Intelligent* Task Scheduling

Luke Weber, *Undergraduate*, Washington State University,
Professor Jana Doppa, *Ph.D.*, Computer Science 570

***Abstract*—Here we focus on methods for optimally assigning tasks to persons within organizations of varying populations. Users are selected based on their matching criteria with the given task, which is trained by some moderating agent, on the fly. Consequently, these individuals are assigned tasks which, over enough training, represent a current understanding of their many different measured proficiencies, as we test and attempt to model every person's performance on tasks which address different performance metrics.**

I. Introduction

A. Motivation

Past the submission deadline for my course project proposal, I discovered not only that I had not submitted the proposal, but that I was not even aware of the existence of the assignment. Naturally, I was quite worried. Hurrying to think of something, with not many creative problems coming to mind. It was in my search that, as I was sitting at my computer brainstorming ideas, my roommate knocked on the door and asked me to do some dishes. He is always on me to do chores. This sparked a chain of thoughts: the comparison between me and him is not fair. Our schedules are not even close to similar; I'm simply a much busier student than him, and he has more time for such things, but that does not mean I get to slack off and make him do all the work!

I told him what I always tell him and what he always hates to hear: "I'm busy right now, but I might be able to do some chores later." He walked away, dejected as usual.

It was about here that I reflected on this little domestic issue a tad further. What if we could develop a model to understand how busy people are, what they are available to do, and what they really *like* doing? It's a far-reaching problem, and this is where my course project begins.

B. Challenges

"Biting off more than you can chew." This is arguably my primary fault, my downfall on many tasks and pasts projects. I know for myself that I simply get too ambitious (or perhaps nave), disregarding obvious warnings and constraints—it sometimes even seems that I believe there are more than 24 hours in a day, more than 7 days in a week, more than 4 weeks in a month, and more than 12 months in a year. This all sparks from my love for tackling big projects, and this is a pretty big project.

In fact, a main problem faced was the scale; finding a scalable and general solution is pretty tough, and tougher still when it must be done within a semester amongst other demanding coursework.

Nonetheless, not all challenges were strictly about the size of the project. Concerning technical issues, it took some time coming up with methods to obtain features that mattered. Many searches were conducted and all lead to the realization that, on the surface, the data available wasn't directly portable into my application. Although, even if I had such satisfactory data, some things were still not clear to me. There was an important decision to be made on which features really counted, i.e. what information would allow us to learn what we needed to learn. This took a unique brew of consideration, experimentation, and then more consideration.

Another aspect to consider on this project was the conditional dependence between features, and if we could even simplify the process by assuming no conditional independence. Furthermore, I didn't know off the top of my head if the data is or could be linearly separable, especially as features would grow and shrink. These concerns were both subject to experimentation.

Finally, a big challenge of mine was to develop a program that is better than a mere score-counter, recommending only the person with the highest score for a prompted task. This is not intelligent, not what machine learning is about, and not the intention of the project.

C. Solution Approach

A key insight into approaching a solution was that the problem could be simplified into something which wouldn't be the source of a heart attack—something simple. For this reason, it was decided that we only solve the issue of determining the best person in an organization for any given task for which we know its approximate feature composition, i.e. the weights of each feature (performance metric) with respect to the task.

It was mentioned that getting input to our algorithm would be tough. The process of actually acquiring the data was solved with one of my few "ah-ha!" moments on this project: we don't need data to start. Let's learn it on the fly, starting from nothing.

Upon this, another question arose. Where do we get features, and which ones matter? This was another thought with a nontrivial answer. The solution was to generalize the features to be set by the implementing organization. I know it sounds like a cop-out, but it isn't necessarily so. Rather it allows for a more general utilization by not fixing the features so that we are limited to specific or generic performance measures.

At first, before I had this realization, I thought it pertinent to use features that each are on a 0-10 scale with respect to the "Big Five" personality traits:

* Getting there.

1) Extraversion,
2) Agreeableness,
3) Openness,
4) Conscientiousness, and
5) Neuroticism.

But, immediately this did not seem sufficient. It's fairly easy to see that correlations between these traits and performance on any particular task would be too weak to deem useful. Could someone really tell how good of a programmer, dishwasher, leader, or cook based soley on these traits? I didn't think so, so I looked for something better.

Getting closer, I found myself looking at an article by *CrowdFlower*, a company based on solving data problems using machine learning, which gave a much more detailed and relevant set of performance areas that would be much more likely to help a classifier learn a person's skills. They are as follows:

1) Short-term memory,
2) Attention,
3) Sentiment,
4) Training,
5) Visual complexity,
6) Verbal complexity,
7) Opinion, and
8) Categorization.

Better, but not quite as general as is preferred. There is no telling that these features will help every organization assign tasks. This is about when I converged on adopting the general form of definable features as

$$x_i =< f_0, f_1, ..., f_m >$$

where $0 \leq i \leq n$, for n training examples, and $0 < m$ is the number of features defined for the given organization. This way, they can define what matters most to them from the start, and our solution is not restricted.

D. Results

There are no empirical results from the experiments. They are still being conducted. However, based on the experiments carried out thus far, there is more of a reason to believe that, with properly crafted features, my program will achieve an acceptable accuracy applicable to many different fields. Although, this is just a belief.

II. Problem Setup

We mentioned before that our goal is as follows: learn the proficiencies of each person within an organization so that we may approach optimally assigning them tasks. Each group has a unique set of m features with very low conditional dependence.

We need a solution which performs well when there is *concept drift*, so that we are always on our toes keeping up with any given individual's current skill-set. Also, it is required that our solution is multi-class, where each class represents a person's ID number.

III. Experiment and Results

Once all I tried my luck with a Support Vector Machine (via LibSVM), I decided to use a Passive-aggressive classifier (using scikit-learn library) primarily because of the support for concept drift, which is pretty essential in our case. Also, the library which implemented the PA algorithm had pretty good default hyper-parameters setup, ready to go.

The main (partially successful) experiment, which can be partially seen in Figure 1, was conducted by first creating artificial information about a non-existent software company and its employees. The features were as follows:

1) Software development
2) Web development,
3) Information technology,
4) Social skills, and
5) Software testing.

I then hashed-out the skillset of each of the employees; in this case, there were just four: Jessica, Ron, Patricia, and Qu. Each of them had a variety of proficiencies, all of them with at least one dominating skill. Using these as a guide for my training of the PA classifier, I then formulated tasks which first would target specific features as being most important, and then threw in some combination ones, all the while "knowing" which employee would perform the best given there skillsets.

It is noted that this experiment is inherently biased, thus the reason I have claimed no empirical results.

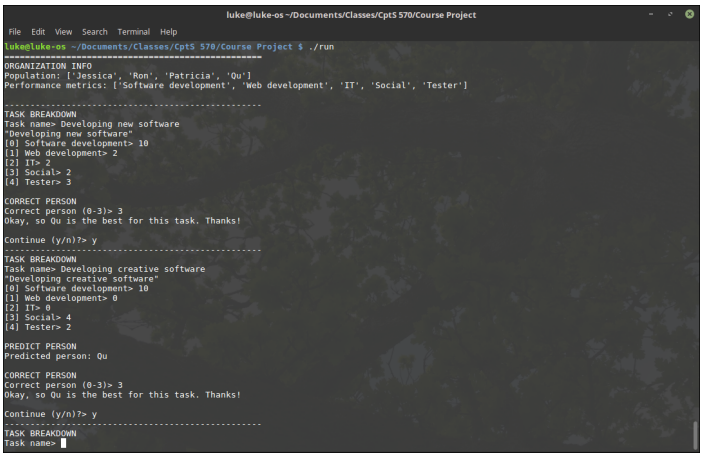

Fig. 1. Terminal in which interactive experiment was conducted

IV. Conclusion and Future Work

Working on this problem was very satisfying, although I wasn't able to complete my work. Once this component of the whole "Intelligent Task Scheduling" project is tested and validated, there can then be progress on the other parts. I plan on taking this up in the following months, where one of the primary points of focus is finding data that is not self-generated, and then following up by running a real experiment in a real organization, however big the population.