Exploring Automated Methods for Supporting Worker Re-skilling

by

Xiaomin Wang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2020

Author .....
Department of Electrical Engineering and Computer Science
May 12, 2020

Certified by .....
Deb Roy
Professor
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Exploring Automated Methods for Supporting Worker Re-skilling

by

Xiaomin Wang

Submitted to the Department of Electrical Engineering and Computer Science
on May 12, 2020, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

There are plenty job portals (e.g., linkedin.com, indeed.com, ziprecruiter.com etc), that leverage machine learning models to connect employers and job seekers via job or candidate recommendations. However, much less attention is paid to recommending specific skills that would help workers reskill or employers identify how to retrain their employees. This thesis seeks to build a system that recommends skills with the following two properties: 1) recommended skills are similar to a worker's existing skills so they are more likely to try and acquire them; 2) recommended skills increase chances of income enhancement. Existing research has largely focused on building models with employee data such as resumes and LinkedIn profiles. We instead explore the value of much-less-used employer data, i.e. language contained in job postings. The last few years have seen tremendous advances in natural language processing (NLP), including the rise of dense vector representations for text (i.e. "text embeddings") to help solve a plethora of prediction, classification, and other tasks. In building our system, we compare the performance of several language embedding models and skill valuation models to identify and recommend opportunities for re-skilling.

Thesis Supervisor: Deb Roy
Title: Professor

Acknowledgments

This project would not have been possible without Nabeel Gillani. Apart from giving both high-level guidance and technical help throughout the course of this project, he was also a constant source of inspiration and emotional support. I feel extremely lucky to have him as my PhD advisor. And I will be forever grateful to him. I would like to thank professor Deb Roy for giving me a home and an opportunity to meet the wonderful people at LSM. People at LSM came from very different backgrounds and have a wide range of research interests. But they all share a dedication to making the world a better place with technology. I learned a lot from them. Lastly, I want to thank my parents and friends for their continuous support and encouragement.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

The current wave of technological revolution, especially advancements in artificial intelligence, has profound impact on the labor market. Understanding this impact and designing policies that mitigate any challenge society faces are of great interest to researchers, policy makers, and educators. One piece of the puzzle lies in shifts taking place in workplace skills. Job skill requirements are changing so quickly that many people will not able to keep up, and are at risk of being shut out of the workforce. A report by MIT's Task Force on the Work of the Future [16] argued that the challenge is not a lack of jobs, but unequal access to opportunities between high-skilled and low-skilled workers [1] [2]. Constantly learning and training for new skills is the only way to survive this demanding time. Frank et al. [15] identified a shortage of up-to-date and high resolution skills data as a barrier to their "scientific modeling of technological change and the future of work". Data (O*NET database, Current Population Survey from the U.S. Census Bureau and the BLS, etc.) available to researchers does not keep pace with changes in the labor market. Statistics are often at an aggregated level, making it difficult to design specific and actionable strategies. Many researchers also lack the resources and expertise to build useful tools that can realize their insights.

On the other side of the world, career service providers hold an massive amount

of highly dynamic data. They are also capable of rapidly releasing products that reach a large audience. They are financially incentivized to fill the needs in the labor market, which eases some pain workers experience in this challenging environment. These companies seem well positioned to solve the problems faced by workers today. However, the problem they are eager to solve is limited to helping employers find qualified candidates. Supporting workers to re-skill takes much longer and is much more difficult to monetize. As a result, society is left with a critical problem, and at the same time, resource that may hold the answer. Yet, not enough people are taking a stab at the matter. We decided to become part of this group. The way we want to help is by building a system that recommends skills to workers.

1.2 Related Work

Two lines of research are highly relevant and helpful to this project. One in skill extraction [9] and normalization [13, 7]. The other in job recommendation. A large body of work has been dedicated to building job recommender systems. [25] compared various content-based systems that try to match user and job attributes. [18] recommended jobs that similar users have applied to (a form of user-based collaborative filtering). Graph-based models were explored in [23, 11]. Though skills are intricately tied to jobs, recommending skills has a very different objective. In the case of job recommendations, users are more interested in positions they qualify for right now. But our goal is to find skills that allow users to qualify for more jobs in the future. [10] made use of skill embedding, but their objective is to predict someone's next career move, again different from ours. [5] proposed a representation learning framework that recommends both jobs and skills. The representations are learned only using data on job and skill interaction, and job to job transition, while we plan to also utilize the rich text content of job postings.

"DATA AT WORK" project [1] tries to build a more open and interoperable workforce data ecosystem. The project implemented an open-source `Python` library, Skills-

[1] http://dataatwork.org/

ML [21], for competency extraction and occupation classification from unstructured text data. Their work has much overlap with this thesis project in terms of data source and methodology. By the time we found out about this project, we had already completed much ground work the Skills-ML library is intended for. It was unfortunate some work was duplicated, but encouraging to see similar approach being adopted by other people. Using the data and toolkit released by the "DATA AT WORK“ project, researchers at University of Michigan built and evaluated an application, DreamGigs [6], that identifies skills a job seeker needs to develop to achieve their dream job, and suggests work opportunities where these skills may be acquired. This work is very user-oriented, presenting a great example of how a system like ours can be turned into something useful in the real world.

1.3 Our Approach

The main objective of this project is to build a skill recommender system that helps workers looking to re-skill. The recommended skills should satisfy the following two properties: 1) recommended skills are similar to a worker's existing skills so they are more likely to try and acquire them; 2) recommended skills increase chances of income enhancement. The first property calls for a model that learns a numerical representation for skills so their similarity can be measured. The second property requires assigning quantitative values to skills.

Most of the models that drive our recommendations are built with online job posting data. A large body of previous work on employment tools used career transition data, probably because such data can be easily converted to matrices, a type that works with most machine learning algorithms. Text is unstructured, and has to go through many pre-processing steps before it can be passed into a machine learning model. This barrier leaves job postings an under-utilized source of data for building employment applications. Recent progress in text embedding models presents an opportunity to build better machine learning systems with job posting data. On the other hand, employer data such as job postings is better at representing needs in

the labor market in real time than employee data such as career transitions, further motivating the use of job posting data. Recruiters often use the most space in a job posting to describe the tasks and skills required to be successful at the position. As a skill can be defined by the tasks it performs, the text in a job posting encodes important characteristics of a skill. Learning embeddings for job postings, thus, not only helps finding similar jobs, but also similar skills. If two skills are similar, text that describes them should have resemblance that can be captured by vectors that summarize the text content.

Before actually building the recommender system, we combined information from multiple sources to create a list of skills as recommendation candidates. We present details of this process in Section 2.2.3. The system consists of several components, a job posting encoder, a skill tagger, a salary estimator, a similar occupation finder, and a recommendation engine. Each job posting is converted into an embedding by the job posting encoder. The skill tagger identifies a list of skills from the job posting. Each job posting belongs to a job title. We average the embeddings of the job postings with the same job title to derive a vector representation for the job title. We also average the embeddings of the job postings that are tagged with the same skill to derive a vector representation for the skill. The system takes in a job title, the similar occupation finder then finds the most similar job titles that also pay better, by comparing the distances calculated from the vector representations of job titles. Lastly, the recommendation engine outputs a list of recommended skills by comparing skills required by the input job title and those required by similar titles. The recommended skills are restricted to a list of skills close to those required by the input job title, where closeness is measured using the vector representations of skills.

We supplement the model we just described with a recommendation model based on the job-skill interaction matrix. In this model, we first construct the job-skill interaction matrix from the output of the skill tagger. We then learn an embedding for skills from the interaction matrix. The salary estimator either extracts the salary figure directly from the job posting, or estimates based on similar job postings. We assign values to skills using the coefficients of a salary predictor. This model takes in

a skill, and recommends skills based on skill similarity and skill value.

We elaborate the various parts of the system in Section 3.

Chapter 2

Data

In this section, we describe in detail where data used in this project was collected, and how it was processed.

2.1 Data Sources

2.1.1 O*NET

The Occupational Information Network (O*NET) [1] was a program sponsored by the US Department of Labor/Employment and Training Administration. It collected and organized a variety of occupational information and made the data freely available to the public. The data has been an invaluable resource for job seekers, policy makers and social researchers, or anyone interested in the labor market, as it has created standardized language for interested parties to engage in discussions. Another strength of the data is that it is continuously updated from input by a broad range of workers in each occupation. We used the following datasets from the O*NET database:

- **Occupation Data**: this file contains 1,110 job titles (which we will call O*NET Titles) with their O*NET-SOC Code and descriptions.
- **Job Zones**: this file contains 969 job titles and their corresponding Job Zone numbers. Job Zone groups occupations based on their associated levels of edu-

[1] https://www.onetonline.org/

cation, experience, and training. The values range from 1 to 5. The higher the value, the more preparation the job requires.

- **Knowledge**: this file contains domains of knowledge and their importance levels associated with O*NET Titles. There is a total of 33 unique domains. Some examples are 'Law and Government', 'Physics', 'Economics and Accounting'.

- **Skills**: this file contains general skills and their importance levels associated with O*NET Titles. There is a total of 35 unique skills. Some examples are 'Reading Comprehension', 'Negotiation', and 'Troubleshooting'.

- **Abilities**: this file contains abilities and their importance levels associated with O*NET Titles. There is a total of 52 unique abilities. Some examples are 'Memorization', 'Arm-Hand Steadiness', and 'Inductive Reasoning'.

- **Technology Skills**: this file contains technology skills associated with occupations found in **Occupation Data**. There is a total of 8,824 unique technology skills, each comes with a classification under the United Nations Standard Products and Services Code (UNSPSC). An example

- **Tools Used**: this file contains tools associated with O*NET Titles. There is a total of 18,172 unique tools, each comes with a classification under UNSPSC. An example

2.1.2 Indeed

Indeed [2] is an American worldwide employment-related search engine for job listings launched in 2004. The site also has salary data on different occupations calculated based on user input and job advertisements in the past 36 months at any given time. For occupations with enough data, salary figures are also averaged over cities and states, in addition to the national average.

[2] https://www.indeed.com/

2.1.3 Wikipedia

Wikipedia [3] is "a multilingual online encyclopedia created and maintained as an open collaboration project by a community of volunteer editors, using a wiki-based editing system".

2.2 Data Preparation

We used aforementioned sources to create three sets of data: salaries, job postings and skills. O*NET data was downloaded directly from its website. Indeed data was mined using a custom scraper implemented in `Python` package `scrapy`. Wikipedia data was retrieved using `Wikipedia-API`, a `Python` wrapper for Wikipedia's official API.

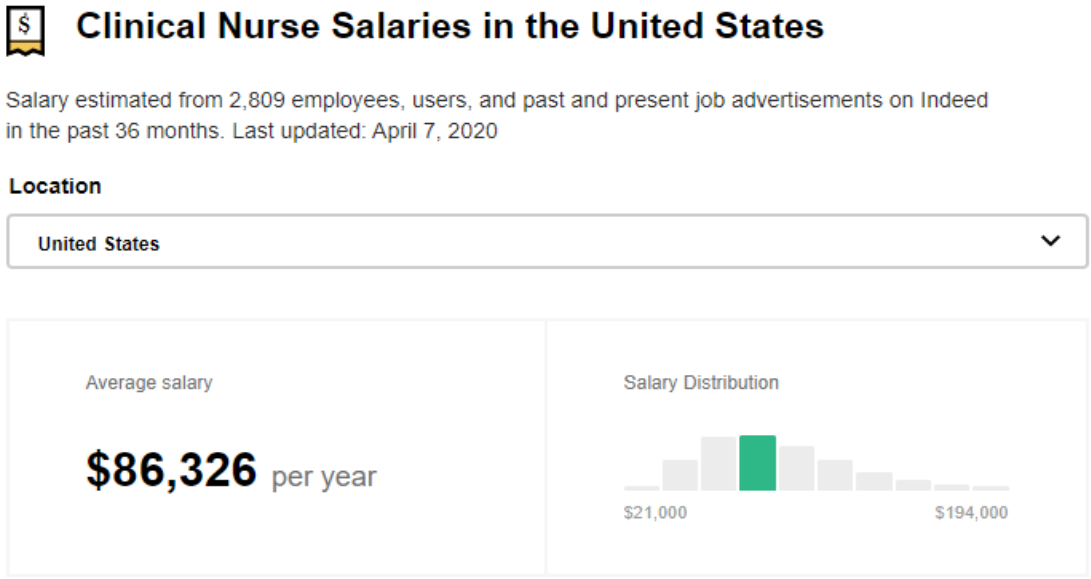
2.2.1 Salaries

Indeed users can search salary figures by job titles. If the searched job title exists in Indeed's database, users will find average salaries at various locations in the United States by selecting different values in the location dropdown menu (Figure 2-1a) . If the searched job title does not exist, users will instead find links to related job titles (Figure 2-1b). We wrote a scraper that programmatically searched salary figures for O*NET Titles. State and city figures were saved when available. Salary can be quoted yearly, monthly, weekly or hourly. We converted all numbers to yearly assuming 40 hours per week, and 52 weeks or 12 months a year. The scraper would also follow any link to related jobs it could find. We ended up with 4,714 distinct job titles (which we will call Indeed Titles) with salary information.

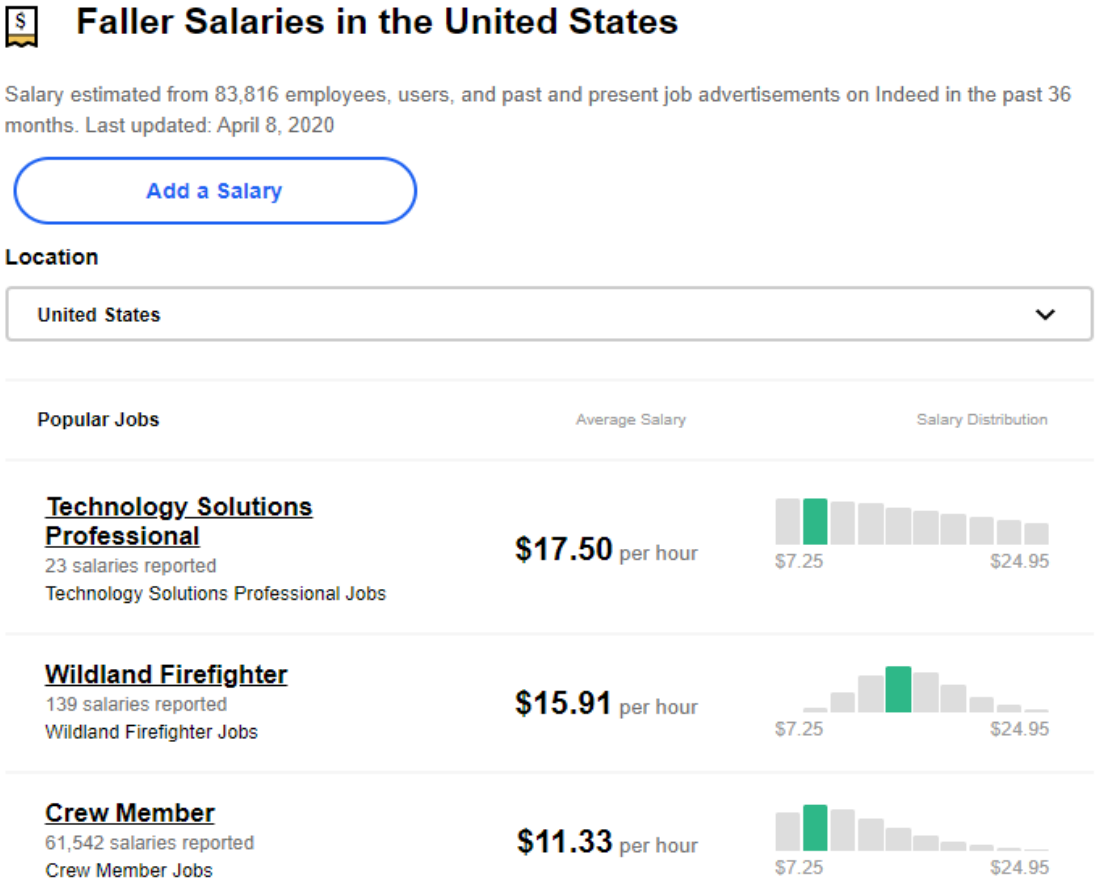
2.2.2 Job Postings

We wrote a scraper that scraped the first 10 pages of job postings for each Indeed Title. For each posting, the scraper extracted the position title, the company name,

[3] https://en.wikipedia.org/



(a) Salary Page When Title is Found



(b) Salary Page When Title is not Found

Figure 2-1: Indeed Site When Searching Salary by Job Title

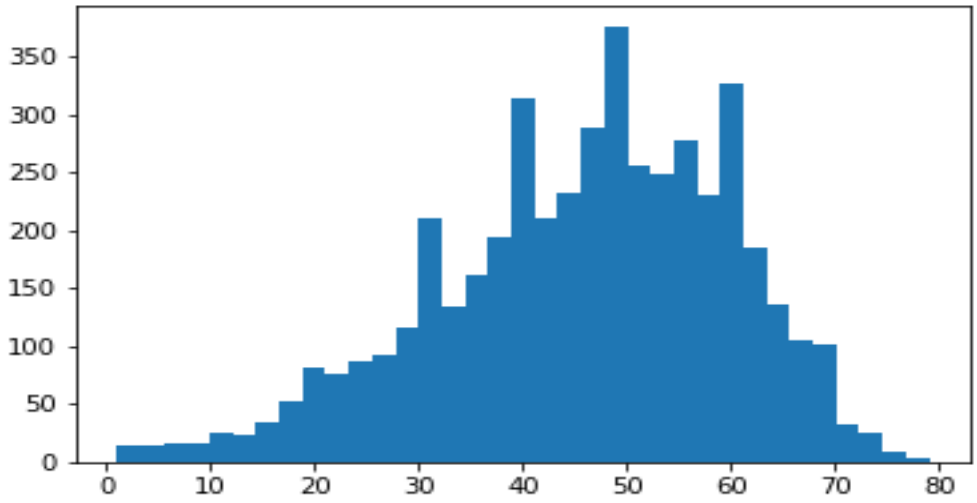


Figure 2-2: Histogram of Number of Jobs per Indeed Title

the location, the job summary, the detailed job description, and the posting url. 28% of the scraped postings included expected salary in the job description. When salary is not explicitly stated, we estimated salary to be the average for the Indeed title at the smallest available location. For example, if the average salary for a plumber in Boston, MA was not available, we would use the average for Massachusetts, if still unavailable, we would use the national average. We ran the scraper in the summer of 2019, and got 211,490 different job postings. Two postings are considered different if any one of the extracted fields is different. Figure 2-2 shows a histogram of number of jobs per Indeed Title.

2.2.3 Skills

The set of skills we used in this project is the union of the following sets of skills:

- All knowledge, abilities, general skills, technology skills and their UNSPSC categories, tools used and their UNSPSC categories from O*NET. We will call this set of skills O*NET Skills.

- We had previously scraped job postings on Indeed in 2018. At the time, each

posting came with a list of skills Indeed deemed necessary for the job. Indeed had disabled this feature when we scraped in 2019. Out of 253,680 postings we scraped in 2018, there were only 1,114 distinct skills, most of which did not appear in O*NET Skills. We will call this set of skills Indeed Skills.

- We ran Google search using `Python googlesearch` package for each of O*NET Skills and Indeed Skills. If one of the top 20 search results was a Wikipedia page, we would save the title of the first Wikipedia page result as a skill. We will call this set of skills Wiki Skills.

Chapter 3

Models

3.1 Job Posting Encoder

The first part of the system involves converting each job posting (only the position title and the job description) into an vector of real numbers, also called an embedding. We experimented with 4 models that generate numerical representations from text.

3.1.1 Tf-idf

Tf-idf stands for term frequency-inverse document frequency. It measures how important a word is to a document in a collection of documents. In this case, a document is a job posting. We calculated tf-idf using `scikit-learn`[1], a `Python` machine learning library. Please refer to the `scikit-learn` documentation for its implementation details.

Preprocessing. We took the following steps to preprocess the job postings before calculating their tf-idf scores:

1. Find all skills (Section 2.2.3) that contain punctuation or numbers. Keep words that match these skills in the job postings as they are. For the rest of the job posting, do:

2. Remove all punctuation and numbers.

[1] https://scikit-learn.org/

3. Split text into words that are separated by space.

4. Convert all words to lower case.

5. Remove all stop words as defined in `nltk`[2], a `Python` library for natural language processing.

6. Lemmatize words using `spacy`[3], another `Python` library for natural language processing.

Furthermore, we passed `max_df=0.6, min_df=5` as parameters when training tf-idf vectorizer, effectively removing words that appear too often (more than 60% of all job postings) or too infrequently (less than 5 job postings).

After training, the tf-idf model learned a vocabulary of size 60,095. Thus each job posting would be represented by a vector of length 60,095.

3.1.2 Universal Sentence Encoder

Universal Sentence Encoder(USE) [4] is a model developed at Google that converts longer-than-word text into a 512 dimensional vector. The encoder was trained simultaneously on a variety of supervised and unsupervised tasks such as semantic similarity classification, entailment, and surrounding sentences prediction. The training data came from Wikipedia, web news, web question-answer pages and discussion forums.

3.1.3 BERT

Bidirectional Encoder Representations from Transformers [8] (BERT) is another language representation model developed at Google. At the time of its publication, BERT achieved state-of-the-art performance on a number of natural language understanding tasks. BERT was trained on the Wikipedia corpus for two tasks: masked

[2]https://www.nltk.org/
[3]https://spacy.io/

word prediction and next sentence prediction. The result is a pre-tranied word encoder that generates a high dimensional vector for each word given a piece of text. The use of transformer, an attention mechanism, allows BERT to be context-aware, which means the same word could have different representations in different sentences.

We used a `Python` implementation of BERT by HuggingFace [24] to generate job embeddings, which are vectors of length 768, calculated as the average of the posting's word embeddings.

3.1.4 Doc2Vec

Le and Mikolov[17] proposed an algorithm to learn paragraph vectors as an extension to word vector models [14][22]. In a typical word vector model, word embeddings are learned by maximizing the probability of predicting the next word given a sequence of previous words. The paragraph vector model adds another vector representing the paragraph to the training sequence.

Gensim[4] implemented a framework, Doc2Vec, to train paragraph vectors. We treated each job posting as a paragraph. The model was trained for 20 epochs to produce vectors of size 1000. Similar to tf-idf, we removed words that appeared less than 5 times.

3.1.5 Job Similarity

We want the best encoding model capable of identifying job postings that are similar to each other. In other words, the vector representations of two similar job postings should be closer in distance than the vectors of two dissimilar job postings. We verified that all 4 models were able to do that to a reasonable extent. We randomly selected 200 Indeed Titles. For each of these titles, we got the job postings that belonged to the title (group A), as well as an equal number of postings selected (without replacement) randomly from all job postings (group B). For each encoding model, we measured the pair-wise cosine distance for postings within group A and

[4]https://radimrehurek.com/gensim/

Model	Random	Same Title
tfidf	0.949	0.817
USE	0.41	0.279
BERT	0.163	0.13
doc2vec	0.871	0.783

Table 3.1: Average Distance between Postings with Same Title vs Random Postings

Model	% Titles with p-value $< 5\%$
tf-idf	100%
USE	97.5%
BERT	91.5%
Doc2Vec	98%

Table 3.2: % of 200 Indeed Titles Rejecting Null Hypothesis in Welch's t-test

postings within group B. The cosine distance between two job embeddings, E_{j1} and E_{j2}, is calculated as in Equation 3.1. We report the average distance over 200 titles in Table 3.1. We also performed a Welch's t-test on the distances from group A and group B. Table 3.2 shows the percentage of the 200 Indeed Titles that reject the null hypothesis at 5% significance level, the null hypothesis being the average distance between two job postings with the same Indeed Title is the same as or larger than between two random postings.

$$distance(E_{j1}, E_{j2}) = \frac{E_{j1} \cdot E_{j2}}{\|E_{j1}\| \, \|E_{j2}\|} \tag{3.1}$$

We also ran an Amazon Mechanial Turk (MTurk) job to further evaluate the 4 encoding models. Tf-idf was selected as the best model based on MTurk results. We elaborate on the experiment setup and analysis in section 4.1. For the rest of the project, we used tf-idf embeddings to represent job postings. We then converted each Indeed Title to the average tf-idf embeddings of all job postings that belong to the title. Table 3.3 has some examples of which Indeed Titles are most similar to some reference titles according to this representation.

Reference Title	Similar Titles
food clerk	sales clerk, store clerk, bakery clerk, general merchandise clerk, food service associate
health services administrator	program administrator, behavioral health professional, health information management manager, registered nurse, medical director
senior process engineer	process engineer, senior manufacturing engineer, senior lead engineer, development operations engineer, facilities engineer ii
metal worker	metal fabricator, sheet metal mechanic, entry level production worker, recycling worker, general worker
security guard	residential security officer, security supervisor, armed security officer, guard, security escort

Table 3.3: Examples of Similar Indeed Titles Based on Tf-idf Encoder

3.2 Skill Tagger

An important part of this project was to figure out what skills a job requires. The O*NET database does provide a mapping from O*NET Titles to O*NET Skills. However, our set of job titles and skills is larger than that of O*NET's. We needed a way to do our own mapping.

The plan was to aggregate skills from job postings grouped by Indeed Titles. That required each job posting tagged with a list of skills. At first, we attempted to train a model that predicts required skills from a job posting, using the Indeed data we scraped in 2018. We assumed the skills that came with each post were labeled by the recruiter that wrote the post, and hence the ground truth. Only after spending much effort on building a skill prediction model, we realized some skills were clearly mislabeled. For example, we found a company listed martial arts as a required skill for multiple unrelated positions. It was because the company wrote in all its job descriptions that the gym in its building offers martial arts lessons. Without well labeled data, we could not really build an intelligent machine learning model. We resorted to a much simpler algorithm, string matching. A skill is considered necessary for a job if it is mentioned in the job posting.

We took a few steps to improve the quality of string matching. Word boundary

was enforced so that "java" would match "java is an object oriented language", but not "javap is another tool in JDL bundle". Matching was case insensitive unless the skill contained only capital letters. For example, the tagger doesn't consider 'spice' or 'Spice' a match for SPICE (a general-purpose, open-source analog electronic circuit simulator), but both 'SharePoint' and 'Sharepoint' are considered a match for 'sharepoint' (a Microsoft web-based collaborative platform). To reduce false positive, we removed the following ambiguous (often mislabeled) skills from our skill list:

(transportation, recruiting, square, principle, monitors, interviewing, route, smart, https, rules, security, network, self, impact, maintenance, auto, player, basic, levels, experience, bridge, impact, smart, code, policy, stage, travel, certain, reduce, tax, platforms, forms, skill, go, fear, rest, google, screens, bar, keys, facebook, instagram, twitter, windows, picks, tips, frames, televisions, principal, guides, s, j, r, c, html)

After going through all job postings, 4,676 skills appeared in at least 5 postings. These 4,676 skills constitute our final skill list. Any skill we recommend will come from this list. The set of skills we map to an Indeed Title T, which we will call $\mathbf{S}_T$, is the union of skills that are matched to at least 3 out of all job postings that belong to the Indeed Title.

3.3 Skill Similarity

Similar to job similarity, skill similarity can be calculated as the cosine distance between skill embeddings. We introduce 2 skill embedding methods.

3.3.1 Text Embedding

The way a skill is mentioned in a job description can reveal properties of the skill. Since we already have the embeddings of the job postings, a skill can simply be represented as the average embedding of the jobs that mention the skill. We only took the embedding of sentences that the skill appeared in, as most of other sentences in the job posting are unrelated to skills.

3.3.2 Job-Skill Interaction Matrix

We construct a job-skill interaction matrix, M, of shape 211,490 (number of job postings) by 4,676 (number of skills). $M_{j,s}$ is 1 if job j mentions skill s, 0 otherwise. Since two similar skills are likely to appear in the same job postings, representing a skill as its corresponding column in the job-skill interaction matrix can help find similar skills. However, having a vector size of 211,490 is not scalable. A lower-rank vector representation can be obtained by performing truncated singular value decomposition (SVD) on the interaction matrix. The idea is borrowed from latent semantic analysis, where SVD yields a compact representation of documents from a term-document matrix. We chose 200 as the dimension of the output vector. The top 200 singular values explain 67% of the variance in the original matrix.

3.4 Skill Recommendation

We present 2 skill recommendation models based on the 2 skill embedding methods.

3.4.1 Text Embedding

Given an Indeed Title T, find top 10 similar (based on tf-idf embeddings) Indeed Titles that have higher average salaries than T. For each similar title T', find the skills that are mapped to this title, but not to T, i.e. $\mathbf{S}_{T'} - \mathbf{S}_{T}$, and assign each of these skills a value of the salary difference divide by the number of missing skills. For skills that appear in multiple similar titles, the values accumulate. We filter the skills by a similarity threshold, de-duplicate using Wikipedia, and then recommend the top ones sorted by value. The pseudocode is listed in Algorithm 1.

3.4.2 Job-Skill Interaction Matrix

Running truncated SVD on the job-skill interaction matrix M, gives each skill an embedding of length 200. Let the full embedding matrix of 4,676 skills be ES. Then we can find a low-rank approximation of M by $M \cdot ES$. We will call this approximation

Algorithm 1 Text Embedding Recommendation

procedure RECOMMEND(T)
    similar_titles = empty list
    while size(similar_titles) $<$ 10 **do**
        T' = next closest Indeed Title
        if salary(T') $>$ salary(T) **then**
            similar_titles.add(T')
    skill2value = empty dictionary
    for T' in similar_titles **do**
        missing_skills = $\mathbf{S}_{T'} - \mathbf{S}_T$
        for S in missing_skills **do**
            value = $\frac{(salary(T') - salary(T))}{size(missing\_skills)}$
            if S in skill2value **then**
                skill2value[S] = value
            else
                skill2value[S] = skill2value[S] + value
    similar_skills = empty list
    for S in $\mathbf{S}_T$ **do**
        similar_skills.add(S's 10 most similar skill)
    skills2recommend = empty list
    Sort skills in skill2value by value
    for S in skill2value **do**
        if S in similar_skills and not duplicated in skills2recommend **then**
            skills2recommend.add(S)
    Return skills2recommend

Skill	Value	Skill	Value
cpa	15068.4	carts	-9320.4
senior management	11461.2	forklift	-9252.0
physics	11413.0	laundry	-8551.1
senior leadership	11231.6	cash handling	-8099.8
python	11092.8	first aid	-8008.7
strategic planning	10601.4	housekeeping	-7967.5
saas	10503.4	merchandising	-7859.6
design	10158.9	photography	-7073.1
architecture	9889.9	receptionist	-7046.5
machine learning	9680.3	food service	-6977.1
underwriting	9652.1	front desk	-6878.4
registered nurse	9523.0	clerical	-6814.4
software development	8582.1	event planning	-6543.9
information security	8346.5	sanitation	-6536.0
management experience	8124.0	data entry	-6288.1

Table 3.4: Examples of High and Low Value Skills With Dimension Reduction

matrix M', and it is of shape 211,490 by 200. Let Y be the vector of salaries for the job postings. We built a ridge regression model using M' to predict Y. The objective function for the ridge regression is shown in Equation 3.2, where θ contains the regression weights and is of length 200. λ is set to 10 based on cross-validation. The dot product between ES and the optimal θ can be interpreted as the estimated values for skills. Table 3.4 lists the 15 most and least valuable skills based on $ES \cdot \theta$. We could run ridge regression directly with M instead of M', and take the regression coefficient as the value for a skill. However, the values are less interpretable (ranging from 60,000 to -50,000). The highest and lowest valued skills (Table 3.5) also seem less consistent with common sense.

Now we have both the embeddings and the values for skills. Given a skill S, we can find a list of skills that have high cosine similarity with S and sort them by their values.

$$Loss_{ridge} = \frac{1}{211490}(M'\theta - Y)^T(M'\theta - Y) + \lambda \|\theta\|^2 \tag{3.2}$$

Skill	Value	Skill	Value
intraoral cameras	65964.8	rangefinders	-52155.3
bc/be	61733.9	backbone.js	-42681.0
pupillometer	47822.7	zookeeper	-38281.4
dea certification	47400.5	encryption software	-31737.5
telephone test set	45698.7	marklogic	-31442.5
penultimate	45410.0	ppm tools	-28419.4
abap	43954.8	watir	-27943.9
heavybid	40339.4	pharmacy intern license	-26933.7
licensed psychologist credentials	38919.6	deacom	-26559.5
psychiatric mental health nurse	38179.5	wolters kluwer	-24044.6
ixl learning	37343.4	web server software	-23709.3
sas jmp	33196.4	siemens plm software	-23335.3
blackbaud crm	32889.4	sendgrid	-22809.6
cargo dollies	32687.1	awk	-22481.8
jda software	31775.5	patient restraints	-22421.0

Table 3.5: Examples of High and Low Value Skills Without Dimension Reduction

Chapter 4

Results

In this section, we detail the results of the job posting encoder experiment, as well as output from the recommendation system.

4.1 Amazon Mechanical Turk

We ran an MTurk job to compare performance of the 4 text embedding models at identifying similar job postings. We first chose 59 O*NET Titles from different job zones. More specifically, 11 from Zone 1 (lowest education and training requirement), 12 from Zone 2, 13 from Zone 3, 11 from Zone 4, and 12 from Zone 5 (highest education and training requirement). We then found 59 Indeed Titles (Table 4.1) similar to the O*NET titles. We randomly selected a job posting for every Indeed Title. The 4 embedding models picked the job posting closest to each of the 59 selected posts. When choosing Indeed Titles, we made sure the 4 embedding models would pick different posts.

MTurk workers were given 3 job postings, 1 as reference, the other 2 selected by 2 of the 4 models. The workers were then asked to choose which one out of the two postings is more similar to the reference, as well as the reasons for their choice. Figure 4-1 shows an example task a worker got. Following this setup, a complete ranking of 4 models requires 6 comparisons. We decided to run binary comparisons instead of ranking the 4 models directly, hoping an easier task would improve the quality of

Job Zone	# Titles	Titles
1	11	seamstress, assistant cook, quarry manager, drain technician, crop manager, rental agent, farm laborer, roustabout, sorter, crossing guard, parking attendant
2	12	yard worker, cake decorator, extraction technician, plasterer, metal fabricator, emergency dispatcher, bus driver, child life specialist, lumber associate, cargo agent, meat cutter, press operator
3	13	senior legal assistant, php developer, laser technician, police officer, test engineer, tool and die maker, massage therapist, hair stylist, satellite installer, funeral director, senior electrician, diesel technician, insurance broker
4	11	environmental compliance specialist, human resources specialist, clinical laboratory scientist iii, broadcast engineer, manufacturing engineer, industrial engineer, energy consultant, distribution manager, multimedia specialist, religious education teacher, quality assurance engineer
5	12	ecologist, elementary school teacher, economist, recovery coach, senior research associate, computer technician, clinical study manager, speech therapist, historian, surgeon, epidemiologist, radiologist

Table 4.1: Selected Indeed Titles for Mturk

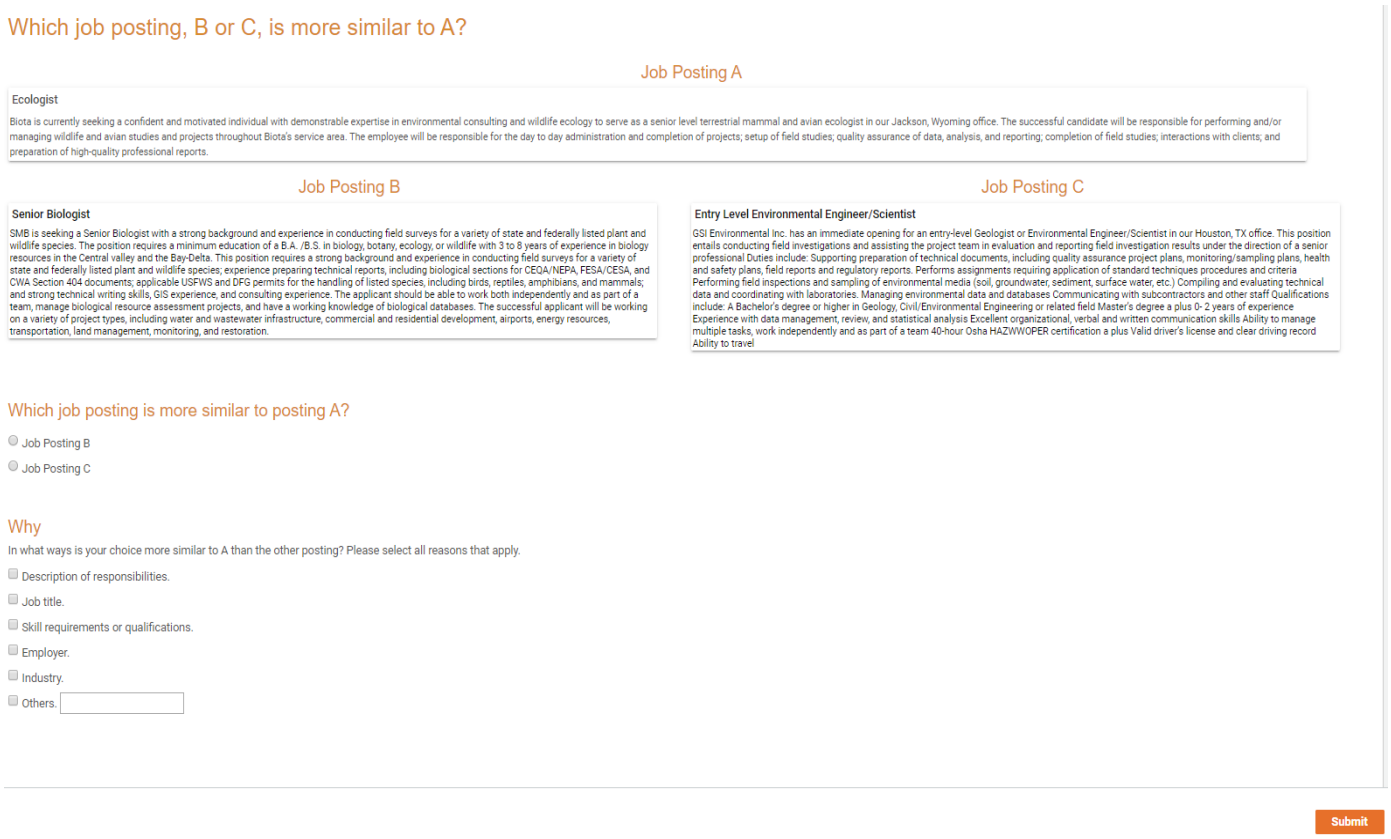


Figure 4-1: Example MTurk Task

worker answers. Another strategy we took to ensure quality was to have 3 workers labeling the same task. The majority answer was treated as the truth.

We didn't publish all MTurk tasks at the same time. After gathering results for 34 titles, we noticed that Doc2Vec had consistently poor performance. Thus for the rest of 25 titles. We only compared tf-idf, USE, and BERT to save cost.

We made a number of observations:

- Tf-idf is overall the best model. We counted pair-wise wins and losses (Table 4.4). Tf-idf has the highest win rate (Table 4.5), followed by USE, BERT, then Doc2Vec. Averaging MTurkers' votes across Indeed Titles (Table 4.2, 4.3) gave us the same ranking.

- When tf-idf did poorly, it was more likely to be a job in high O*NET Job Zone (4 and 5). Table 4.6 lists the Indeed Titles that tf-idf received 0 or 1 vote from MTurkers.

Model	tf-idf	USE	BERT	Doc2Vec	All Models
Average Votes	6.1	4.7	4.0	3.2	4.5
Standard Deviation	2.2	2.3	2.1	2.5	2.5

Table 4.2: MTurker Votes by Model for first 34 Indeed Titles

Model	tf-idf	USE	BERT	All Models
Average Votes	4.1	2.7	2.2	3
Standard Deviation	1.8	1.7	1.6	1.8

Table 4.3: MTurker Votes by Model for the Rest of 25 Indeed Titles

- Responsibilities and skills are the most common reason for job similarity (Table 4.7). It shows that people consider skills as a key distinguishing factor in a job posting, which lends support to our belief that job postings encode important information about skills.
- We didn't find significant difference between the 4 embedding models regarding reasons for job similarity.
- We didn't find significant difference between the 5 Job Zones regarding reasons for job similarity.

4.2 Skill Recommendation

We qualitatively evaluated the recommendations made by our system. Table 4.8 4.9 4.10 are some examples of top 10 recommendations from the text embedding model (Section 3.4.1). In addition, the system points out related occupations that pay better

Model1	#wins	Model 2	#wins
tf-idf	28	Doc2Vec	6
tf-idf	45	BERT	14
tf-idf	39	USE	20
USE	23	Doc2Vec	11
USE	34	BERT	25
BERT	20	Doc2Vec	14

Table 4.4: Pair-wise Wins and Losses

Model	Win Rate
tf-idf	74%
USE	51%
BERT	39%
Doc2Vec	30%

Table 4.5: Model Win Rate

Indeed Title	Posting Title	Tf-idf Votes	Job Zone
manufacturing engineer	Manufacturing Engineer	0	4
multimedia specialist	Multimedia Creative Specialist	0	4
epidemiologist	Military Health Epidemiologist, Part-Time	0	5
diesel technician	Marine Diesel Technician	1	3
distribution manager	Distribution Manager - Overnights	1	4
environmental compliance specialist	Environmental/Regulatory Compliance Specialist	1	4
broadcast engineer	Freelance Maintenance Engineer, KABC-TV	1	4

Table 4.6: Indeed Titles with Poor Tf-idf Performance

responsibilities	skills	title	industry	employer	other
0.687	0.564	0.529	0.3	0.151	0.014

Table 4.7: % MTurkers Chose as Reason for Similarity

than the current job. The results also reveal some problems in the system. Firstly, some skills that look essential to a job are missing from its list of skills when they do not appear in job postings for this title. But they show up in job postings for a very similar job title. The system mistakenly thinks these are good skills for workers to acquire when the workers already possess them. An example is "pallet jacks" for lumber associate 4.8. Secondly, many job titles are common in different industries. These job's skill set may include the ones that are industry-specific. Recommending these skills to workers in a different industry would not be useful. "Patient care" 4.9 was recommended to a human resources specialist in order to transition to positions like "director of human resources", probably because we have job postings for these positions at hospitals. Thirdly, as we rely on Indeed for salary data, any inaccuracy could lead to bad recommendations. Skills helpful for a lower-paying job are not ideal candidates for re-skilling.

Table 4.11 shows the top 10 recommendations based on the job-skill interaction matrix model (Section 3.4.2). We chose a few from both highest value and lowest value skills (Table 3.4) as the starting skill. Most recommendations seem quite reasonable. But further investigation is required to validate these preliminary results. An interesting case is for "front desk". A large number of recommended skills are in the medical field. This could be because front desk jobs at doctor offices usually pay better than other places, and/or our system scraped a large number of job postings for this position. Two surprising recommendations are "playing cards" for "forklift" and "car wash" for "management experience". After some investigation, we realized many jobs in a casino require using forklift and many car washes were hiring managers. This reveals a shortcoming of the job-skill interaction matrix model. This model measures similarity by co-occurrence, i.e., skills often show up together are considered similar to each other. However, co-occurring skills don't necessarily perform similar tasks, nor does knowing one make learning the other easier.

Skill	Potential Future Jobs	Value
pallet jacks	forklift operator, stacker, replenishment associate, stock checker	983.2
inside sales	yard supervisor, yard specialist	607.1
delivery trucks	yard worker, yard specialist, stock checker	468.2
performance management	yard supervisor	412.4
coaching	yard supervisor	412.4
scaffolding	yard coordinator	388.1
truck driver	yard driver	350.3
reach trucks	replenishment associate	330.1
order pickers	replenishment associate	330.1
plumbing	yard specialist	194.7

Table 4.8: Recommendations for Lumber Associate

Skill	Potential Future Jobs	Value
employment law	human resources manager, director of human resources, senior human resources associate, senior director of human resources, senior human resources manager, head of human resources, vice president of human resources, hr consultant, labor relations specialist, human resources business partner	19790.6
continuous improvement	human resources manager, director of human resources, senior human resources associate, senior director of human resources, senior human resources manager, head of human resources, vice president of human resources, labor relations specialist, human resources business partner	18877.1
senior leadership	director of human resources, senior human resources associate, senior director of human resources, senior human resources manager, head of human resources, vice president of human resources, labor relations specialist, human resources business partner	18093.6
regulatory compliance	human resources manager, director of human resources, senior human resources associate, senior director of human resources, senior human resources manager, head of human resources, vice president of human resources, human resources business partner	17342.6
change management	director of human resources, senior human resources associate, senior director of human resources, head of human resources, vice president of human resources, hr consultant, human resources business partner	15164.9
data analysis	director of human resources, senior director of human resources, senior human resources manager, head of human resources, hr consultant, labor relations specialist, human resources business partner	14241.2
professional in human resources	director of human resources, senior director of human resources, senior human resources manager, labor relations specialist, human resources business partner	10276.2
patient care	director of human resources, vice president of human resources, labor relations specialist, human resources business partner	7596.2
senior professional in human resources	director of human resources, senior director of human resources, labor relations specialist	7058.3
business management	director of human resources, vice president of human resources, labor relations specialist	6686.0

Table 4.9: Recommendations for Human Resources Specialist

Skill	Potential Future Jobs	Value
process engineering	manufacturing engineer, senior manufacturing engineer, senior process engineer, process engineer, sustainability engineer, senior manufacturing supervisor, development operations engineer, materials engineer	5072.4
chemistry	senior process engineer, process engineer, sustainability engineer, development operations engineer, materials engineer	3713.7
coaching	senior industrial engineer, senior manufacturing engineer, senior process engineer, sustainability engineer, senior manufacturing supervisor, development operations engineer, lean six sigma specialist	3644.4
quality management	manufacturing engineer, senior manufacturing engineer, senior process engineer, process engineer, lean six sigma specialist	3346.8
statistical process control	manufacturing engineer, senior manufacturing engineer, senior process engineer, materials engineer, lean six sigma specialist	2993.4
technical writing	senior industrial engineer, manufacturing engineer, process engineer, development operations engineer, lean six sigma specialist	2893.5
program management	manufacturing engineer, sustainability engineer, senior manufacturing supervisor, materials engineer, lean six sigma specialist	2225.4
cad software	senior industrial engineer, manufacturing engineer, senior manufacturing engineer, development operations engineer	1914.5
microsoft excel	senior industrial engineer, senior manufacturing engineer, sustainability engineer, lean six sigma specialist	1792.5
process mapping	senior industrial engineer, senior process engineer, lean six sigma specialist	1629.5

Table 4.10: Recommendations for Industrial Engineer

Existing Skill	Recommended Skills
forklift	playing cards, automated export system, cutting machinery, gouges, compact tractors, gas grills, maximo software, load boxes, flatbed truck, barcode system
housekeeping	level probes, peoplenet, hour meters, buffing machine, mail bags, safety showers, sewing machines, mud pumps, sample containers, chemical tankers
front desk	registered dental assistant, dental assistant, x-ray certification, massage, medical records software, web design software, tegra, open dental, mindbody, vendor management system
physics	optics, matlab, professional engineer, calculus, hydraulics, signal processing, engineering and technology, image processing, fundamentals of engineering, failure analysis
machine learning	data science, ai, big data, artificial intelligence, data mining, hadoop, spark, computer vision, internet of things, natural language processing
management experience	profit and loss, market segmentation, client system, sap ariba, sciencedirect, sales software, adapt it, whiteboards, certified supply chain professional, car wash

Table 4.11: Recommendations by Job-Skill Interaction Matrix Model

Chapter 5

Discussion

5.1 Job Posting Embedding Models

Out of the 4 text embedding models we experimented with, tf-idf is the clear winner, albeit being the simplest. Recent advancement in NLP, although successful at improving state-of-art for standardized tasks, is not helpful in our scenario. This finding is not exciting news, yet admittedly not the most surprising. Work by Arora et al. [19] showed simple sentence embedding model (a varied version of tf-idf) is a formidable baseline when labeled training data is scarce or nonexistent (which unfortunately is our situation). One major advantage of newer models like BERT and USE over tf-idf is that they are aware of word order, making them particular powerful at tasks like sentiment analysis and entailment, where meaning can drastically change when word order changes. However, language used in job postings tends to be simple and unambiguous. When a job posting mentions a skill, it is much more likely this skill is required than not. On the other hand, a much smaller embedding size for BERT, USE and Doc2Vec means less information from the job postings is encoded compared to tf-idf. Another reason for the underperformance of BERT and USE can be the presence of out-of-vocabulary words in job postings. Job postings can be quite different from the corpus BERT and USE were pretrained on, while the tf-idf model was directly trained on the job posting data. Doc2Vec was also trained on the job posting data, but had the worst performance. Our explanation is Doc2Vec

cannot encode as much as information as tf-idf as its embedding is much shorter. At the same time, it suffers from having a much simpler neural network architecture compared to BERT and USE. Between USE and BERT, USE did consistently better in our experiment, though BERT has better performance in standardized NLP tasks. The reason can be two fold: 1) USE was designed to be more flexible for downstream tasks, as it was trained simultaneously on multiple tasks, textual similarity being one of them; 2) BERT was a word embedding model, and simply taking the average of all words in a job posting may not be the most optimal.

The ranking among the models is strongly backed by MTurk results. We then looked for more fine-grained patterns: does tf-idf have weakness, is there a type of posting USE does particularly well on, etc. The observation that all job postings that tf-idf did poorly on are from Job Zone 4 and 5 made us wonder if there is any correlation between tf-idf performance with Job Zone or length of posting (job from higher Job Zones tend to have longer job postings). But only very weak negative correlation was found. Anecdotally, we saw tf-idf choosing a post from the same company but for a very different position as the most similar, while other models were able to find a post from a different company but for a much closer position. This could happen when the keywords describing job responsibilities are common words, which get low tf-idf score, but the words describing the company are less common. Decision made by tf-idf in this situation is biased towards how similar the company is due to the way tf-idf score is calculated. Other models suffer less from this issue. However, we did not test this theory systematically.

5.2 Skill Recommendation

Our system has two ways of recommending skills. Model 1 recommends skills based on a worker's current occupation (Section 3.4.1). Model 2 recommends based on an existing skill (Section 3.4.2). Model 1 can be easily extended to multiple occupations by combining recommendations for each occupation and selecting the top ones by some metric. Model 2 can similarly be extended to more than 1 existing skill.

The two models nicely complement each other. A strength of Model 1 is its ability to recommend future jobs in addition to skills. Workers would also get a sense of estimated salary increase for such career transitions. On the other hand, Model 2 provides a way for workers to search for new skills based on a particular skill. This is especially useful if a worker has expertise in a skill and is specifically looking to pivot from that. Model 2 also assigns values on a skill-level. This can be helpful if workers have difficulty deciding which skill to acquire next.

One pitfall we observe in Model 2's recommendations is, even if we select the highest valued skills among the list of similar skills, skills that co-occur with low-value skills are often low-value skills too. Most skills and tools recommended to "housekeeping" involve manual work (Table 4.11). Workers re-skilling this way could be going from one low-paying job to another. Model 1, in comparison, would more likely recommend skills that improve workers' upward mobility. For example, "inside sales", "performance management", "coaching" are among the top recommendations for Lumber Associate (Table 4.8). Improvement in these soft skills could be a path to more responsibilities and higher income. Model 1 also addresses Model 2's other issue: recommending skills that co-occur, but perform very different functions. Distance between skills is measured by semantic similarity in Model 1. It effectively matches skills that involve similar tasks. Besides comparing text that mention skills in job postings, we also tried to compare the Wikipedia definition of skills. However, only about a third of our skills have Wikipedia entries. It would be difficult to integrate with our system.

Overall, we found the skills recommended by our system relevant and have great potential to enhance income. Our objectives were mostly met. We describe the limitations of our system in the next section, and propose improvements in the following section.

5.3 Limitations

As we use Indeed as our main source of data, the quality of our skill recommendations depends heavily on the quality of Indeed data. Our system assumes that Indeed classifies the job postings correctly under different job titles. In addition, our skill recommendations would only improve workers' future prospect if the salary figures on Indeed reflect the reality.

A majority of our skills set is technical in nature. This was a conscious choice as we believe it is often unclear how one would go about acquiring a non-technical skill. Online job postings over-represent positions intended for high-skilled and highly educated workers [3]. As such, skill recommendations for low-skilled workers, who probably need the most help, may be limited.

Perhaps, the usefulness of our system is most limited by the possibility that workers will not act on any of the recommendations even if they are good. If the biggest obstacle for re-skilling is not a lack of information, but a lack of motivation, a system like ours hardly helps.

5.4 Future Work

More data is almost always beneficial to machine learning systems. It's not difficult to set up a program that continuously scrapes for more job postings, which would update our embedding and valuation models in real time. Having a history of job postings would open up more research possibilities such as trends in skill requirements and salaries over time. If a continuous system were to be built, Tf-idf may not be the most suitable embedding model. The tf-idf encoder has to be retrained and job postings embeddings have to be recalculated every time new job postings come in. It would not be necessary for the USE encoder or the BERT encoder since they are pre-trained.

We can also collect data from more sources. Cross-referencing salary figures from multiple sources would improve the accuracy of our valuation models. Scraping job

postings from more websites could increase the range of occupations and skills the system covers. However, standardizing data cross platforms can be challenging.

Another direction of improvement is developing a more sophisticated skill tagger. Which skills get tagged to a job posting directly influences what skills our system believes an occupation requires, which in turn changes the skill embeddings. A more intelligent tagger should be able to identify more skill terms and lower false negatives. It should also disambiguate skills from their other meanings to reduce false positives. The end result would be an expanded skill list, which means more and potentially better choices for workers. Equally importantly is a skill normalizer that can recognize the same skill in different forms. Past work on skill identification and standardization [13] [7] [9] [12] could be very helpful to this project. Unfortunately, none of the work we found was open sourced. We decided against spending too much time on building a better skill tagger. Hopefully we have demonstrated that our recommendations make sense despite using a simple skill tagger. But we admit much improvement can still be made.

The usefulness of our system cannot be measured by cosine similarity or p-values. The ultimate test should be done by workers. That requires adding user interface to our system, and a user survey. When designing the interface, we could borrow ideas from past work that studied the effectiveness of employment tools. [20] identified design concepts that could address job seekers' social, personal, and societal needs, and give users a sense of empowerment. The DreamGigs project [6] reinforced the importance of raising self-efficacy in promoting action and achieving positive employment outcome. Both research target low-resource workers, a group our system may under-serve. To make our recommendations more actionable, we should include resources, such as online courses or training programs, that can help workers gain recommended skills.

Chapter 6

Conclusion

In this work, we introduce a system that recommends skills to workers looking to improve their employment prospect by re-skilling. In building the system, we compared different text embedding methods, and to our disappointment, saw new and hot neural network models getting beat by old-school tf-idf. The result demonstrates that state-of-the-art models for standardized machine learning problems do not necessarily work the best in a real world application.

Our system has two ways of recommending skills, one based on past jobs, the other based on existing skills. In addition to skills, the system also suggests career opportunities the recommended skills may open up. The system also gives users a sense of how valuable different skills are. We qualitatively evaluated the recommendations made by our system. It would be interesting to test the usefulness on potential users. We leave building the user interface as future work.

Bibliography

[1] Sun L AlShebli B Hidalgo C Rahwan I Alabdulkareem A, Frank MR. Unpacking the polarization of workplace skills. *Sci Adv 4:eaao6030*, 2018.

[2] David H. Autor. Why are there still so many jobs? the history and future of workplace automation. *Journal of Economic Perspectives*, pages 3–30, 2015.

[3] Tamara Jayasundera Carnevale, Anthony and Dmitri Repnikov. Understanding online job ads data. Technical report, Georgetown University, 2014.

[4] Sheng-yi Kong Nan Hua Nicole Limtiaco Rhomni St. John Noah Constant Mario Guajardo-Cespedes Steve Yuan Chris Tar Yun-Hsuan Sung Brian Strope Ray Kurzweil Daniel Cer, Yinfei Yang. Universal sentence encoder. *arXiv:1803.11175*, 2018.

[5] Vachik Dave, Baichuan Zhang, Mohammad Hasan, Khalifeh Aljadda, and Mohammed Korayem. A combined representation learning approach for better job and skill recommendation. *CIKM*, 2018.

[6] Tawanna R. Dillahunt and Alex Lu. Dreamgigs: Designing a tool to empower low-resource job seekers. CHI Conference on Human Factors in Computing Systems, 2019.

[7] Thomas Mahoney Matt McNair Faizan Javed, Phuong Hoang. Large-scale occupational skills normalization for online recruitment. the Twenty-Ninth AAAI Conference on Innovative Applications, 2017.

[8] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

[9] Ilkka Kivimäki, Alexander Panchenko, Adrien Dessy, Dries Verdegem, Pascal Francq, Hugues Bersini, and Marco Saerens. A graph-based approach to skill extraction from text. *TextGraphs@EMNLP*, 2013.

[10] H. Tong J. Yang Q. He L. Li, H. Jing and B.C. Chen. Nemo: Next career move prediction with contextual embedding. the 26th International Conference on World Wide Web Companion, page 505–513, 2017.

[11] E. Viennet M. Diaby and T. Launay. Toward the next generation of recruitment tools: an online social network-based job recommender system. the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, page 821–828, 2013.

[12] William Vaughan Sam Shah Peter Skomoroch Hyungjin Kim Sal Uryasev Mathieu Bastian, Matthew Hayes and Christopher Lloyd. Linkedin skills: Large-scale topic extraction and inference. *the 8th ACM Conference on Recommender Systems*, 2014.

[13] Ferosh Jacob and Matt McNair Meng Zhao, Faizan Javed. Skill: A system for skill identification and normalization. the Twenty-Ninth AAAI Conference on Artificial Intelligence, page 4012–4017, 2015.

[14] Chen Kai Corrado Greg Mikolov, Tomas and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013.

[15] James E. Bessen Erik Brynjolfsson Manuel Cebrian David J. Deming Maryann Feldman Matthew Groh José Lobo Esteban Moro Dashun Wang Hyejin Youn Iyad Rahwan Morgan R. Frank, David Autor. Toward understanding the impact of artificial intelligence on labor. *PNAS*, 2019.

[16] MIT's Task Force on the Work of the Future. The work of the future: Shaping technology and institutions. Technical report, MIT, 2019.

[17] Tomas Mikolov Quoc V. Le. Distributed representations of sentences and documents. *arXiv:1405.4053*, 2014.

[18] K. Bradley R. Rafter and B. Smyth. Automated collaborative filtering applications for online recruitment services. *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, page 363–368, 2000.

[19] Yingyu Liang Sanjeev Arora and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. *International Conference on Learning Representations*, 2017.

[20] Alex Lu Earnest Wheeler Tawanna R. Dillahunt, Jason Lam. Designing future employment applications for underserved job seekers: A speed dating study. Designing Interactive Systems Conference, 2018.

[21] Matt Gee Christina Sung Tristan Crockett, Eddie Lin. Skills-ml: An open source python library for developing and analyzing skills and competencies from unstructured text. Technical report, University of Chicago, 2018.

[22] Ratinov Lev Turian, Joseph and Yoshua Bengio. Word representations: A simple and general method for semisupervised learning. In *Association for Computational Linguistics*, the 48th Annual Meeting of the Association for Computational Linguistics, 2010.

[23] Mohammed Korayem Layla Pournajaf Khalifeh AlJadda Shannon Quinn Walid Shalaby, BahaaEddin AlAila and Wlodek Zadrozny. Help me find a job: A graph-based approach for job recommendation at scale. *arXiv:1801.00377*, 2018.

[24] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *arXiv:1910.03771*, 2019.

[25] Houssem Jerbi Xingsheng Guo and Michael P O'Mahony. An analysis framework for content-based job recommendation. *22nd International Conference on Case-Based Reasoning (ICCBR), Cork, Ireland*, 2014.