Humboldt Universität zu Berlin

Career Recommendations Using Supervised Latent Dirichlet Allocation

Master's Thesis submitted to

Prof. Dr. Sonja Greven

Prof. Dr. Brenda López Cabrera

by *Thomas Siskos (580726)*

In partial fulfillment of the requirement for the degree of

Master of Science in Statistics

March 30, 2020

Acknowledgments

I would like to thank

the people at Growify

for keeping me fed.

Abstract

For most people, looking for a new job is a chore at best. Picking a career is also a challenge. And even if one has a job she wishes to pursue, it is often not clear which skills should be acquired next.

To facilitate these problems we propose a model that is capable of predicting a person's occupation based on their provided set of skills. Using the estimated parameters of this model we can suggest the next skill a user may acquire in order to pursue a given career. Finally, the model allows for an automated grouping of skills into courses. We derive the steps of a Bayesian Variational Inference algorithm called Supervised Latent Dirichlet Allocation which extends original Latent Dirichlet Allocation algorithm in order to predict a multi-class label and implement it. We compare our implementation against multiple out-of-the-box algorithms for multi-class prediction using simulated data in various simulation settings.

We are able to show that Supervised Latent Dirichlet Allocation is capable of extracting the innate structure in most of the simulation settings. Finally, we demonstrate that the latent factors of Supervised Latent Dirichlet Allocation form logically coherent clusters of skills that are closely related to their respective profession and thus may lend themselves to the automatic collection of online training videos for online courses.

Keywords: Job-Recommendation, Skill-Recommendation, Supervised Latent Dirichlet Allocation, Variational Inference

Contents

1 Introduction **1**

1.1 Growify . . . 2

1.2 Problem Statement . . . 2

1.3 Proposed Solution . . . 3

2 Related Work **4**

2.1 Methods . . . 4

2.1.1 Extensions to Collaborative Filtering . . . 4

2.1.2 Latent Variables . . . 5

2.1.3 Machine Learning . . . 6

2.1.4 Graph Based Models . . . 6

2.2 Framing the Problem . . . 7

3 Supervised Latent Dirichlet Allocation **7**

3.1 Growify Data Model . . . 8

3.2 Notation and Terminology . . . 9

3.3 The Generative Process . . . 9

3.4 Posterior Inference . . . 12

3.5 Variational Inference . . . 12

3.6 Parameter Estimation . . . 16

3.6.1 Estimating ϕ . . . 17

3.6.2 Estimating γ . . . 19

3.6.3 Estimating α . . . 19

3.6.4 Estimating β . . . 21

3.6.5 Estimating η . . . 22

3.7 Job Prediction . . . 22

3.8 Skill Recommendation . . . 23

4 Data **25**

4.1 O*net 25
4.2 Problems 28

5 Simulations **29**

6 Results **32**

7 Conclusion **41**

8 Declaration of Authorship **47**

List of Algorithms

1 Maximize the Evidence Lower Bound ($ELBO$) 16
2 Generating synthetic data 30

List of Figures

1 Simplified Growify Data Model 8
2 Most Frequent Skills by O*net Category 26
3 Distribution of Importance Rating by Category 28
4 Results for Simulation A 34
5 Results for Simulation B 35
6 Results for Simulation C 36
7 Results for Simulation D 37
8 Model Weights η for each Class 38

List of Tables

1 Parameter Settings 30
2 Overall Accuracy across Algorithms 33
3 Skills associated with Latent Topics 39

1 Introduction

For most people looking for jobs is a quite jarring experience. In a conventional career a worker was expected to change her employer once maybe twice, during her entire lifetime, thus enjoying a relative level of job security. This is no longer the case. The traditional career model, of staying employed at one or two firms during ones lifetime, has become outdated. Changing the working environment has become increasingly more prevalent, due to firms downsizing their workforce in order to become more flexible with regards to global competition or in order to respond to technological advancements. Especially the rapidly changing technological requirements force today's workers to exchange their performance for continuous learning opportunities, instead of job security, such that they can maintain their marketability [27].

All of these developments leave today's employees in a precarious situation. Since they are forced to switch their occupation more often, they are spending more and more of their time looking for a new job. But even if someone is employed, often the further development of ones career is another problem to be solved. Thankfully, since the advent of the internet there exists an abundance of job recommendation portals, aiming to connect aspiring job-seekers and employers. The most advanced job recommendation portals identify the skill gap between a candidate and her desired new place of work. However, this prospective new job should not only offer adequate remuneration, ideally it also provides opportunities for personal growth and advancement, thus facilitating the next job transition as well.

From the firms' perspective, it is also desirable to maintain a workforce which is constantly learning and adapting to environmental changes. With a flexible labor pool like that, firms are able to maintain their profitability both in the short as well as in the long run. In order to achieve this, many firms offer online training programs to their employees. Nevertheless, those online courses more often than not cover a specific topic rather than to try and offer a selection of courses which enable an employee to undergo an entire professional track. This means that while there is enough supply of single courses, often there is little guidance as to which course to choose. Growify attempts to bridge that gap.

1.1 Growify

Growify is a start-up located in Berlin that works on a personnel development platform. The multi sided platform provides employees with learning content, that is structured in detailed learning paths. Workers can partake in a selection of learning paths, which contain learning material, e.g. relevant online job courses. Firms, on the other hand are able to customize the offered vocational tracks to their specific needs. However, currently these tracks are curated manually, which means that the process of creating such a track is arduous and requires extensive domain knowledge of each offered occupation. This constellation poses serious limitations to the scalability of Growify's business model.

1.2 Problem Statement

In particular, the problem of job and skill recommendations presents itself from three sides. First, users might want to change their current occupation out of their own volition, all the while being employed by the same firm, and are currently left with little or no guidance as to which job to pursue. That naturally leads to the first problem: which job is the most suited for a user, given her already acquired set of skills?

Secondly, users might be already steadfast in their choice of occupation. In this situation the more pressing question would be: how can a user improve her current set of skills as to be both more fit for her current job and well prepared for her next job transition?

Finally, is it possible to fully, or at least partly automate the generation of courses? While this question has nothing to do with the user's utility provided by the system it is essential from a business perspective. As the start-up grows, it will become increasingly difficult to continue generating the courses manually, since this requires intricate knowledge about the respective industries. Currently, this knowledge is provided by experts which have extensive experience in the desired fields. However, having to rely on in-house experts does not scale and will cause substantial problems in the long run. Mitigating this bottleneck will be of crucial importance for Growify's future.

1.3 Proposed Solution

This work proposes a Supervised Latent Dirichlet Allocation (*SLDA*) algorithm to tackle all three of the outlined problems. The main idea of *SLDA* is to extract latent variables as features which assist in predicting a target variable. *SLDA* is an algorithm that was originally developed in the context of natural language processing. As such it is a refinement of the original Latent Dirichlet Allocation algorithm and has found its use in different fields as well [4, 7].

In the following we will investigate the suitability of *SLDA* for recommending jobs and skills to users and for automatically generating contents for courses. If the *SLDA* algorithms yields satisfying results we can deploy it directly to answer the first problem, i.e. to provide direct job recommendations to users using the extracted latent topics.

The answer to the second problem, i.e. which skill to choose in order to improve a candidate's suitability for a given job, can be answered through the estimated probabilities of a skill conditional on the latent topic. I.e. if it is possible to identify the latent topic that has the highest marginal effect on the probability of a particular job, it is possible to recommend the skill that has the highest conditional probability and has not yet been already acquired by the user.

Finally, it may be possible to leverage the latent variables themselves in order to suggest courses in an automated fashion. These machine generated courses will still require some extend of human oversight. Nonetheless, it would constitute a substantial improvement with regards to the currently needed amount of manual maintenance.

The paper is organized as follows: section 2 summarizes the related work and the current state of the literature. Section 3 explains and derives the algorithm for Supervised Latent Dirichlet Allocation in a multi-class prediction problem. Section 4 demonstrates the available data and explains the need to rely on simulations, which are described in detail in section 5. Section 6 presents the results, which are discussed in section 7.

2 Related Work

The main way we will compare our algorithm is by evaluating its strength as a job recommendation engine. Thus it makes sense to examine the current state of the literature on this topic. Primarily we are interested in the way other recommendation systems frame their problem as well as what specific methods are employed.

The term recommendation system was coined in 1997 and used to include only Collaborative Filtering algorithms [23]. Since their inception recommendation systems have gathered a great deal of attention. They are ubiquitous and the most important tool when navigating the abundance of information created by the internet. In this work we focus our attention mostly on the literature regarding job recommendations.

2.1 Methods

The most common used algorithms are still extensions to the original Collaborative Filtering approach. Despite their age Collaborative Filtering methods are by far the most popular routines. The first recommendation system, Tapestry, was an experimental mail system, which enabled users to tag, filter and share documents among each other. The saved tags were then used to propose suggestions of similar documents for further reading. Thus, in Collaborative Filtering the basic idea is to find similarities across users of a platform and then to recommend items based on the preferences of the most comparable other users. In a sense, this is akin to opinion sharing with other people, hence why these algorithms are considered to be 'collaborative' [12, 23, 25].

2.1.1 Extensions to Collaborative Filtering

Collaborative Filtering still plays a big role in the field of career and occupation recommendation. Numerous authors have contributed to the refinement of the method by finding the most informative attributes for inferring the similarity between two users. Ochirbat and Shih deployed theoretic concepts from psychology such as Holland's interest theory and the Big Five Personality traits to model the similarity across users [20]. Similarly, Lekakos and Giaglis modeled the personality of users by extracting human factors in both

experimental settings [17] and Mamadou et al. tried the same using social network sites such as Facebook [10].

Other researchers have focused on the scalability of the Collaborative Filtering approach, which suffers greatly whenever the number of users or items grows too large. Deshpande's and Karypis' remedy for this problem is to cluster items or users beforehand and use only the relevant clusters for constructing the recommendations [9].

2.1.2 Latent Variables

The second major branch of the current literature for recommendation systems focuses on creating predictions using latent factor models. Often these methods involve a dimensionality reduction technique via a matrix factorization. The matrix of interest is the so called user-item matrix, which collects the ratings of each user on the rows and each item on the columns. This user-item matrix is then decomposed by a product of two fitting, and considerably smaller, matrices. These matrices are able to encode the data's column space into a latent subspace all the while maintaining the data's key characteristics [13, 14, 33].

Approaching the problem at hand via a latent variable model is tempting, but it also has a major flaw. Most of these algorithms try to predict a numerical rating. However, in this work's context, casting the prediction task this way is problematic. At best, we are only able to observe a users current job and no form of rating that is provided by them. We could assume that the user rates her current job as the best. But how, does a user rate all other jobs? This question has no satisfying answer since these ratings are unobservable. Nonetheless, latent variable models are still an option for not only predicting a rating, but rather a multi-class or binary label directly. A different approach to estimate latent dimensions is by using neural network architectures from the area of natural language processing in order to find a latent subspace that is capable of encoding the data while maintaining its core structure. In particular Nigam et al. were able to show that a Word2Vec architecture is able to provide a latent embedding space which can be exploited by a bidirectional Long-Short-Term-Memory Neural Network in order to predict if a user would interact positively with a suggested job advertisement [19].

2.1.3 Machine Learning

The problem of recommending jobs to users has also found considerable traction within the machine learning community. Multiple competitions are being held in which teams of researchers compete against each other in finding the most well performing predictive algorithms. The most prominent of these challenges, RecSys, is held by the job portal Xing. In this challenge researchers are provided a historical record of applicant data, job postings as well as an indication whether users chose to interact with a given job posting. Armed with this training set the teams are expected to deliver real-time, online predictions for a fraction of the site's traffic for a short period of time following the challenge's deadline. The teams are finally graded based on their performance on the real data [1].

Naturally, this challenge has drawn a lot of attention. Sato et al. as well as Yagci and Gurgen used Boosted Trees effectively in this context [24, 32]. Furthermore, Carpi et al. developped ensembles of different algorithms that exploit the cosine similarity of both users and items and incorporate this information into a mixture of Collaborative Filtering Approaches and Content Based Algorithms [6].

2.1.4 Graph Based Models

Another fruitful avenue for recommending jobs to users is the use of graphs. Graphs can be adopted to model the transition between jobs and to predict the future employer of a given user, given her occupation history [8, 22].

Kutty et al. used bipartite graphs to model the relationship between job-seeker and employers. A graph is called bipartite if it contains two distinct sets of nodes and all edges are connect only nodes from the opposite subsets. That is, bipartite graphs contain no edges between nodes that share the same subgroup. Bipartite graphs between employers and job-seekers have been proven to provide useful inputs to Support Vector Machines [16].

Similarly, graphs can be used to model the relationship between the skills, thus forming a so called skill ontology. A skill ontology is used to structure different skills according

to their properties. A general concept, for example, could be called 'Object Oriented Programming Languages' and could be comprised of 'Java', 'Python', or 'C++'. Exploiting such skill ontologies enabled Bradley et al. to provide handy features for K-Nearest Neighbour algorithms [5]. Still, these skill ontologies have to be curated manually again. For the task at hand this therefore would mean to put the cart before the horse.

2.2 Framing the Problem

In general, most of the literature regarding job recommendation algorithms can be divided into two parts. The first group tries to predict the ranking a particular user would give to the entire set of available job offers [6, 9, 13, 14, 16, 17, 26, 33].

The other major part of the literature tries to formulate the research question as a binary prediction problem. In this setting each user and each job opening are considered as a pair and the task is to predict whether the pair will result in a match. That is, to predict if a user will click or in other ways engage with the particular job offer or not [5, 10, 11, 15, 19, 24, 29]. One advantage of this formulation is that it reduces the dimensionality of the prediction problem. We do not follow this approach, since especially for job recommendations, we are interested in a ranking of different jobs. Thus our problem requires a formulation as a multi-label prediction task.

3 Supervised Latent Dirichlet Allocation

With a brief but succinct overview of the current state of the literature in place, we can continue with the three problems at hand. Again, these are job prediction, skill recommendation and an automated grouping of skills into courses. A promising candidate solution for overcoming all three obstacles might be Supervised Latent Dirichlet Allocation (*SLDA*) [3]. *SLDA* is an extension to Latent Dirichlet Allocation, a dimensionality reduction technique developed for the field of natural language processing. The goal of the original Latent Dirichlet Allocation algorithm is to find a grouping of words inside a document that share some semantic meaning [4]. If we were to extract some latent

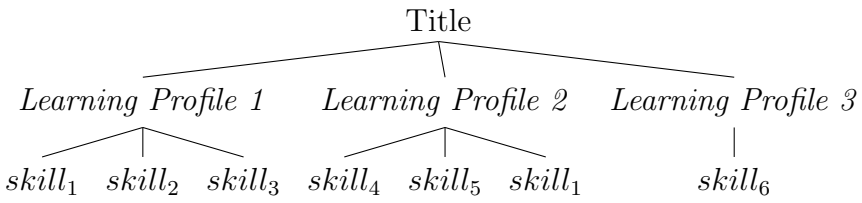


Figure 1: Simplified Growify Data Model

relationships between the skills of all relevant employees of a firm, it would be possible to use this lower dimensional subspace as input to the most common predictive algorithms. This is, in principle, very similar to running a multinomial regression on the first few principal components of a given data set's correlation matrix.

However, when the goal of the analysis lies in prediction, using the standard latent topics that are extracted by Latent Dirichlet Allocation might not be the most efficient route to pursue [3]. Instead of just finding semantically connected skills, we additionally require them to contain some predictive power with regards to a target variable. In contrast to its predecessor, *SLDA* does exactly that. The first proposition of Supervised Latent Dirichlet Allocation suggested an Expectation Maximization algorithm for estimating normally and poisson distributed target variables, i.e. for unconstrained, numerical variables and for count data [3]. This method has been later extended in order to accommodate multi-class labels [7]. In our case, we will use this extension to extract latent topic variables that are indicative of a user's job title. But why might this be a good idea?

3.1 Growify Data Model

The main reason why we deploy *SLDA* lies in the way Growify structures the relationship between skills and jobs. According to Growify's very own data model, a job title consists of a selection of, possibly overlapping, sets of skills. These skill sets are so called 'Learning Profiles'. Job titles are therefore characterized as mixtures of Learning Profiles. In turn, Learning Profiles are characterized as mixtures of skills. The relationship between jobs, Learning Profiles and skills is depicted in Figure 1. Note that a particular skill can be an element of two Learning Profiles, as is $skill_1$ in Figure 1.

This modelling choice for job titles is quite fortunate, as it fits naturally into the

presupposed structure of *SLDA*. Remember, in *SLDA* targets are assumed to be traceable to mixtures of latent topics and latent topics are essentially distributions over words in a document. For us, the target variable is a user's job. And what used to play the role of words in the natural language processing context will be replaced by a user's skills. Multiple skills can be collected into Learning Profiles. To solve our problem we assume that jobs can be expressed as mixtures of Learning Profiles. The Learning Profiles then, can be thought of as distributions over skills.

3.2 Notation and Terminology

Before we start with a derivation of *SLDA* it is of importance to define the terms we will use throughout the entirety of this paper. A skill is the basic unit of discrete data, which is an item from a vocabulary indexed by $\{1, \dots, S\}, S \in \mathbb{N}$. Skills are modelled as one-hot encoded unit vectors such that $\mathbf{s}_i = 1$ and $\mathbf{s}_j = 0 \; \forall i \neq j, i, j \in \{1, \dots, S\}$.

For our purposes a user consists of two parts. A user can have one of $J \in \mathbb{N}$ jobs, which constitute her class label $y \in \{1, \dots, J\}$. Additionally we can describe a user by her sequence of $N \in \mathbb{N}$ skills,

$$\mathbf{s}_{1:N} = (\mathbf{s}_1, \dots, \mathbf{s}_N),$$

where $\mathbf{s}_n$ is the nth skill in the sequence. We formally describe a user u as the tuple

$$u = (\mathbf{s}_{1:N}, y).$$

A pool of users $\mathcal{P}$ is a collection of $U \in \mathbb{N}$ users

$$\mathcal{P} = \{u_1, \dots, u_U\}.$$

3.3 The Generative Process

For fixed parameters $K \in \mathbb{N}, \alpha \in \mathbb{R}^K, \beta_{1:K} \in \mathbb{R}^{S \times K}$ and $\eta \in \mathbb{R}^{J \times K}$ we could model the data generating process. Here, K is the number of latent topics in the pool. Unfortunately,

K is a hyperparameter and as such has to be specified beforehand. As for the other parameters, α is the K-dimensional parameter to the Dirichlet Distribution. The columns of the $S \times K$ matrix $\beta_{1:K}$ contain the probabilities for each skill conditional on the latent topic. Naturally, this means that each column of $\beta_{1:K}$ has to sum up to one. Finally η is a $J \times K$ matrix that stores the parameters that determine the response variable. We model the generative process for one user in the following way:

1. Draw the proportions of topics $\theta|\alpha \sim Dir(\alpha)$.
2. For each skill in $n = 1, \ldots, N$:
   a) Assign a topic $z_n|\theta \sim Cat(\theta)$.
   b) Draw a skill $\mathbf{s}_n|z_n, \beta_{1:K} \sim Cat(\beta_{z_n})$.
3. Draw the response variable $y|z_{1:N}, \eta$,

where we make use of a sigmoid for modeling the probability of a given job

$$p(y_j|z_{1:N}, \eta) = \frac{\exp\left(\eta_j^T \bar{z}\right)}{\sum_{c=1}^{J} \exp\left(\eta_c^T \bar{z}\right)}. \tag{1}$$

Note that here we assume the skills to be conditionally independent of each other, given their latent topic assignment. Now we can model the joint probability for one user as

$$p(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta) = p\left(\theta|\alpha\right) \left(\prod_{n=1}^{N} p\left(z_n|\theta\right) p\left(\mathbf{s}_n|z_n, \beta_{1:K}\right) \right) p\left(y|z_{1:N}, \eta\right),$$

where

$$p\left(\theta|\alpha\right) = \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma\left(\alpha_k\right)} \prod_{k=1}^{K} \theta^{\alpha_k - 1}$$

is the probability density function of the Dirichlet distribution [7]. Next, the probability of a particular latent topic can be expressed as

$$p\left(z_n|\theta\right) = \prod_{k=1}^{K} \theta_k^{z_n^k},$$

which is exploiting that z_n is a one hot encoded vector. Thus, the above term is simply selecting the appropriate index position of θ.

Finally, the probability of a particular skill $\mathbf{s}_n$ appearing in the context of the kth latent topic,

$$p\left(\mathbf{s}_n|z_n, \beta_{1:K}\right) = \prod_{k=1}^{K} \prod_{s=1}^{S} \beta_{\mathbf{s}_n^s}^{z_n^k},$$

is simply selecting the appropriate entry of $\beta_{1:K}$, where the skill $\mathbf{s}_n$ selects the row of $\beta_{1:K}$ and the latent topic z_n determines the column. That is, both z_n and $\mathbf{s}_n$ are indicator vectors and as such z_n's only nonzero entry selects the position and all other factors are set to 1.

With a generative model in place we are still left with four questions. The first is, how to derive the posterior distribution of the latent variables θ and $z_{1:N}$. The next two problems call for ways to estimate parameters and for procedures to obtain predictions. Finally we wish to find rules for skill recommendations.

3.4 Posterior Inference

The key inferential problem lies in finding the posterior distribution of the latent variables. By applying the definition of conditional distributions we get

$$\begin{aligned}
&p(\theta, z_{1:N}|\mathbf{s}_{1:N}, y, \alpha, \beta_{1:K}, \eta) \\
&\quad= \frac{p\left(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta\right)}{p\left(\mathbf{s}_{1:N}, y, |\alpha, \beta_{1:K}, \eta\right)} \\
&\quad= \frac{p\left(\theta|\alpha\right)\left(\prod_{n=1}^{N} p\left(z_n|\theta\right) p\left(\mathbf{s}_n|z_n, \beta_{1:K}\right)\right) p\left(y|z_{1:N}, \eta\right)}{\int \sum_{z_{1:N}} p\left(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta\right) d\theta} \\
&\quad= \frac{p\left(\theta|\alpha\right)\left(\prod_{n=1}^{N} p\left(z_n|\theta\right) p\left(\mathbf{s}_n|z_n, \beta_{1:K}\right)\right) p\left(y|z_{1:N}, \eta\right)}{\int \sum_{z_{1:N}} p\left(\theta|\alpha\right)\left(\prod_{n=1}^{N} p\left(z_n|\theta\right) p\left(\mathbf{s}_n|z_n, \beta_{1:K}\right)\right) p\left(y|z_{1:N}, \eta\right) d\theta}
\end{aligned}$$

Note however, that the integral in θ, which is also called the evidence, is K-dimensional. Therefore it is not efficiently computable [7]. Standard Bayesian Methods would try to compute the integral through the use of Markov-Chain-Monte-Carlo Methods, like the Metropolis-Hastings algorithm or Gibbs-Sampling. These methods however, tend to be slow and to not scale well to large data. For this reason we will try to approximate the integral using variational inference, which has been shown to be faster and more capable of handling large data [2].

3.5 Variational Inference

The idea of Variational Inference is to find the member of a family of densities $\mathcal{D}$, which minimizes the Kullback-Leibler divergence to the posterior. That is

$$q^*(\theta, z_{1:N}|\gamma, \phi) = \underset{q(\theta, z_{1:N}|\gamma, \phi) \in \mathcal{D}}{\arg\min}\, KL\big(q(\theta, z_{1:N}|\gamma, \phi) \;||\; p\left(\theta, z_{1:N}|\mathbf{s}_{1:N}, y, \alpha, \beta_{1:K}, \eta\right)\big).$$

We define the variational distribution as

$$q(\theta, z_{1:N}|\gamma, \phi_{1:N}) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n).$$

This can be thought of as trying to find a good proxy γ for the latent topic proportions θ and finding N proxies ϕ_n for each skill $\mathbf{s}_n$.

Nevertheless, this objective still involves the unobtainable evidence and is thus intractable as well. To make this apparent, consider

$$\begin{aligned}
&KL\big(q(\theta, z_{1:N}|\gamma, \phi_{1:N}) \;||\; p\left(\theta, z_{1:N}|\mathbf{s}_{1:N}, y, \alpha, \beta_{1:K}, \eta\right)\big) \\
&\quad = E_q\left[\log q(\theta, z_{1:N}|\gamma, \phi)\right] - E_q\left[\log p\left(\theta, z_{1:N}|\mathbf{s}_{1:N}, y, \alpha, \beta_{1:K}, \eta\right)\right] \\
&\quad = E_q\left[\log q(\theta, z_{1:N}|\gamma, \phi_{1:N})\right] - E_q\left[\log\left(\frac{p\left(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta\right)}{p\left(\mathbf{s}_{1:N}, y, |\alpha, \beta_{1:K}, \eta\right)}\right)\right] \\
&\quad = E_q\left[\log q(\theta, z_{1:N}|\gamma, \phi_{1:N})\right] - E_q\left[\log p\left(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta\right)\right] + p\left(\mathbf{s}_{1:N}, y, |\alpha, \beta_{1:K}, \eta\right),
\end{aligned}$$

where the subscript of the expectation operator denotes that the expectation is taken with respect to q . Note that we can drop the expectation operator in the last term, since it does not depend on the latent variables θ and $z_{1:N}$. Therefore it is not random with respect to q either [2].

As we cannot compute the Kullback-Leibler divergence, we optimize an alternative objective which we obtain by dropping the last term. We define

$$\mathcal{L}\left(q\right) = E_q\left[\log p\left(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta\right)\right] - E_q\left[\log q(\theta, z_{1:N}|\gamma, \phi)\right], \tag{2}$$

which is the negative Kullback-Leibler divergence from before, minus the evidence. This means in particular, that maximizing this objective is equivalent to minimizing the Kullback-Leibler divergence between the variational distribution and the posterior. This objective is also called the Evidence Lower Bound ($ELBO$) [4]. To see where this name stems from consider the log likelihood of the evidence

$$\begin{aligned}
\log p\left(\mathbf{s}_{1:N}, y, |\alpha, \beta_{1:K}, \eta\right) &= \log \int \sum_{z_{1:N}} p\left(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta\right) d\theta \\
&= \log \int \sum_{z_{1:N}} \frac{p\left(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta\right) q(\theta, z_{1:N}|\gamma, \phi)}{q(\theta, z_{1:N}|\gamma, \phi)} d\theta \\
&= \log E_q\left[\frac{p\left(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta\right)}{q(\theta, z_{1:N}|\gamma, \phi)}\right].
\end{aligned}$$

Now we can lower bound this term using Jensen's Inequality

$$\log p\left(\mathbf{s}_{1:N}, y, |\alpha, \beta_{1:K}, \eta\right) \geq \underbrace{E_q\left[\log\left(\frac{p\left(\theta, z_{1:N}, \mathbf{s}_{1:N}, y|\alpha, \beta_{1:K}, \eta\right)}{q(\theta, z_{1:N}|\gamma, \phi)}\right)\right]}_{\mathcal{L}(q)}$$

[7]. Note however, that this formulation turned out to be numerically unstable. Especially for large sigmoid weights η the objective tends to blow up. In order to alleviate that problem, we introduce additionally one last term $\lambda||\eta||^2$, which is a L2-penalty term whose weight in the objective function is governed by λ [31]. The point of this term is to discourage further optimization procedures down the line from choosing large values for η. This effectively solves all numerical issues while, admittedly, weakening the argument about the KL divergence between the variational density and the posterior distribution. We can no longer claim directly that maximizing the *ELBO* is equivalent to minimizing the KL-divergence. Instead, all we can say is that maximizing this constrained *ELBO* is minimizing the KL-divergence subject to the constraint of having small weights.

The *ELBO* of a single user is

$$\begin{aligned}
\mathcal{L}(q) &= E_q\left[\log p\left(\theta|\alpha\right)\right] + \sum_{n=1}^{N} E_q\left[\log p\left(z_n|\theta\right)\right] + \sum_{n=1}^{N} E_q\left[\log p\left(\mathbf{s}_n|z_n, \beta_{1:K}\right)\right] \\
&\quad + E_q\left[\log p\left(y|z_{1:N}, \eta\right)\right] - E_q\left[\log q\left(\theta|\gamma\right)\right] - \sum_{n=1}^{N} E_q\left[\log q\left(z_n|\phi\right)\right] - \lambda||\eta||^2 \\
&= \log\Gamma\left(\sum_{k=1}^{K}\alpha^k\right) + \sum_{k=1}^{K}\log\Gamma\left(\alpha^k\right) - \sum_{k=1}^{K}(\alpha^k - 1)\left[\Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right] \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K}\phi_n^k\left[\Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right] \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{s=1}^{S}\phi_n^k\mathbf{s}_n^s\log\beta_k^s \\
&\quad + \sum_{c=1}^{J} y^c\left[\eta_c^T\bar{\phi} - \log\left\{\sum_{c=1}^{J}\prod_{n=1}^{N}\left(\sum_{k=1}^{K}\phi_n^k\exp\left(\frac{\eta_c}{N}\right)\right)\right\}\right] \\
&\quad - \log\Gamma\left(\sum_{k=1}^{K}\gamma^k\right) + \sum_{k=1}^{K}\log\Gamma\left(\gamma^k\right) - \sum_{k=1}^{K}(\gamma^k - 1)\left[\Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right] \\
&\quad - \sum_{n=1}^{N}\sum_{k=1}^{K}\phi_n^k\log(\phi_n^k) - \frac{1}{2}\lambda||\eta||^2
\end{aligned}$$

Where $E_q\left[\log p(\theta^k|\alpha)\right] = \Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)$ which can be obtained since the Dirich-

let distribution is a member of the exponential family. As such it is possible to derive the expectation of the sufficient statistic θ^k by computing the first derivative of the log normalization factor [4].

Here Ψ denotes the digamma function, which is the first derivative of the log-Gamma function. Also note, that $\sum_{c=1}^{J}\prod_{n=1}^{N}\left(\sum_{k=1}^{K}\phi_n^k\cdot\exp\left(\frac{\eta_c^k}{N}\right)\right)$ is a linear function of ϕ_n and can be written as the scalar product $h\left(\eta\right)^T\phi_n$, where $h\left(\eta\right)=[h_1,\ldots,h_K]$, is only a function of all other $\phi_i, i\neq n$ [7]. To demonstrate that, consider the last variational parameter in a user's skill sequence ϕ_N:

$$\begin{aligned}\sum_{c=1}^{J}\prod_{n=1}^{N}\left(\sum_{k=1}^{K}\phi_n^k\cdot\exp\left(\frac{\eta_c^k}{N}\right)\right) &= \underbrace{\exp\left(\frac{\eta_1}{N}\right)^T\phi_1\cdot\exp\left(\frac{\eta_1}{N}\right)^T\phi_2\cdot\cdots\cdot\exp\left(\frac{\eta_1}{N}\right)^T}_{:=d_1^{-N}}\phi_N\\ &\quad+\underbrace{\exp\left(\frac{\eta_2}{N}\right)^T\phi_1\cdot\exp\left(\frac{\eta_2}{N}\right)^T\phi_2\cdot\cdots\cdot\exp\left(\frac{\eta_2}{N}\right)^T}_{:=d_2^{-N}}\phi_N\\ &\quad\cdots\\ &\quad+\underbrace{\exp\left(\frac{\eta_J}{N}\right)^T\phi_1\cdot\exp\left(\frac{\eta_J}{N}\right)^T\phi_2\cdot\cdots\cdot\exp\left(\frac{\eta_J}{N}\right)^T}_{:=d_J^{-N}}\phi_N\end{aligned}$$

For each of the J summands, we can single out a $\phi_n, n\in\{1,\ldots,N\}$ and formulate the rest of the product as $d_c^{-n}, c\in 1,\ldots,J, n\in 1,\ldots,N$. With that in place it is easy to see

$$\begin{aligned}\sum_{c=1}^{J}\prod_{n=1}^{N}\left(\sum_{k=1}^{K}\phi_n^k\cdot\exp\left(\frac{\eta_c^k}{N}\right)\right) &= (1,\ldots,1)\cdot\begin{bmatrix}d_1^{-N}\\ d_2^{-N}\\ \vdots\\ d_J^{-N}\end{bmatrix}\cdot\phi_N\\ &= \left[D_{-N}\mathbb{1}_J\right]^T\phi_N\\ &= h\left(\eta\right)^T\phi_N,\end{aligned}$$

where $\mathbb{1}_J$ is a J-dimensional vector of ones. In general we can formulate

$$h\left(\eta\right)^T \phi_n := \sum_{c=1}^{J} \prod_{n=1}^{N} \left(\sum_{k=1}^{K} \phi_n^k \cdot \exp\left(\frac{\eta_c^k}{N}\right) \right) = \left[D_{-n}\mathbb{1}_J\right]^T \phi_n.$$

With all that in place, we are ready to start deriving parameter updates in order to maximize the *ELBO*.

3.6 Parameter Estimation

To maximize the *ELBO* we wish to formulate a Coordinate Ascent Algorithm, where we maximize one set of parameters while holding all others fixed. We start by maximizing the *ELBO* for each user's vector of skills, that is we maximize each users ϕ and γ vectors first and then turn our attention to the model's parameters $\alpha, \beta_{1:K}$ and η [7]. The procedure is summarized in pseudo-code in algorithm 1.

It is also possible to interpret this procedure as an Expectation Maximization Algorithm [4]. The first part, maximizing with respect to the user specific variational parameters is akin to estimating the expected values of the true topic memberships $z_{1:N}$ by $\phi_{1:N}$. The second step then tries to maximize the pool level *ELBO* conditional on the

Algorithm 1 Maximize the Evidence Lower Bound (*ELBO*)

INPUT: a dataset
OUTPUT: user parameters γ_u, ϕ_u, model parameters α, β, η
INITIALIZE: $\alpha_k \leftarrow 1/K, \beta_s^k \leftarrow 1/S, \eta \leftarrow \mathbb{1}_C \mathbb{1}_K^T \gamma_u^k \leftarrow 1/K, \phi_{un}^k \leftarrow 1/K$
while `pool_elbo` has not converged **do**:
  for each `user` in `pool` **do**
    maximize `user_elbo` w.r.t its' variational parameters
      $\Rightarrow$ maximize w.r.t and
        ϕ using equation (4) and
        γ using equation (5)
  end for
  maximize `pool_elbo` w.r.t. the model parameters
    $\Rightarrow$ maximize w.r.t
      α using equation (6)
      $\beta_{1:K}$ using equation (7)
      η using equations (8) and (9)
end while
return α, β, η

expectations of the previous step.

In the following sections we first derive the update equations for the user level variational parameters ϕ and γ. After that we derive update equations for the pool level parameters $\alpha, \beta_{1:K}$ and η. For each set of parameter we first collect all the relevant terms of the *ELBO*, state the first order condition and derive the update equations wherever possible.

3.6.1 Estimating ϕ

The terms of the user-level *ELBO*, which depend on ϕ_n^k are:

$$\begin{aligned}\mathcal{L}_{|\phi_n^k|} = &\sum_{n=1}^{N}\sum_{k=1}^{K}\phi_n^k\left[\Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right] \\ &+ \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{s=1}^{S}\phi_n^k \mathbf{s}_n^s \log\beta_k^s - \sum_{n=1}^{N}\sum_{k=1}^{K}\phi_n^k\log(\phi_n^k) \\ &+ \sum_{c=1}^{J} y^c\left\{\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\eta_c^k\phi_n^k - \log\left(h\left(\eta\right)^T\phi_n\right)\right\}.\end{aligned}$$

Additionally, we require that the K components of ϕ sum to one. By augmenting the previous equation with appropriate Lagrange multipliers this leaves us with the objective

$$\begin{aligned}\mathcal{L}_{|\phi_n^k|} = &\sum_{n=1}^{N}\sum_{k=1}^{K}\phi_n^k\underbrace{\left\{\left[\Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right] + \sum_{s=1}^{S}\mathbf{s}_n^s\log\beta_k^s\right\}}_{=:b^k} \\ &+ \sum_{j=1}^{J} y^j\left\{\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\phi_n^k - \log h\left(\eta\right)^T\phi_n\right\} \\ &- \sum_{n=1}^{N}\sum_{k=1}^{K}\phi_n^k\log\phi_n^k - \xi\left(\sum_{k=1}^{K}\phi_n^k - 1\right).\end{aligned}$$

However, this objective is still troublesome. Especially, the coupling between $h\left(\eta\right)$ and ϕ_n in the log of the first line is burdensome. We therefore will use the quantity

$$\log(x) \leq \zeta^{-1}x + \log\zeta - 1 \tag{3}$$

to lower bound this term, where we set $x = h\left(\eta\right)^k\phi_n^k$ and $\zeta = h\left(\eta\right)^T\phi_n^{old}$ [7]. With this

maneuver we approximate $\log h(\eta)^T \phi_n$ by a first-order Taylor approximation around the previous' iterations value for ϕ. Therefore, it still constitutes a valid parameter update and due to the negative sign in front of the term we ensure that the objective we wish to maximize is made artificially lower. Also note that equation (3) holds with equality if and only if $x = \zeta$, which should hold at least when the algorithm converges. Now we get that

$$\mathcal{L}_{|\phi_n^k|} \geq \sum_{k=1}^{K} \phi_n^k b^k + \sum_{j=1}^{J} y^j \left\{ \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \eta_j^k \phi_n^k - \left[\left(h(\eta)^T \phi_n^{old} \right)^{-1} h(\eta)^T \phi_n + \log \left(h(\eta)^T \phi_n^{old} \right) - 1 \right] \right\} - \sum_{n=1}^{N} \sum_{k=1}^{K} \phi_n^k \log(\phi_n^k) - \xi \left(\sum_{k=1}^{K} \phi_n^k - 1 \right).$$

And its first derivative

$$\frac{\partial \mathcal{L}_{|\phi_n^k|}}{\partial \phi_n^k} = b^k + \frac{\eta_j^k}{N} - \left(h(\eta) \phi_n^{old} \right)^{-1} h(\eta)^k - \log \phi_n^k - 1 - \xi \stackrel{!}{=} 0,$$

which yields the update

$$\phi_n \propto \exp \left\{ \Psi(\gamma) \log \beta_k^s - \left(h(\eta)^T \phi_n^{old} \right)^{-1} h(\eta) \right\}, \tag{4}$$

where the proportionality sign means that we normalize the updated ϕ vector to sum to one manually, since we dropped some terms for convenience [7].

3.6.2 Estimating γ

The terms that depend on γ are

$$\begin{aligned}
\mathcal{L}_{|\gamma^k|} &= \sum_{k=1}^{K}(\alpha_k - 1)\left[\Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right] \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K}\phi_n^k\left[\Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right] \\
&\quad - \log\Gamma\left(\sum_{k=1}^{K}\gamma^k\right) + \sum_{k=1}^{K}\log\Gamma\left(\gamma^k\right) - \sum_{k=1}^{K}(\gamma^k - 1)\left[\Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right] \\
&= \sum_{k=1}^{K}\left[\Psi\left(\gamma^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right]\left(\alpha^k - \gamma^k + \sum_{n=1}^{N}\phi_n^k\right) - \log\Gamma\left(\sum_{k=1}^{K}\gamma^k\right) + \sum_{k=1}^{K}\log\Gamma(\gamma^k) \\
&= \sum_{k=1}^{K}\left\{\Psi(\gamma)\left[\alpha^k - \gamma^k + \sum_{n=1}^{N}\phi_n^k\right] - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\left[\alpha^k - \gamma^k + \sum_{n=1}^{N}\phi_n^k\right]\right\} \\
&\quad - \log\Gamma\left(\sum_{k=1}^{K}\right) + \sum_{k=1}^{K}\log\Gamma(\gamma^k).
\end{aligned}$$

With first order condition

$$\begin{aligned}
\frac{\partial\mathcal{L}_{|\gamma|}}{\partial\gamma^k} &= \Psi^T(\gamma^k)\left[\alpha^k - \gamma^k + \sum_{n=1}^{N}\phi_n^k\right] - \Psi(\gamma^k) \\
&\quad - \left\{\Psi^T\left(\sum_{k=1}^{K}\right)\left[\alpha^k - \gamma^k + \sum_{n=1}^{N}\phi_n^k\right] - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right\} - \Psi\left(\sum_{k=1}^{K}\gamma^k\right) + \Psi(\gamma^k) \\
&= \left[\Psi^T(\gamma^k) - \Psi\left(\sum_{k=1}^{K}\gamma^k\right)\right]\cdot\underbrace{\left(\alpha^k - \gamma^k + \sum_{n=1}^{N}\phi_n^k\right)}_{\text{it suffices if this part is zero}} \overset{!}{=} 0,
\end{aligned}$$

which yields the update for γ

$$\gamma = \alpha + \sum_{n=1}^{N}\phi_n, \tag{5}$$

which is exactly the same as in the original Latent Dirichlet Allocation algorithm [4].

3.6.3 Estimating α

Next, we want to find an iterative update for α. As this is the first model parameter, we focus our attention to the pool-level-*ELBO*, which is the sum of all individual *ELBO*-

contributions The terms that depend on α are

$$\begin{aligned}\mathcal{L}_{|\alpha|} &= \sum_{p=1}^{P}\left\{\log\Gamma\left(\alpha^k\right) - \sum_{k=1}^{K}\log\Gamma(\alpha^k) + \sum_{k=1}^{K}(\alpha^k-1)\left[\Psi\left(\gamma_p^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma_p^k\right)\right]\right\} \\ &= P\left\{\log\Gamma\left(\alpha^k\right) - \sum_{k=1}^{K}\log\Gamma(\alpha^k)\right\} + \sum_{p=1}^{P}\sum_{k=1}^{K}(\alpha^k-1)\left[\Psi\left(\gamma_p^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma_p^k\right)\right].\end{aligned}$$

Due to the structure of the Dirichlet distribution's Hessian we can find an efficient Newton-Raphson update for α. We just need to derive the gradient and the Hessian. The first derivative is

$$\frac{\partial\mathcal{L}_{|\alpha|}}{\partial\alpha^k} = \underbrace{P\left[\Psi\left(\sum_{k=1}^{K}\alpha^k\right) - \Psi\left(\alpha^k\right)\right] + \sum_{p=1}^{P}\left[\Psi\left(\gamma_p^k\right) - \Psi\left(\sum_{k=1}^{K}\gamma_p^k\right)\right]}_{:=g}.$$

For the Hessian we look at the two second derivatives, first on the main diagonal:

$$\frac{\partial\mathcal{L}_{|\alpha|}}{\partial\alpha^k\alpha^k} = P\left[\Psi'\left(\sum_{k=1}^{K}\alpha^k\right) - \Psi'(\alpha^k)\right] = P\cdot\Psi'\left(\sum_{k=1}^{K}\alpha^k\right) - P\cdot\Psi'(\alpha^k).$$

And on the off-diagonals we get

$$\frac{\partial\mathcal{L}\left(q\right)}{\partial\alpha^k\alpha^l} = P\cdot\Psi'\left(\sum_{\alpha^k=1}^{\alpha^K}\right).$$

Note that the two equations only differ in one term. We can therefore write the Hessian as

$$H = \left[Q + \mathbb{1}\mathbb{1}^T z\right],$$

where $Q = \operatorname{diag}\left(-\Psi'(\alpha)\right)$ is a diagonal matrix of dimension K, $\mathbb{1}$ is an appropriate vector of ones and $z = \Psi'\left(\sum_{k=1}^{K}\alpha^k\right)$. We can now define the Newton-Raphson step as

$$\alpha^{new} = \alpha^{old} - H^{-1}g, \tag{6}$$

where

$$H^{-1} = Q^{-1} - \frac{Q^{-1}\mathbb{1}\mathbb{1}^T Q^{-1}}{\frac{1}{z} + \mathbb{1}^T Q^{-1}\mathbb{1}},$$

which does not require inverting the Hessian and instead relies on the straightforward inversion of a diagonal matrix[18]. Note however, that tuning α frequently leads to numerically unstable behavior, which is the reason why it is usually not optimized, but rather left at its initial value [7].

3.6.4 Estimating β

Next, we derive the update equations for $\beta_{1:K}$. The terms that depend on $\beta_{1:K}$ are

$$\mathcal{L}_{|\beta|} = \sum_{p=1}^{P}\sum_{n=1}^{N^p}\sum_{k=1}^{K}\sum_{t=1}^{S} \phi_{pn}^k \mathbf{s}_{pn}^t \log \beta_k^t.$$

Additionally we require that each column in $\beta_{1:K}$ sums to one. This leaves us with the Lagrangian

$$\mathcal{L}_{|\beta|} = \sum_{p=1}^{P}\sum_{n=1}^{N_p}\sum_{k=1}^{K}\sum_{t=1}^{S} \phi_{pn}^k \mathbf{s}_{pn}^t \log \beta_k^t - \sum_{k=1}^{K} \nu_k \left(\sum_{t=1}^{S} \beta_k^t - 1 \right).$$

The first order conditions are:

$$\frac{\partial \mathcal{L}_{|\beta|}}{\partial \beta_k^t} = \sum_{p=1}^{P}\sum_{n=1}^{N_p} \phi_{pn}^k \mathbf{s}_{pn}^t \frac{1}{\beta_k^t} - \nu_k \stackrel{!}{=} 0.$$

Which yields the parameter update for $\beta_{1:K}$:

$$\beta_k^t \propto \sum_{p=1}^{P}\sum_{n=1}^{N_p} \phi_{pn}^k \mathbf{s}_{pn}^t. \tag{7}$$

Keep in mind that $\mathbf{s}_{pn}$ is an indicator vector. This way it is easy to see that the probability of a skill conditional on a topic is proportional to the estimated membership across the entire pool [4, 7].

3.6.5 Estimating η

Finally, the terms that depend on η are:

$$\mathcal{L}_{|\eta|} = \sum_{p=1}^{P} \left\{ \eta_{jp} \bar{\phi}_p - \log \overbrace{\left(\sum_{c=1}^{J} \sum_{n=1}^{N_p} \underbrace{\left(\sum_{k=1}^{K} \phi_{pn}^k \exp\left(\frac{\eta_c^k}{N_p} \right) \right)}_{=: f_n(\eta_c^k)} \right)}^{=: \kappa_d} \right\} - \frac{1}{2} \lambda ||\eta_c||^2. \tag{8}$$

In order to obtain the gradient to this expression we need to consider the product rule for more than two factors.

$$\frac{d}{dx} \left[\prod_{i=1}^{I} f_i(x) \right] = \left(\prod_{i=1}^{I} f_i(x) \right) \left(\sum_{i=1}^{I} \frac{f_i'(x)}{f_i(x)} \right)$$

[7]. Here, we get that

$$\frac{\partial \mathcal{L}_{|\eta|}}{\partial \eta_c^k} = \sum_{p=1}^{P} \left\{ \bar{\phi}_p \mathbb{I}_{\{y_p = c\}} - \kappa_p^{-1} \frac{d}{d\eta_c^k} \prod_{n=1}^{N_p} f_n(\eta_c^k) \right\} \cdot \left(\sum_{n=1}^{N_p} \frac{\phi_{pn}^k \exp\left(\frac{\eta_c^k}{N_p} \right) \frac{1}{N_p}}{\sum_{k=1}^{K} \phi_{pn}^k \exp\left(\frac{\eta_c^k}{N_p} \right)} \right) - \lambda \eta_c, \tag{9}$$

where $\mathbb{I}_{\{y_p = c\}}$ is an indicator that is unity if the class of the user p is equal to c and zero else. Thus the first term in equation (9) is a sum of all $\bar{\phi}$ of the particular class. Unfortunately, setting this to zero and trying to solve for η_c does not yield a closed-form solution. In order to still get a parameter update for η_c we are forced to fall back to numerical methods. In this case we use a Conjucate Gradient Algorithm to obtain the parameter update for η_c [7].

3.7 Job Prediction

Our first and foremost goal with this analysis is to perform prediction. For this, we would continue the very same way as we set out before. We would first try and maximize the *ELBO* on the user's sequence of skills and then estimate the probability of obtaining the class label c by replacing the true posterior with the variational approximation. However,

for a user with an unknown class label we are not able to continue in the exact same way as before, since we needed the information on the class label in equation (4). Therefore, we drop all terms in equation (4) that incorporate the information of the class label. This leaves us with the new update for ϕ_{new}^{pred}:

$$\phi_{new}^{pred} \propto \exp\left(\Psi(\gamma) + \log \beta_k^s\right). \tag{10}$$

In a second step we estimate the probability of label c by making use of the variational distribution

$$\begin{aligned} E_q\left[p\left(y, \theta, z_{1:N} | \alpha, \beta_{1:K}, \eta\right)\right] &= \int \sum_{z_{1:N}} \exp\left(\eta_c^T \bar{z} - \log\left(\sum_{j=1}^{J} \exp\left(\eta_j^T \bar{z}\right)\right)\right) q(\theta, z_{1:N} | \gamma, \phi_{1:N}) d\theta \\ &= E_q\left[\exp\left(\eta_c^T \bar{z} - \log\left(\sum_{j=1}^{J} \exp\left(\eta_j^T \bar{z}\right)\right)\right)\right] \\ &\geq \exp\left(E_q\left[\eta_c^T \bar{z}\right] - \underbrace{E_q\left[\log \sum_{j=1}^{J} \exp\left(\eta_j^T \bar{z}\right)\right]}_{constant}\right), \end{aligned}$$

which leads us to the prediction rule

$$y^* = \underset{c \in \{1,\ldots,J\}}{\arg\max}\, E_q\left[\eta_c^T \bar{z}\right] = \underset{c \in \{1,\ldots,J\}}{\arg\max}\, \eta_c^T \bar{\phi} \tag{11}$$

where we can neglect the second term in the exponential function as it is the same for all classes [7].

3.8 Skill Recommendation

The second goal is to recommend skills. With a model in place the thought of using it for skill recommendation comes naturally. If a user is already adamant in her job choice and wants to extend her skills $\mathbf{s}_{1:N}$, which new skill $s^* \in \{1, \ldots, S\} \setminus \mathbf{s}_{1:N}$ should she pursue according to our model? One way of giving an answer to this problem is to look at the estimated class probability and recommend the novel skill that leads to the highest increase in probability for that particular class.

In the generative process the class probability is modeled by the sigmoid

$$p(y = j|z_{1:N}, \eta) = \frac{\exp\left(\eta_j^T \bar{z}\right)}{\sum_{c=1}^{J} \exp\left(\eta_c^T \bar{z}\right)},$$

which is not estimable due to the unobservable variable $z_{1:N}$, instead we estimate the class probability using $\bar{\phi}$, i. e. the expected values under the variational distribution given by

$$p(y = j|\phi, \eta) = \frac{\exp\left(\eta_j^T \bar{\phi}\right)}{\sum_{c=1}^{J} \exp\left(\eta_c^T \bar{\phi}\right)}. \tag{12}$$

We first need to find the latent Learning Profile that has the highest marginal effect on the estimated probability. That is, we need to find the k such that

$$\begin{aligned}
k^* &= \underset{k\in\{1,\dots,K\}}{\arg\max} \frac{\partial p\left(y = j|\phi, \eta\right)}{\partial \bar{\phi}^k} \\
&= \underset{k\in\{1,\dots,K\}}{\arg\max} \frac{\exp\left(\eta_j^T \bar{\phi}\right) \eta_j^k \cdot \sum_{c=1}^{J} \exp\left(\eta_c^T \bar{\phi}\right) - \exp\left(\eta_j^T \bar{\phi}\right) \sum_{c=1}^{J} \exp\left(\eta_c^T \bar{\phi}\right) \eta_c^k}{\left[\sum_{c=1}^{J} \exp\left(\eta_c^T \bar{\phi}\right)\right]^2} \\
&= \underset{k\in\{1,\dots,K\}}{\arg\max} \frac{\overbrace{\exp\left(\eta_j^T \bar{\phi}\right)}^{constant} \sum_{c=1}^{J} \exp\left(\eta_c^T \bar{\phi}\right) \left[\eta_j^k - \eta_c^k\right]}{\underbrace{\left[\sum_{c=1}^{J} \exp\left(\eta_c^T \bar{\phi}\right)\right]^2}_{constant}}.
\end{aligned}$$

This leads us to the rule for finding the latent learning profile k^*

$$k^* = \underset{k\in\{1,\dots,K\}}{\arg\max} \sum_{c=1}^{J} \exp\left(\eta_c^T \bar{\phi}\right) \left[\eta_j^k - \eta_c^k\right]. \tag{13}$$

Knowing this k^* enables us to recommend the skill

$$s^* = \underset{t\in\{1,\dots,S\}\setminus \mathbf{s}_{1:N}}{\arg\max} \beta_t^{k^*}, \tag{14}$$

which is the skill that is associated with the k^*th column of $\beta_{1:K}$, that has the highest

conditional probability and is not inside the user's set of skills already.

4 Data

With the model firmly established and the mathematical foundations laid we now only need data to train it with. The ideal data set for this task would consist of a unified taxonomy of skills and occupation titles. That is, a naming convention that ensures that each skill and each job is uniquely identified. Additionally, we require a collection of participants in a survey, which ideally are current or previous incumbents of their specific occupation title.

Unfortunately, to the best of our knowledge, such a data set does not exist. Or if it exists, it is not available freely for research purposes. While many companies are interested in creating a so called 'skill-graph' of their employees for internal purposes, this knowledge is usually kept a secret, presumably due to data privacy concerns.

Nonetheless, in the future Growify will collect such data from its customers, but for the present analysis and for creating a proof of concept, we have to make due with a data set that is available. One such data set is provided by O*net.

4.1 O*net

The Occupational Information Network (O*net) is sponsored by the U.S. Department of Labor/Employment and Training Administration and its main goal is to provide information about the professional occupation of U.S. citizens. O*net's self reported ambition is to provide valid data which is deemed essential for understanding the quickly changing nature of work [21].

The heart of the project is the O*net database, which contains hundreds of standardized occupation-specific descriptors on roughly 950 occupations and around 4400 unique skills. Along with these skills, O*net provides both ratings for the importance of these skills as well as ratings for how frequently the skills are used. Furthermore, O*net summarizes their skills under six categories: *ability, knowledge, skill, style, technology* and

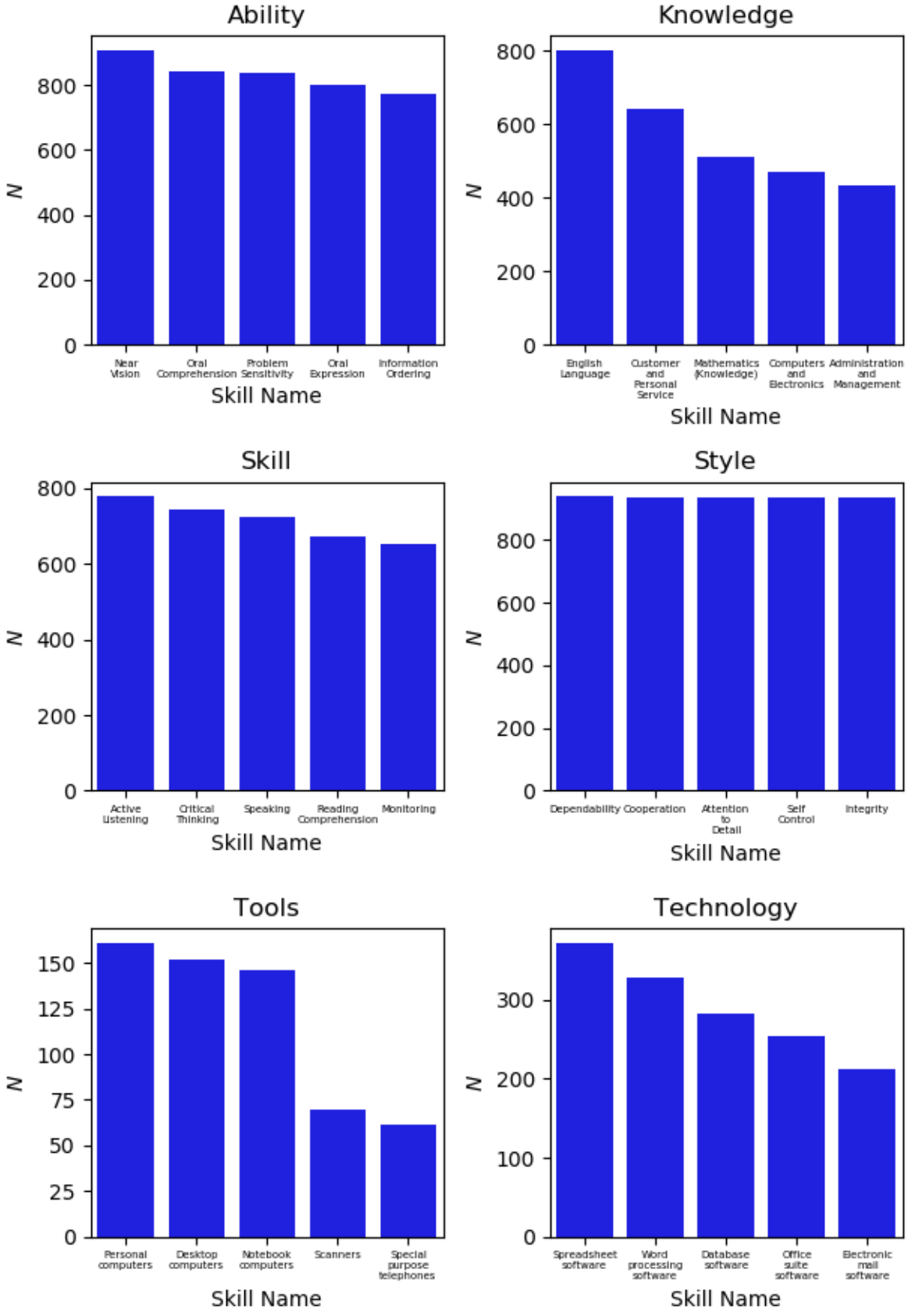


Figure 2: Most Frequent Skills by O*net Category

tools [21].

O*net defines *abilities* as enduring attributes of an individual, that affect performance. In particular, O*net discriminates between cognitive, physical, psychomotor and sensory abilities. Cognitive abilities influence the acquisition and application of knowledge. Physical abilities affect strength, endurance, flexibility, balance and coordination. Psychomotoric abilities are abilities that alter the capacity to manipulate and control objects. Finally, sensory abilities determine visual, auditory and speech perception [21].

According to O*net *knowledge* is an ordered set of principles and facts that applies in general domains. Most entries in this categories are skills that are traditionally acquired within an educational institution.

Skills are, as reported by O*net, developed capacities that ease learning or enable a faster acquisition of knowledge. Furthermore, complex problem solving skills are capacities used to solve novel, possibly ill-defined problems in complex real-world settings. Resource Management Skills facilitate the efficient allocation of resources. Social Skills are skills which aid in the work with people and in achieving goals. System Skills are developed capabilities to understand, monitor and improve socio-technical systems like social media. Finally, Technical Skills are skills used to design, set-up, operate and repair malfunctions of machines or technological systems.

Styles are personal characteristics that may affect job performance. For example, the 5 most frequent styles, summarized in figure 2, are dependability, cooperation, attention to detail, self control and integrity. Dependability in this context means the fulfilling of obligations. Cooperation requires being pleasant with others on the job. Attention to detail means a thoroughness in completing work tasks. Self control requires an incumbent to maintain composure, especially in difficult situations. Integrity asks of its holders to be both honest and ethical.

Technology collects information about information technology and software skills essential to the functions of an occupational role.

Finally, *tools* are defined as machines, equipment, and tools essential to the performance of an occupational role.

4.2 Problems

The fact that the data provided by O*net is the only data set available does not mean that it is perfect. Ideally we would have liked a data set that collects information on the skills of actual people along with their occupational title. This is decidedly not what O*net provides. As do all other providers of such data like the European Union or the OECD, O*net also provides only aggregated data. With only aggregate data at hand the picture one is able to paint can only be very broad. Since the unit of observation in this data set is not a single person but rather a skill itself, the only conclusion one can derive is an overview of an idealized version of any given occupation title. For example, the tool set of programming languages for a statistician consists, according to O*net of `R, SAS, MATLAB, Python, SQL` and `Microsoft Access`. However, one can be an exceptional statistician even if one possesses knowledge of only a subset of these tools.

A different avenue of problems lies in the currency of the data. The last update of O*net's database was in 2010. At the time of writing this paper, that was ten years ago. In an environment where occupational requirements change frequently the relevancy of this data may be prohibitively outdated.

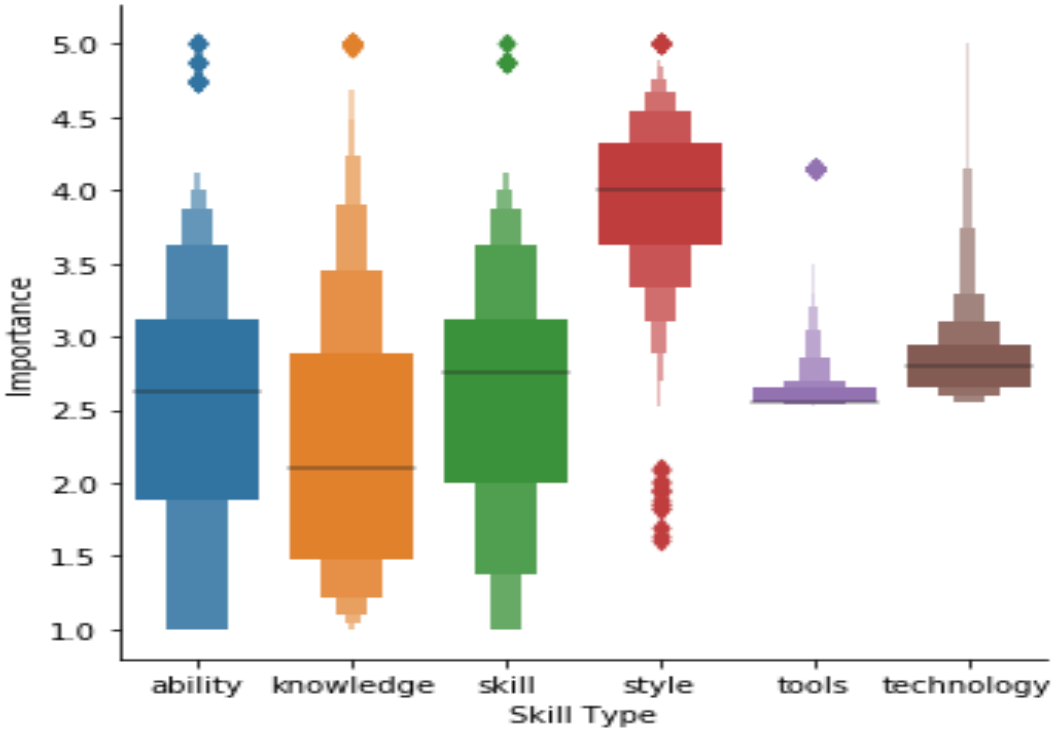


Figure 3: Distribution of Importance Rating by Category

Finally, the veracity of the data itself may come into question. Let's take for example a closer look at the *styles* category. The middle right panel of Figure 2 indicates that the top 5 skills in this category have been reported for any of the 940 occupations. While it may be true that dependability, cooperation, attention to detail, self control and integrity are important for a wide variety of jobs, their discriminating power as indicators suffers greatly if they appear in every occupational title. Which is made worse, as figure 3 indicates, by the fact that *styles* tend to get, on average and in most of the quantiles, the highest importance ratings.

On the other hand, the categories for *tools* and *technology* did not get a rating at all. We had to infer a rating based on the number of items per occupational title in both of the categories. Currently each item in the two categories gets the same weight plus a flat bonus for all skills. However, this means that an occupational title which is associated with many *tools* or *technologies* will get a lower importance rating for each item. On the other hand, if an occupational title is associated with few *tools* or *technologies* these items will get a high weight.

Of course the data has its flaws, but the alternative would have been to scrape data from curriculum vitae off of the internet, which would have brought its own set of problems. In scraped data one needs to curate a taxonomy of skills by oneself in order to make sure that one and the same skill is always associated with one unique name. However, for a proof of concept and a prototype we decided that this would be excessive. So instead we opted for using the O*net data set in order to create synthetic data.

5 Simulations

The lack of dependable data necessitates the creation of a synthetic testbed of data. We will assume that the distribution of skills provided in the O*net-database is indicative of the true distribution of skills. Furthermore, we assume that the importance rating of a skill that has been provided by O*net is proportional to the true probability of an occupation title's holder to have this particular skill. These are, most likely, very strong assumptions. Note that this influences the following interpretation of the results dramatically.

Due to the synthetic nature of the data these experiments can only claim to be valid within the confines of the simulation setting. However, we can try and mimic the data we expect to collect in the future. This way we still can gain valuable insights into the usefulness of the postulated model.

	Setting			
	(A)	(B)	(C)	(D)
λ	15	150	150	150
N	10	10	10	100
`noise`	0.1	0.1	0.3	0.3
`min_skills`	5	50	50	50

Table 1: Parameter Settings

The way we sample synthetic incumbents is as follows. First, for each occupation title and subject we determine her total number of skills n by drawing an integer from a Poisson distribution with rate λ. Next, we locate the subset of skills in the O*net-database that corresponds to the subject's job. These skills belong to all of the O*net-categories and constitute the true distribution of skills for this particular job. Out of this set of skills we draw n skills without replacement, where the probability of each skill is calculated to be proportional to its respective O*net importance rating. Finally, out of all other skills, which according to O*net do not belong to that particular occupation title's skill set, we sample a fraction of the user's n skills and add them as noise to the synthetic subjects skill set. We repeat this process N times for each of the occupation titles. The procedure is summarized in pseudo code in algorithm 2.

This leaves us with four essential hyperparameters for the simulations. The total number of subjects N per occupation title that are to be sampled. Their average number

Algorithm 2 Generating synthetic data

```
INPUT: O*net dataset, N, λ
OUTPUT: dataset of skills, along with labels
for all job ∈ O*net dataset do
    for 1, ..., N do
        n ~ Pois(λ)
        job_skills ← gather all relevant skills for job
        noise_skills ← {O*NET-skills}\ {job_skills}
        sample skills from job_skills
        sample skills from noise_skills
    end for
end for
```

of reported skills λ. The fraction `noise` of their total number of skills that is to be sampled as nuisance skills and a parameter `min_skills` which constitutes a lower bound to the number of skills that are to be sampled. All parameters of the simulations are summarized in table 1. With these four hyperparameters we can already cover a large number of scenarios.

Of these scenarios, setting A is intended to encompass the nascent phase of a start-up. The total number of incumbents to each job N is assumed to be as low as ten. With a user base that is not ready to engage with the system, that is the average number of reported skills is also assumed to be at fifteen which is assumed to be very low. The saving grace of this scenario is that the fraction of misreported skills is assumed to be as low as ten percent. This setting should prove challenging to all deployed algorithms. Unfortunately, we also deem this scenario to be very likely at least in the starting phase of a newly acquired customer.

Setting B is a juxtaposition to the previous case. If Setting A was to be the worst-case scenario, then setting B is in many ways the best-case scenario. The number of incumbents for each position is still very low, but the user's engagement with the platform is assumed to have improved sharply. With an average of 150 reported skills and a comparatively low percentage of misreported skills it should be easy for all deployed algorithms to achieve good predictive results.

Setting C is very similar to its predecessor in that there are again only a few holders for each occupation title, who nonetheless choose to engage with the system. The main difference lies in the fraction of nuisance skills, i.e. skills that do not belong to an incumbents' job's true skill distribution. We expect all indicators for the predictive performance of all algorithms to drop.

Next, Setting D is trying to investigate if it is possible to alleviate the possible misreporting of skills by a high number of subjects. That is, the `noise` parameter is still chosen to be high, but in response to that we sample more synthetic data points.

Finally, the last concession we had to make due to time and resource constraints was to limit the number of occupation titles. In order to still choose the selection of jobs that

proves to be the most challenging prediction task, we ordered all occupation titles by their skill overlap. To measure the overlap between two sets we use the overlap-metric for sets which is sometimes called the Szymkiewicz - Simpson coefficient [30]. It is defined as

$$c(X, Y) = \frac{|X \cap Y|}{min(|X|, |Y|)}.$$

We computed all pairwise overlap coefficients between all available occupation titles and chose the ten pairs with the highest overlap between them. This left us with 16 occupations which between themselves have almost identical skill sets. The only difference lies in the importance ratings provided by O*net which propagate down to the respective sampling probability for each skill conditional on the occupation title.

We claim that this limitation in the number of classes to predict does not compromise the validity of the results any further. In a real setting it would also be more appropriate to just include occupation titles that are already part of the organizational structure of a firm - and subsequently exclude unrelated occupations. For example it would certainly not make sense to encumber the training data of our models with the skill sets of nurses, when the future client is a small business specialized in electronics.

6 Results

In order to evaluate the predictive performance of the Supervised Latent Dirichlet Allocation algorithm we compare it against five alternative out-of-the-box algorithms. We investigate the algorithm in terms of accuracy, recall and precision in order to enable us to make an informed decision whether it makes sense to continue developing and refining the algorithm further or if it is more fruitful to look for alternatives. The accuracy score is defined as the number of correctly classified cases divided by the total number of cases. The precision of a particular class c is defined as

$$P(c) = \frac{number\ of\ correctly\ predicted\ cases\ of\ class\ c}{total\ number\ of\ predicted\ cases\ of\ class\ c}.$$

The recall for a particular class c is defined as

$$R(c) = \frac{number\ of\ correctly\ predicted\ cases\ of\ class\ c}{total\ number\ of\ cases\ that\ belong\ to\ class\ c}$$

[10]. We compare the Supervised Latent Dirichlet Allocation algorithm against a Multiple Logistic Regression, a feed forward Artificial Neural Network and several algorithms that are a combination of the classical Latent Dirichlet Allocation (LDA) algorithm whose output was used as input to a number of traditional classifiers. Those classifiers are a K-Nearest-Neighbor algorithm (KNN), a Logistic Regression (LR) and another feed forward Artificial Neural Network (ANN).

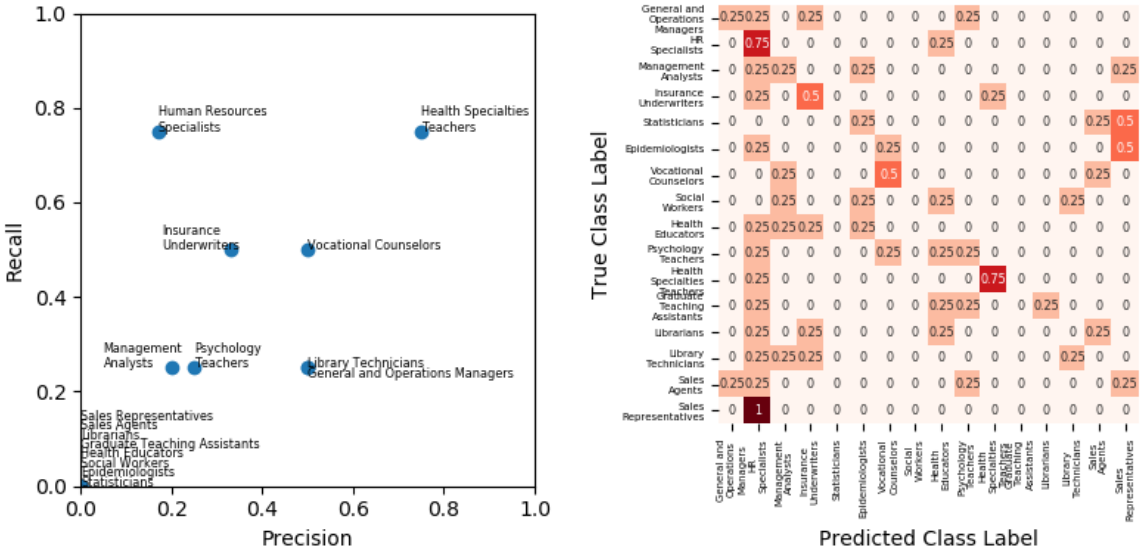
We chose the algorithms in order to compare a class of algorithms that operate on the unaltered space of the data, that is the Logistic Regression and the Artificial Neural Network use binary inputs where the skill set of each user is represented as a multi-hot encoded vector of size S. That tantamounts to a matrix X of dimensions $N \cdot J \times S$ for each simulation.

The other algorithms try to find a latent subspace of dimensionality K in the data, where K is much smaller than S. The key difference is that the classical Latent Dirichlet Allocation does so in an unsupervised fashion. We hope therefore to see a substantial improvement using the supervised version.

The hyperparameters for each algorithm have been tuned by a grid search on a train set, the accuracy has been evaluated on a separate test set. For the Logistic Regressions the most important parameter was to find the most suitable regularization parameter.

	Simulation			
	(A)	(B)	(C)	(D)
Logistic Regression	0.313	1.000	1.000	1.000
Artificial Neural Network	0.094	0.953	0.984	1.000
LDA + KNN	0.078	0.766	0.203	0.534
LDA + LR	0.203	0.828	0.578	0.694
LDA + ANN	0.156	0.344	0.593	0.687
Supervised LDA	0.219	0.813	0.672	0.516

Table 2: Overall Accuracy across Algorithms



(a) Recall vs. Precision

(b) Confusion Matrix

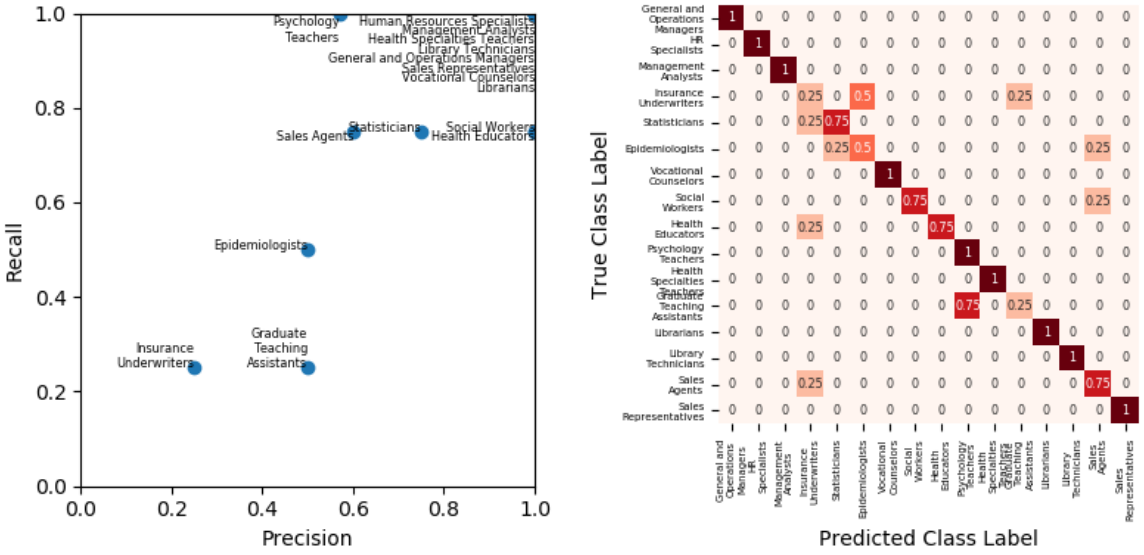
Figure 4: Results for Simulation A

For the Artificial Neural Networks we tried different architectures, optimization routines, batch sizes and learning rates. For all Latent Dirichlet Variants, supervised or not, the most important hyperparameter was K, i.e. the number of latent dimensions that are to be extracted. The last hyperparamter that was tuned was λ in the Supervised Latent Dirichlet algorithm, which controls the extend of the L_2 penalty, however unless it was set to zero we noted miniscule changes in the accuracy at best. If λ was set to zero the algorithm ran into numerical problems due to unstable weights in the sigmoid part of equation (12).

Table 2 summarizes the total accuracy of each algorithm for each simulation. In broad strokes we see exactly what we expected in the previous section.

All algorithms struggle with setting A, that is the worst-case scenario with only a limited pool of users which are not willing to engage with the system. In setting B, the best-case scenario, almost all algorithms achieve their best results, with a notable exception being the Neural Network that used the extracted topics of the original Latent Dirichlet Allocation algorithm. Also as expected the accuracy drops for most algorithms in the third setting that increased the fraction of nuisance skills for each subject.

In the last setting most algorithms manage to recover from a high fraction of mis-



(a) Recall vs. Precision

(b) Confusion Matrix

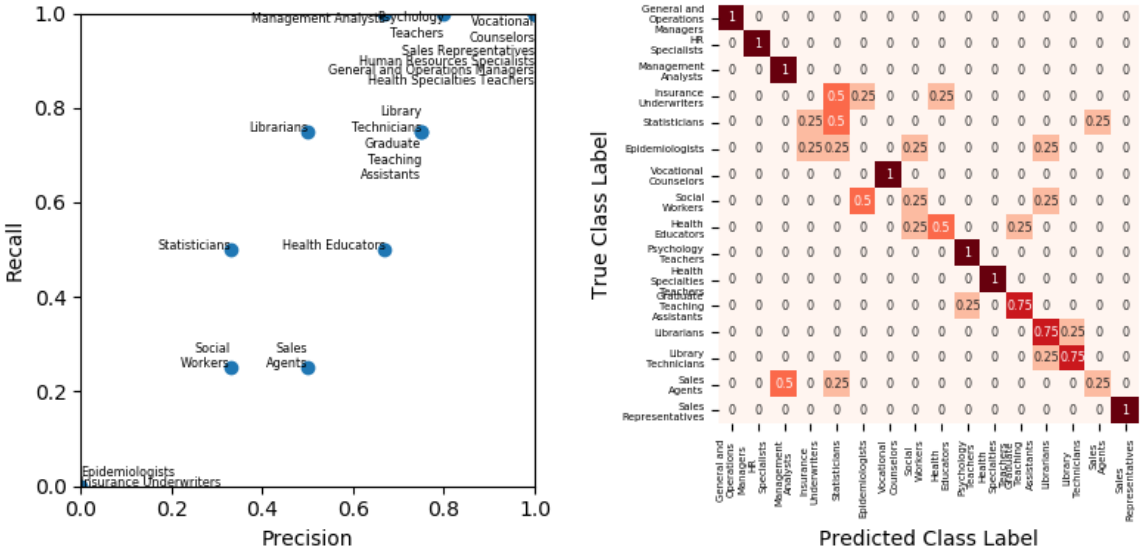
Figure 5: Results for Simulation B

reported skills if the number of users is sufficiently high. Unfortunately, this is not the case for the Supervised Latent Dirichlet Allocation algorithm. Its accuracy cannot quite bounce back to the previous level, leaving it as the worst algorithm in setting D.

It is worth to note that, overall, the algorithms that do not reduce the dimensionality of the input space at all tend to fare best. The multiple Logistic Regression for example achieves the best results across the board, followed by the Artificial Neural Network.

The Supervised Latent Dirichlet algorithm performs for the most part equally well or even better than its unsupervised variants. Especially in the heavy duty scenarios A and C it is the most performant of the algorithms that do deploy a dimensionality reduction technique. While it is not the best algorithm per se, these results are promising and do warrant further investigation. Especially, as prediction is not the only task. Let us now investigate the results of the Supervised Latent Dirichlet algorithm in detail.

As expected the results for simulation A are not inspiring. Statisticians get mistaken for epidemiologists, which seems fair but they also get confused with sales agents and representatives, which is less understandable. As indicated in the panels a) and b) of figure 4 in general, most classes do not get accurately predicted at all. Only Human Resource Specialists and Health Specialties Teachers achieve per class accuracies of 0.75%.



(a) Recall vs. Precision

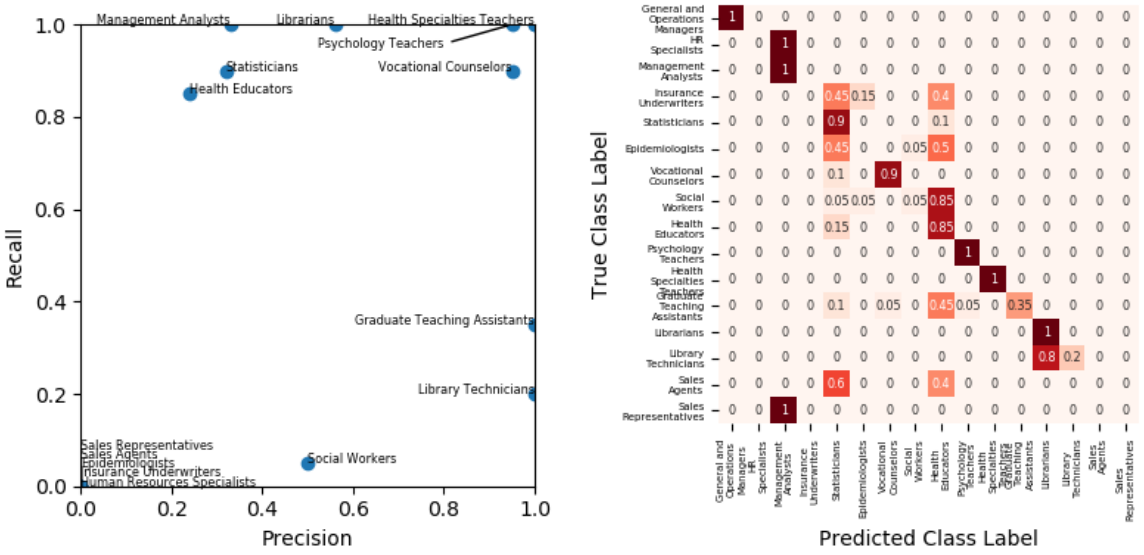
(b) Confusion Matrix

Figure 6: Results for Simulation C

Nonetheless, in this setting Supervised Latent Dirichlet Allocation performs second best, showing that it is capable of making the best use out of only very little information.

In simulation B we see a vast improvement. Keep in mind though, that even if 81.3% of all cases get predicted correctly, the Supervised Latent Dirichlet algorithm is only on the 4th rank of the six algorithms. Nonetheless, this does still constitute a good result, especially since the difference to the next best algorithm, a combination of traditional Latent Dirichlet Allocation and a multiple Logistic Regression, is only at 1.5 percentage points. Remember, the main problem lies in finding predictive latent structures that can be re-purposed into Learning Profiles, thus we are willing to make concessions regarding the accuracy of the Supervised Latent Dirichlet Allocation algorithm. Especially so with respect to the algorithms that do not offer latent dimensions at all.

Panel a) of figure 5 indicates that most, that is 9 of the 16 classes achieve a perfect score in both recall and precision. It is also worth looking into the miss-classifications for this setting. For example, graduate teaching assistants are very likely to get confused for psychology teachers, which looking at their skill sets according to O*net is a very understandable mistake to make. However, the miss-classifications are most endemic for insurance underwriters which get mostly confused with epidemiologists and graduate



(a) Recall vs. Precision

(b) Confusion Matrix

Figure 7: Results for Simulation D

teaching assistants. It may be possible that this peculiarity is a result from stopping the algorithm too early. Currently, the algorithm stops if the percentage change of the pool level *ELBO* is less than 1% and the accuracy on the train set did not improve with respect to the previous iteration. It may be that these criteria for early stopping are too lax in determining the convergence of the algorithm and need a more fine tuning.

For simulation C it is worth noting that, even if the amount of noise skills was increased substantially, the Supervised Latent Dirichlet Allocation algorithm fared quite admirably compared to the unsupervised variants. Out of the dimensionality reducing methods it managed to ignore the nuisance skills and subsequently performed best. Nonetheless, this does not mean that everything is perfect. Some classes get miss-classified in all of their respective cases. Again, as figure 6 shows, insurance underwriters and epidemiologists are the worst offenders. Insurance underwriters get confused with statisticians, epidemiologists and health educators. On the other hand epidemiologists tend to get confounded with insurance underwriters, statisticians, social workers and librarians. This is only explicable by a very unlucky draw in the simulations. In this simulation on average, subjects tend to report 150 skills which are related to their profession and 50 that are not. Under these circumstances it may be possible that one and the same nuisance skill appears in

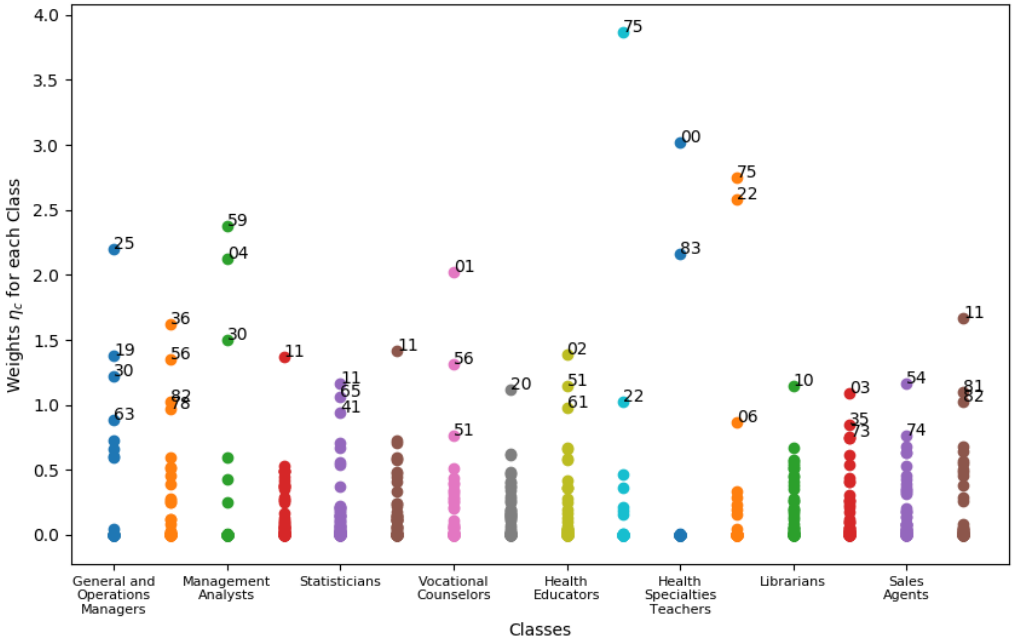


Figure 8: Model Weights η for each Class

multiple subjects or that the sheer number of noise is obfuscating the skills that are truly associated with the particular occupation title. In any case, these results demonstrate that the Supervised Latent Dirichlet Allocation algorithm is capable of extracting the essential skills for a profession even if only provided with little and messy data.

As indicated before increasing the number of subjects per occupation title did not help to improve the results at all. If anything it made them worse. Apparently, the algorithm is not able to make out the core skills for a given profession if the amount of noise is too high. Especially when comparing the panels a) of figure 7 with figure 6 the drop in predictive quality becomes apparent. There are more classes clustered around 0% recall and 0% precision. Also, most classes experience an acute drop in recall, meaning that as more cases belong to a class the model cannot keep up in predicting them correctly anymore. Especially human resources specialists, insurance underwriters, epidemiologists, social workers as well as sales representatives and agents are miss-classified in all their respective cases.

The most important and most difficult to achieve goal was creating automated courses, so called Learning Profiles. So how does the model manage in that regard? In order to

0	2
Writing	Liquid crystal display projector
Active Learning	Graphics or photo imaging software
Medicine and Dentistry	Personal digital assistant PDAs or organizers
Public Safety and Security	Medical software
Medical picture archiving computer systems PACS	Overhead projectors
Adaptability/Flexibility	Rate Control
Social Perceptiveness	Depth Perception
11	**19**
Personal computers	Object or component oriented development software
Learning Strategies	Technology Design
Auditory Attention	Dynamic Flexibility
Fluency of Ideas	Active Listening
Complex Problem Solving	Time Sharing
Speed of Closure	Word processing software
Management of Financial Resources	Far Vision
25	**30**
Hand trucks or accessories	Word processing software
Personal digital assistant PDAs or organizers	Spatial Orientation
Inventory management software	Configuration management software
Time accounting software	Active Listening
High vacuum equipment	Equipment Maintenance
Desktop calculator	Negotiation
Materials requirements planning logistics and ...	Selective Attention
51	**63**
Personal digital assistant PDAs or organizers	Video creation and editing software
Overhead projectors	Mobile phones
Photocopiers	Computer aided design CAD software
Liquid crystal display projector	Business intelligence and data analysis software
Digital video disk players or recorders	Personal computers
Graphics or photo imaging software	Financial analysis software
Customer relationship management CRM software	Operation Monitoring
65	**83**
Object or component oriented development software	Complex Problem Solving
Enterprise resource planning ERP software	Laboratory mixers
Analytical or scientific software	Personnel and Human Resources
Equipment Selection	Urinalysis analyzers
Glare Sensitivity	Posture evaluation kit
Gross Body Equilibrium	Philosophy and Theology
Night Vision	Fume hoods or cupboards

Table 3: Skills associated with Latent Topics

asses that, we look closely at simulation B as it provided the best predictive results, which in turn promises the best extracted latent factors out of all fitted models. Table 3 presents a selection of extracted latent topics and their associated words sorted by each skill's estimated probability conditional on the latent topic. The weights associated with each topic for each profession are visualized using a dot-plot in figure 8. Of course not all topics allow for a meaningful interpretation. Some topics, for example compare topic 0 with topic 11 and topics 51 and 63, seem to almost be permutations of each other. Most of their respective first entries are exactly the same and deviations only start to show in some of their later entries.

Some topics however, carry a serendipitous meaning. The very first topic and topic 83, for example gather some medical terms and also get the two highest weights in the model for health specialties teachers.

Topic 2 also contains terms associated with medical equipment and display technologies, such as projectors, imaging and medical software. It is the most impactful topic for health educators. However, the second most important topic for health educators, topic 51, places the highest probability on very similar skills.

As for statisticians, the most important topic, topic 11, seems to be associated with analytical skills, such as learning strategies, complex problem solving and speed of closure (i.e. readiness of mind). As a nice aside, topic 11 appears in many of the other predominantly analytical professions as insurance underwriters, epidemiologists and sales representatives. The second most important topic for statisticians, topic 65, places the three highest values of probability mass on software related skills.

However, the most appealing results are related to general and operations managers. Their most important latent topic, according to the model, is topic 25. Obviously, hand trucks and high vacuum equipment need to be ignored, but all of the other skills focus around time management and accounting software and devices. The second most important topic, topic 19, involves a mixture of social skills, such as active listening and time sharing, and office related software such as word processing editors. Word processing software also appears in topic 30, which is the third highest ranked topic according to

the model. Topic 30 also seems to constitute a mixture of soft skills like active listening, selective attention and negotiation and some technical skills such as word processing software and configuration management software. Finally, topic 63 consists of tools of the everyday office life such as mobile phones, personal computers, financial and data analysis software.

All in all the topics seem to be very reasonable. Most professions are associated with related skills and the selection of skills into latent factors seems to be, for the most part, logically coherent. Even if some methods are consistently better in the prediction task than the Supervised Latent Dirichlet Allocation algorithm it should not be discounted prematurely, as the other algorithms do not offer the automated grouping of skills into learning profiles. The Supervised Latent Dirichlet Allocation is mostly comparable to it's unsupervised alternative algorithms or even better. It performs especially well under duress, that is if users provide only little information or if they provide substantial amounts of essentially unrelated skills to their profession.

Unfortunately, these results aren't as decisive as we wished for them to be. As the goal of the Supervised Latent Dirichlet Allocation algorithm was to provide a roundabout way of recommending both jobs as well as skills and to aid in the creation of automated courses. We need to concede that not all of the goals have been met. The consequences and ramifications of this are discussed in the next section.

7 Conclusion

In this analysis we have set out to provide an algorithmic solution for career recommendations. We identified the problem to be threefold, that is that a unified recommendation system needs to first, assist in the proper selection of a future profession. Second, it needs to aid in pursuing the most effective way of improving oneself in a given line of work. And finally the system needs to expedite the automatic generation of courses that assist users in achieving their career related goals. These automated courses will, if implemented well, enable Growify to reduce the required domain knowledge for each profession. Thus making its business model economically more viable.

To this effect we formulated a Supervised Latent Dirichlet model that is capable of providing answers to all three questions. We derived ways of estimating the parameters of this model as well as how to obtain predictions from it. Additionally, we sketched ways how to recommend the next skill to be acquired given a trained model. We implemented this model in the `Python` programming language [28].

Next we acknowledged that currently there is no suitable data set for estimating this problem in a credible and valid way. Nonetheless, we managed to construct different simulation settings. These synthetic data sets can of course not replace real life data, however these data sets contain an innate structure and we managed to show that the algorithm is capable of extracting it. The algorithm showed the best results when under duress, that is it was comparably unaffected by a substantial lack of data and it still performed rather well when trained with messy data.

Even if the algorithm was outperformed by methods that do not reduce the dimensionality of the feature space this does not mean that these methods will also outperform the Supervised Latent Dirichlet Allocation on real data. It may even be that the grid used for hyperparameter tuning was simply too coarse and that a more fine grained search would lead to better results. But even if the algorithm continues to get outperformed, there is no reason for not using a multiple logistic regression for the recommendation of skills and jobs instead.

As for the latent topics that are supposed to be the foundation of automated Learning Profiles the results are very satisfying so far. Most of the topics do show a coherent clustering of skills that are relatable to a profession. Thus the most pressing problem seems to have found an adequate solution. Note however, that these findings also have to be evaluated against real data.

For the future it is imperative to acquire real world data from Growify's first customers. Additionally, it would be helpful to increase the convergence speed of the algorithm. As it currently was only a prototype whose task was only to provide a proof of concept, there are still multiple ways of improving the algorithm's implementation. For example, it is possible to drop the one-hot encoding of the skills that were used for notational

convenience in favor of a multi-hot encoding that would facilitate the use of vectorized operations. Finally, as each users *ELBO* is independent of all other users this naturally opens avenues for parallelization, maybe even computation on Graphical Processing Units. If it is possible to speed up the algorithm due to a more efficient architecture it may be possible to tune the hyperparameters on a finer grid which subsequently may lead to better predictive results. However, all of these improvements are out of the scope of this work.

All in all, the prototype constitutes a decent first step to a unified solution to the three problems of job- and skill-recommendation as well as the automated generation of Learning Profiles. Still, further improvements are possible and even required for a use in a production environment.

References

[1] Fabian Abel, Yashar Deldjoo, Mehdi Elahi, and Daniel Kohlsdorf, *Recsys challenge 2017: Offline and online evaluation*, Proceedings of the eleventh acm conference on recommender systems, 2017, pp. 372–373.

[2] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe, *Variational inference: A review for statisticians*, ArXiv **abs/1601.00670** (2017).

[3] David M. Blei and Jon D. McAuliffe, *Supervised topic models*, Nips, 2007.

[4] D.M. Blei, A.Y. Ng, and M.I. Jordan, *Latent dirichlet allocation*, the Journal of machine Learning research **3** (2003), 993–1022.

[5] Keith Bradley, Rachael Rafter, and Barry Smyth, *Case-based user profiling for content personalisation*, Ah, 200006.

[6] Tommaso Carpi, Marco Edemanti, Ervin Kamberoski, Elena Sacchi, Paolo Cremonesi, Roberto Pagano, and Massimo Quadrana, *Multi-stack ensemble for job recommendation*, Recsys challenge '16, 2016.

[7] W. Chong, D. Blei, and F. Li, *Simultaneous image classification and annotation*, 2009 ieee conference on computer vision and pattern recognition, 2009June, pp. 1903–1910.

[8] Vachik S. Dave, Baichuan Zhang, Mohammad Al Hasan, Khalifeh AlJadda, and Mohammed Korayem, *A combined representation learning approach for better job and skill recommendation*, Proceedings of the 27th acm international conference on information and knowledge management, 2018, pp. 1997–2005.

[9] M. Deshpande and G. Karypis, *Item-based top-n recommendation algorithms*, ACM Transactions on Information Systems (TOIS) **22** (2004), no. 1, 143–177.

[10] Mamadou Diaby, Emmanuel Viennet, and Tristan Launay, *Exploration of methodologies to improve job recommender systems on social networks*, Social Network Analysis and Mining **4** (201412).

[11] Jorge Martínez Gil, Bernhard Freudenthaler, and Thomas Natschläger, *Recommendation of job offers using random forests and support vector machines*, Edbt/icdt workshops, 2018.

[12] David Goldberg, David A. Nichols, Brian M. Oki, and Douglas B. Terry, *Using collaborative filtering to weave an information tapestry*, Commun. ACM **35** (1992), 61–70.

[13] Prem Gopalan, Jake Hofman, and David Blei, *Scalable recommendation with poisson factorization*, ArXiv (201311).

[14] Prem Gopalan, Jake M. Hofman, and David M. Blei, *Scalable recommendation with hierarchical poisson factorization*, Uai, 2015.

[15] Miao Jiang, Yi Fang, Huangming Xie, Jike Chong, and Meng Meng, *User click prediction for personalized job recommendation*, World Wide Web **22** (2018), 325–345.

[16] Sangeetha Kutty, Richi Nayak, and Lin Chen, *A people-to-people matching system using graph mining techniques*, World Wide Web **17** (2013), 311–349.

[17] Georgios Lekakos and George. M. Giaglis, *Improving the prediction accuracy of recommendation algorithms: Approaches anchored on human factors*, Interacting with Computers **18** (2006), 410–431.

[18] Thomas Minka, *Estimating a dirichlet distribution*, M.I.T, 2003.

[19] Amber Nigam, Aakash Roy, Hartaran Singh, and Aabhas Tonwer, *Job recommendation through progression of job selection*, ArXiv **abs/1905.13136** (2019).

[20] Ankhtuya Ochirbat and Timothy K.Shih, *Occupation recommendation with major programs for adolescents*, Telematics Informatics **35** (201712), 534–550.

[21] ONET, *National center for onet development*, 2020. https://www.onetonline.org/, Accessed: 2020-03-04.

[22] Ioannis Paparrizos, B. Barla Cambazoglu, and Aristides Gionis, *Machine learned job recommendation*, Proceedings of the fifth acm conference on recommender systems, 2011, pp. 325–328.

[23] Paul Resnick and Hal R. Varian, *Recommender systems*, Commun. ACM **40** (1997), 56–58.

[24] M. Sato, Koki Nagatani, and Takuji Tahara, *Exploring an optimal online model for new job recommendation: Solution for recsys challenge 2017*, Recsys challenge '17, 2017.

[25] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen, *Collaborative filtering recommender systems*, The adaptive web: Methods and strategies of web personalization, 2007, pp. 291–324.

[26] Yelong Shen and Ruoming Jin, *Learning personal + social latent factor model for social recommendation*, Proceedings of the 18th acm sigkdd international conference on knowledge discovery and data mining, 2012, pp. 1303–1311.

[27] Sherry Sullivan, *The changing nature of careers: A review and research agenda*, Journal of Management - J MANAGE **25** (199906), 457–484.

[28] Guido Van Rossum and Fred L. Drake, *Python 3 reference manual*, CreateSpace, Scotts Valley, CA, 2009.

[29] C. Vialardi, J. Braver, L. Shaftr, and A. Ortiaosa, *Recommendation in higher education using data mining techniques*, EDM'09 - Educational Data Mining 2009: 2nd International Conference on Educational Data Mining (200901), 190–199.

[30] M. K. Vijaymeena and K. Kavitha, *A survey on similarity measures in text mining*, Machine Learning and Applications: An International Journal **3** (201603), 19–28.

[31] Chong Wang, *class-slda/opt.cpp*, 2009. https://github.com/blei-lab/class-slda/blob/master/opt.cpp, Accessed: 2020-03-12.

[32] Murat Yagci and Fikret Gurgen, *A ranker ensemble for multi-objective job recommendation in an item cold start setting*, Proceedings of the recommender systems challenge 2017, 2017.

[33] Qing Zhou, Fenglu Liao, Chao Chen, and Liang Ge, *Job recommendation algorithm for graduates based on personalized preference*, CCF Transactions on Pervasive Computing and Interaction (201911).

8 Declaration of Authorship

I, Thomas Siskos, hereby declare that I have not previously submitted the present work for other examinations. I wrote this work independently. All sources, including sources from the Internet, that I have reproduced in either an unaltered or modified form (particularly sources for texts, graphs, tables and images), have been acknowledged by me as such. I understand that violations of these principles will result in proceedings regarding deception or attempted deception.

Berlin, March 30, 2020