Vialardi, J. Bravo, L. Shafti and A. Ortigosa, "Cite a Website - Cite This For Me", *Files.eric.ed.gov*, 2017. [Online]. Available: http://files.eric.ed.gov/fulltext/ED539088.pdf. [Accessed: 02- Nov- 2017].

[9] M. Uddin, S. Banerjee and J. Lee, "Recommender System Framework for Academic Choices: Personality Based Recommendation Engine (PBRE) - IEEE Conference Publication", *Ieeexplore.ieee.org*, 2017. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7785779/. [Accessed: 02- Nov- 2017].

[10]Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", *Nature*, 2017. [Online]. Available: https://www.nature.com/nature/journal/v521/n7553/abs/nature14539.html. [Accessed: 02- Nov- 2017].

[11] A. Drigas and S. Kouremenos, "An expert system for job matching of the unemployed", *Pdfs.semanticscholar.org*, 2017. [Online]. Available: https://pdfs.semanticscholar.org/db83/98a0a519d0d61d6ddf1fe875c62883c057d4.pdf. [Accessed: 02- Nov- 2017].

[12] V. Nair and G. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines", *Citeseerx.ist.psu.edu*, 2017. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.6419&rep=rep1&type=pdf. [Accessed: 02- Nov- 2017].

[13] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. squared error training: a theoretical and experimental comparison.," in INTERSPEECH, 2013, pp. 1756–1760.

[14] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", *Arxiv.org*, 2017. [Online]. Available: https://arxiv.org/abs/1412.6980. [Accessed: 02- Nov- 2017].

[15] R. Elwell and R. Polikar, "Incremental Learning of Concept Drift in Nonstationary Environments", *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517-1531, 2011.

[16] S. Furao, T. Ogura and O. Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning", *Neural Networks*, vol. 20, no. 8, pp. 893-903, 2007.

[17]S. Marsland, J. Shapiro and U. Nehmzow, "A self-organising network that grows when required", *Neural Networks*, vol. 15, no. 8-9, pp. 1041-1058, 2002.

[18] Medium. (2017). What is the best programming language for Machine Learning?. [online] Available at: https://medium.com/towards-data-science/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7 [Accessed 31 Oct. 2017].

[19] Hayashi, Y., Sakata, M. and Gallant, S. (1990). Multi-Layer Versus Single-Layer Neural Networks and an Application to Reading Hand-Stamped Characters. International Neural Network Conference, pp.781-784.

[20] Hecht-Nielsen, R. (1988). Theory of the backpropagation neural network. Neural Networks, 1, p.445.

[21] Visualizing data using t-SNE. (n.d.). Cost-sensitive Machine Learning for Information Retrieval, [online] 33. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.457.7213 [Accessed 31 Oct. 2017]

Appendix

Dataset Used	Experiment ID	Neurons/Hidden Layer	Accuracy (0 to 1)
dataset_1	1	500	0.2289
	2	1500	0.3185
	3	2000	0.3258
	4	2500	0.3308
	5	3000	0.3375
	6	3500	0.3425
dataset_2	7	1000	0.3253
	8	1500	0.3485
	9	2000	0.3605
	10	2500	0.366
	11	4000	0.375
	12	4500	0.375
dataset_3	13	1000	0.3585
	14	2000	0.3875
	15	3000	0.4005
	16	4000	0.4127
	17	4500	0.407
dataset_4	18	4000	0.4252

Figure A-1: A list of every experiment that was performed, with the best models highlighted

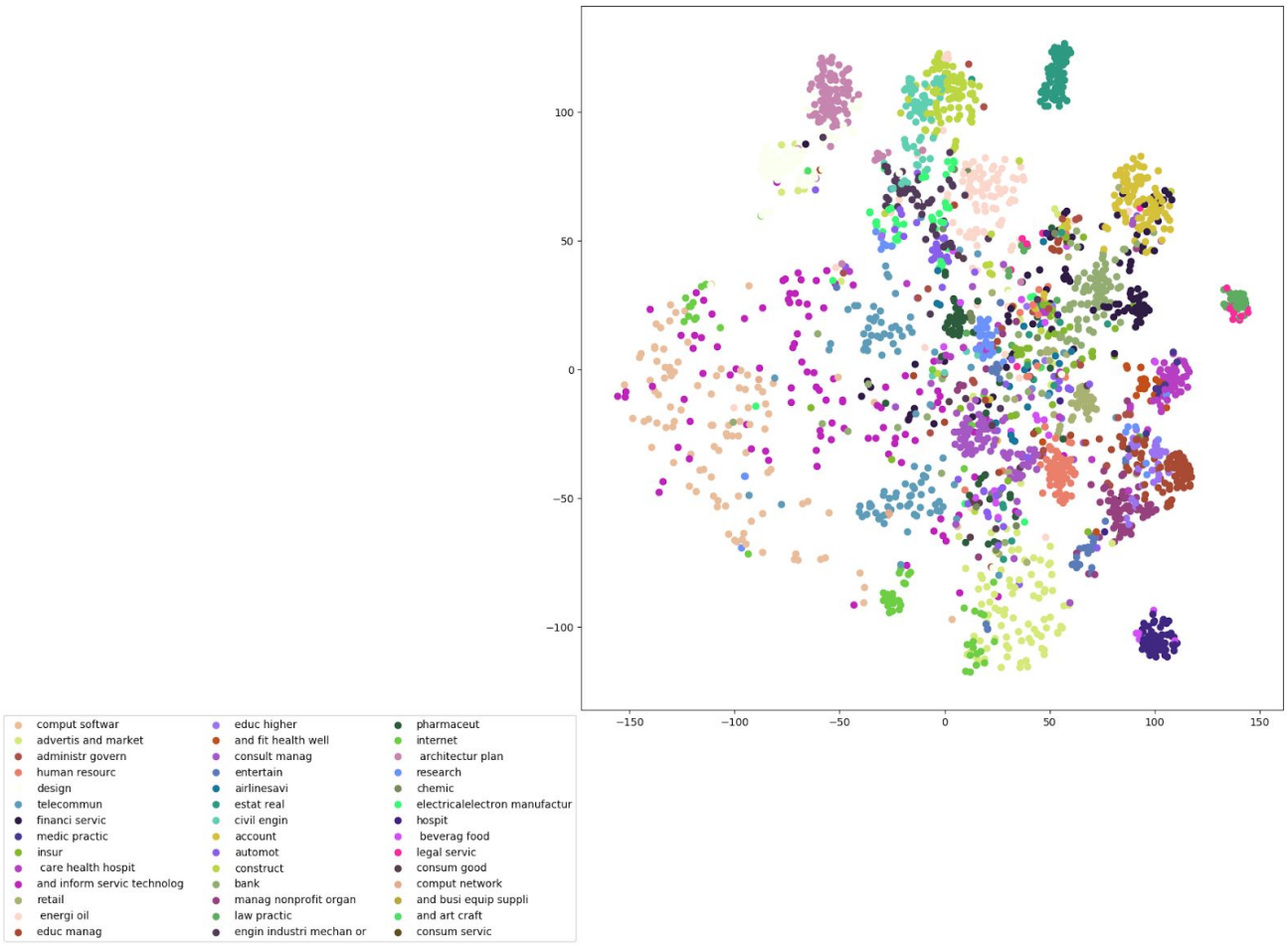

Figure A-2: A t-SNE graph with a full legend. Produced from the model trained on dataset 4 (the most accurate model).