# How I Learned to Stop Worrying about Spamming and Starting Loving Bayes' Theory

Tugluk Letif

May 5, 2024

## 1 Introduction

As part of the data initiatives at Dice[1], we are tasked with tackling one of our most urgent technological issues: spamming. There are two parts to this puzzle: job posts and job candidate profiles. The old solution involved a mass email campaign which, as long as there were job openings in any field, would involve emailing candidates about the job without considering the candidates' skills. This is considered beyond the usual type of advertisement spamming because it involves emails targeting the wrong audiences. For example, a job for a developer emailed to a software engineer is considered appropriate, yet the same job sent to a dental hygienist, although a nice gesture, is still considered spamming. It has cost our company dearly, as it can cause users to disengage with our service and, in the worst-case scenario, sue us for wrongful advertisement.

In this project, we propose a Bayesian solution to address the ongoing spam problem. We begin by examining our internal data, supplemented with some mock labeled data, to tackle this business question. Next, we introduce the data cleaning process for training, followed by an examination of two traditional non-spam solutions (one involving machine learning). We then segue into the Bayesian solution for our spamming problem. We conclude the project with results and a discussion on future work.

## 2 Data

The dataset resides in the company's Athena table, where we queried approximately 500 job postings from our website. These postings are akin to typical job descriptions found on the internet and include candidate keywords. This dataset consists of HTML-converted text data that includes job titles, skill requirements, salary information, and potentially company information as well. The second dataset is our user candidate data, which includes personal information, skills, locations, and previous job titles, among other details, in JSON format from our GraphQL database. For this particular project, we will query approximately 500 candidates' skills and job titles and store them in Python string lists. The variables of interest are the words contained in both the job descriptions and candidate data, e.g., 'Software', 'Python'. The logic is to distinguish the profiles and job IDs while allowing an n-to-n match for the purpose of multiple spam labelings for the same candidate. Main data were fetched before the process represented in the query below:

```sql
SELECT DISTINCT mr.job_id, mr.profile_id,
jd.id, jd.title, jd.description,
ask.concatenated_skills
FROM "prod_matchology_annotations"."match_result" mr
INNER JOIN "prod_matchology_annotations"."job_dice"
    ↪ jd ON mr.job_id = jd.id
INNER JOIN AggregatedSkills ask ON mr.profile_id =
    ↪ ask.profile_id
LIMIT 500
```

Significant pre-processing was performed to merge these two datasets, as represented in the table below:

| | Job 1 | Job 2 | ⋯ | Job n |
|---|---|---|---|---|
| Candidate 1 | Spam | Not Spam | ⋯ | Spam |
| Candidate 2 | Not Spam | Spam | ⋯ | Not Spam |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| Candidate n | Spam | Spam | ⋯ | Not Spam |

Each row represents a candidate, featuring their skill sets and former job titles, while each column represents a job posting with its description. At this stage, if we treat the skill keywords in each column as variables, we should observe a negative correlation between jobs classified as 'Spam' and those classified as 'Not Spam.' In the next section, we will discuss how we clean the data

### 2.1 Data cleaning

Like many data-intensive applications, the biggest challenge of this project is the data itself. We started with no labeled data for spam and non-spam categories. Therefore,

[1]Dice.com is known for providing AI-powered software products and talent acquisition services, delivering career marketplaces specifically tailored to candidates focused on technology roles, such as software engineers.

for the proof of concept (POC) purposes, we queried 500 data points from an old matching system that has some basic matching criteria based on our clients' feedback. There are 30 unique jobs and 12 unique candidate datasets which have an n-to-n match as follows:

| Description | Concatenated Skills |
|---|---|
| Required Skills: Data Lake, DataIKU, Hadoop, ... | Software, ETL, QA, cron, Reporting, Monitoring... |
| Hello My Client is Hiring for Remote Azure Data Architect ... | ETL, Conversion Tool, Q Test, JIRA, TFS, H... |
| DOCTYPE html html head head body p st... | Java, Spring MVC, Software engineering, Multit... |
| Top Skills & Years of Experience: 7+ years of... | Software, JIRA, Linux, Apache Ant, Apache Maven... |

Unfortunately, many job descriptions contained non-rendered HTML. Some job descriptions were too wordy, while others were too brief. To strike a balance, we deleted entries that had more than 200 words or fewer than 20 words, while also converting the remaining HTML back into string text format. We also noticed repetitive and nested skills in the candidate profiles, so we recursively deduplicated that column as well.
Secondly, we decided to pivot the data based on candidate skills and marked the matching data as 'Non-Spam,' while labeling the rest as 'SPAM' to simulate a 2-D dataset. After a long and tedious process, we began to see data resembling the heatmap shown in Figure 1.

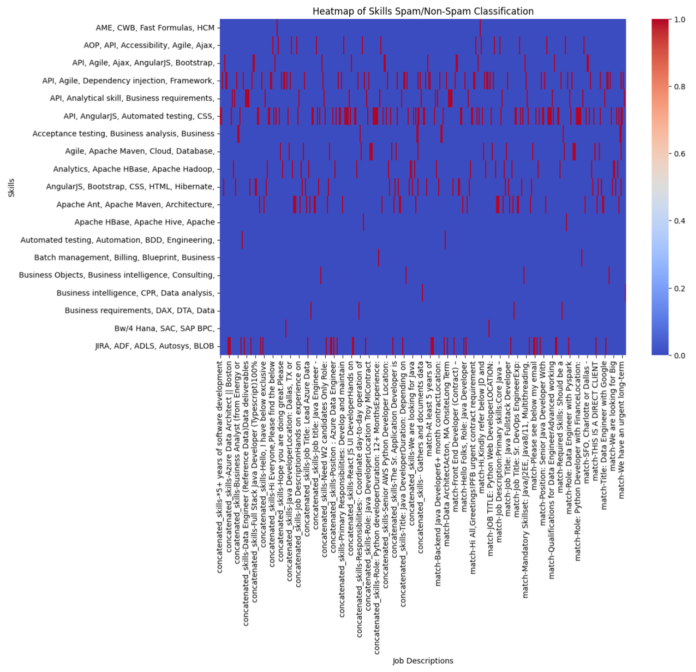

Figure 1: Data heatmap: we observe a negative correlation between jobs classified as 'Spam' and those classified as 'Not Spam.'.

## 2.2 Testing sets

Although our Bayesian approach deviates from traditional machine learning practices, where you typically have three sets of data: training, evaluation (for fine-tuning), and testing on an untrained set, we do reserve a small portion of data for result evaluation purposes. Specifically, 10% of the pivoted data will be set aside and pickled for final result evaluation.

## 3 Methods

Three of our many proposed methods worked without any debugging issues. We will first introduce the non-Bayesian machine learning and regular expression solution. After comparing the pros and cons, we will then delve deeply into our own Bayesian filtering solution.

### 3.1 Approach I: Embedding

Matching algorithms are an actively developing field. One successful model is embedding matching, which, when applied to our problem, can be broken down into two parts:

1. Convert the job description and candidate profile separately into embedding vector space.

2. Compare the cosine similarity between the two vectors and determine whether they are closely related (match/non-spam) or not (spam).

We will not delve into the mathematical details of how embedding work, as this involves complex aspects of natural language processing. However, the basic concept is illustrated in Figure 2, where keywords are categorized with part-of-speech labels and the grammatical relationships between words are identified. Following PCA, the numerical representations are stored in a tuple.

Once the text of both the job description and candidate profile is converted into numerical tuples, we can then use trigonometric formulas to calculate the cosine angle between the job and each candidate. A larger angle represents less similarity, while a smaller angle indicates a match, as shown in Figure 3.
Although this process is complex and beyond the scope of this class, there are many mature libraries, such as SpaCy, that can yield ideal results.

However, there are several drawbacks/cons to this method as well. First and foremost is the size of each package. Although it may not be a significant issue when using a local machine for research and demonstration purposes, the sheer size of each model, typically around 100MB, makes it challenging to implement in a production environment. This is especially true when considering multiple versions and dependencies when building software based on these pre-existing libraries. Second, it does not guarantee high

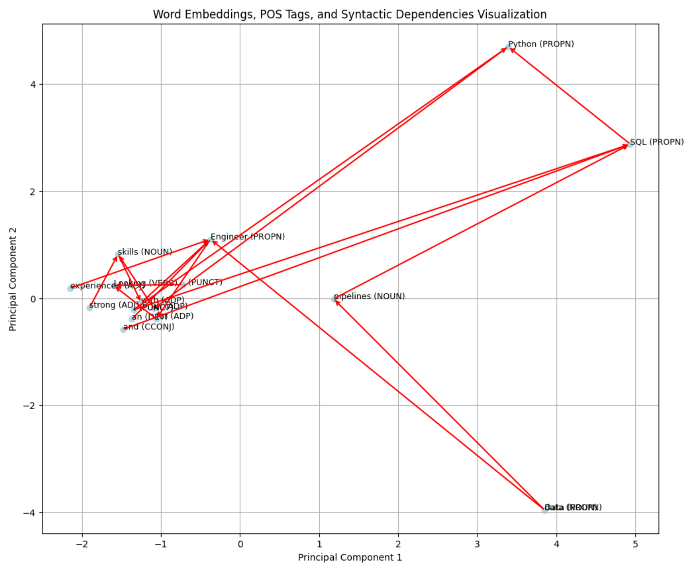

Figure 2: This plot shows the 2D PCA-reduced vector embedding of words from a job description. Arrows indicate syntactic dependencies, highlighting the grammatical structure. Words closer together are more semantically related.

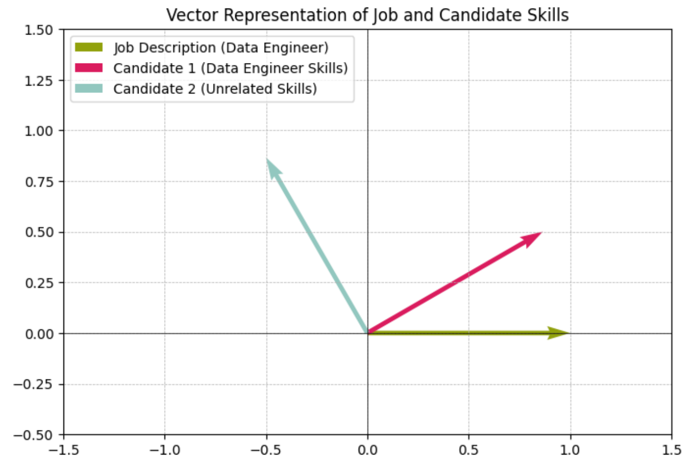

Figure 3: The angular relationships between the job description and candidate skills. Candidate 1's skills are more aligned with the job requirements compared to Candidate 2.

accuracy even if we set the threshold higher. Next, we will introduce another solution that avoids the use of large pre-existing libraries.

## 3.2 Approach II: Bag of words

What if we chose not to use a large pre-built package and instead rely on regular expressions and counting? Our Bag of Words (BoW) approach is a simplified version of Natural Language Processing, although it does not involve embedding. In this method, we collect keywords from candidate skills and compare them with the keywords in the job description. We then determine whether there is a match based on the percentage of keywords that overlap. The simple algorithm is outlined as follows:

1. Normalize and split skills into a set of keywords.
2. Normalize and tokenize the description.
3. Calculate intersection of skills and description words.
4. Calculate percentage of skills present in the description.

However, this method does not achieve high accuracy and struggles to accommodate new skill words. Its functionality in this context is largely due to our extensive list of skill sets, which have been mined through profiling processes for candidate profiles. To address these limitations, we will introduce our Bayesian method in the next section.

## 3.3 Approach III: Bayesian Spam Filter

In this section, we will discuss the method we have chosen for this project: Bayesian spam filtering. The underlying concept is straightforward: Given a job description and candidate skills, the Bayesian matching function calculates the probability of a match based on the relevance (non-spam) versus irrelevance (spam) of the skills. The formula is derived using Bayes' Theorem as follows. Let:

- $P(\text{non-spam})$ be the prior probability of a match (non-spam).
- $P(\text{spam}) = 1 - P(\text{non-spam})$ be the prior probability of no match (spam).
- $P(\text{skills}|\text{non-spam})$ be the likelihood of seeing the candidate's skills given that they are relevant (non-spam).
- $P(\text{skills}|\text{spam})$ be the likelihood of seeing the candidate's skills given that they are not relevant (spam).

Bayes' Theorem is used to update our beliefs about the probability of non-spam based on the observed skills:

$$P(\text{non-spam}|\text{skills}) = \frac{P(\text{skills}|\text{non-spam}) \times P(\text{non-spam})}{P(\text{skills})}$$

where $P(\text{skills}) = P(\text{skills}|\text{non-spam}) \times P(\text{non-spam}) + P(\text{skills}|\text{spam}) \times P(\text{spam})$

Thus, the matching probability (non-spam) and its complement (spam probability) are computed as:

$$P(\text{non-spam}|\text{skills}) = \frac{P(\text{skills}|\text{non-spam}) \times P(\text{non-spam})}{P(\text{skills}|\text{non-spam}) \times P(\text{non-spam}) + P(\text{skills}|\text{spam}) \times P(\text{spam})} \tag{1}$$

On the other hand

$$P(\text{spam}|\text{skills}) = 1 - P(\text{non-spam}|\text{skills}) \tag{2}$$

The key to this method, as with any Bayesian approach, is to continuously update the posterior until we reach plausible results. This is achieved by updating the data on spam and non-spam categories. Suppose we start with:

```
spam_non_spam_data =
{
        'python': (0.5, 0.5),
        'java': (0.5, 0.5),
}
```

after some iterations, we get up grade it to:

```
spam_non_spam_data =
{
        'python': (0.9, 0.1),
        'java': (0.85, 0.15),
        'cooking': (0.01, 0.99),
        'cleaning': (0.01, 0.99)
}
```

Let us look at this example, given the job description and candidate's skills, we compute the likelihoods of the skills being non-spam (relevant) and spam (irrelevant) using Bayesian analysis. Here are the detailed steps and results:

```
job_description_example = "Looking for an experienced
    ↪  Data Engineer with strong skills in Python,
    ↪ SQL, and data pipelines."
candidate_skills_example = "Experienced data engineer
    ↪  with expertise in Python, Java, SQL, and
    ↪ Hadoop."
```

### Prior Probabilities

$$P(\text{non-spam}) = 0.5$$
$$P(\text{spam}) = 0.5$$

### Likelihoods

Assuming neutral probabilities for unlisted skills:

$$\begin{aligned} P(\text{Python}|\text{non-spam}) &= 0.9 \\ P(\text{Python}|\text{spam}) &= 0.1 \\ P(\text{Java}|\text{non-spam}) &= 0.85 \\ P(\text{Java}|\text{spam}) &= 0.15 \\ P(\text{SQL, Data Pipelines}|\text{non-spam}) &= 0.5 \\ P(\text{SQL, Data Pipelines}|\text{spam}) &= 0.5 \end{aligned}$$

### Combined Likelihoods

$$\begin{aligned} P(\text{skills}|\text{non-spam}) &= 0.9 \times 0.85 \times 0.5 \times 0.5 = 0.19125 \\ P(\text{skills}|\text{spam}) &= 0.1 \times 0.15 \times 0.5 \times 0.5 = 0.00375 \end{aligned}$$

### Posterior Probabilities

Using Bayes' Theorem:

$$P(\text{non-spam}|\text{skills}) = \frac{0.19125 \times 0.5}{0.19125 \times 0.5 + 0.00375 \times 0.5} = 0.9808$$
$$P(\text{spam}|\text{skills}) = 1 - 0.9808 = 0.0192$$

### Unknown words

One challenge with Bayesian matching is the issue of 'unknown' skills:skills not present in the training set. For numerical reasons, we decide to assign neutral probabilities, that is, a 50% chance of a match before encountering the skills, updating our belief only once we observe the skills. Due to our limited training samples, there are some unknown skills not covered in the test set, which yield a result of 0.5. This indicates uncertainty about whether it is spam or not spam. Fortunately, only two matches return a probability of 0.5, giving us confidence that expanding the dataset will eventually encompass these unknown skills.

## 4 Results

With 500 matching examples divided in a 90/10 split for training and testing, we have already achieved significantly good results with the Bayesian spam filter.

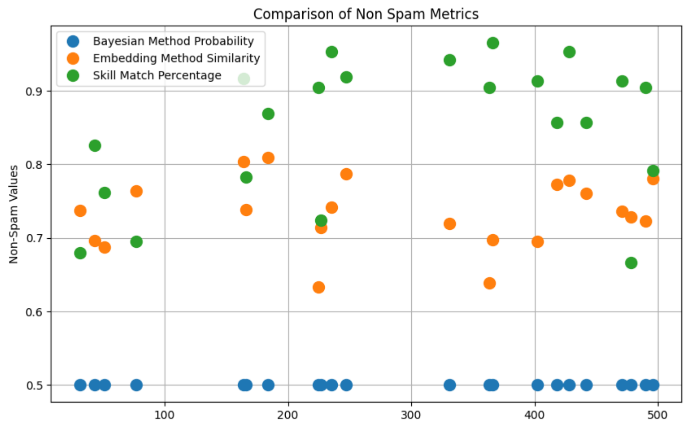

Figure 4: Comparison of metrics including Spam Probability, Cosine Similarity, and Skill Match Percentage: Beginning of the Iterations.

At the beginning of the iteration, as shown in Figure 4, without any priors, both embedding and keyword matching significantly outperform the Bayesian method. The skill matching and embedding values are higher for every single testing data point, whereas the Bayesian method provides uncertain answers. However, within just a few iterations, we observe a noticeable improvement in the probability of identifying the correct matches between datasets, as illustrated in Figure 5.

By the end of our 40th iteration, the Bayesian method surpasses the other methods by a significant margin in every test dataset except for two, as shown in Figure 6. Some results are 100% confident, which never occurred in any other methods, that they represent a good non-spam match.

With modern memory hardware, the computation takes only seconds, and memory usage is limited to the training

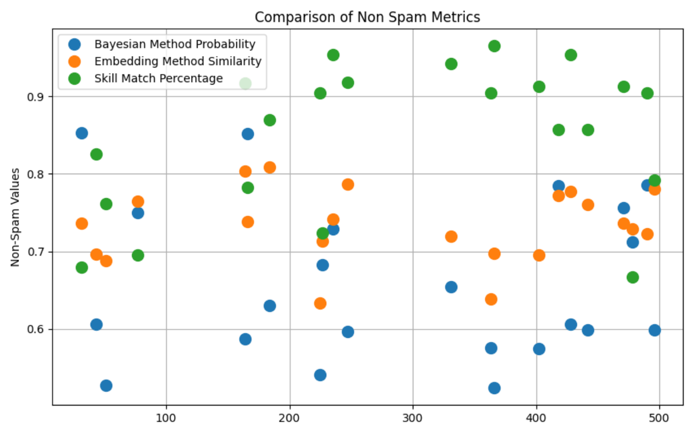

Figure 5: Comparison of metrics including Spam Probability, Cosine Similarity, and Skill Match Percentage:Middle of the Iterations.

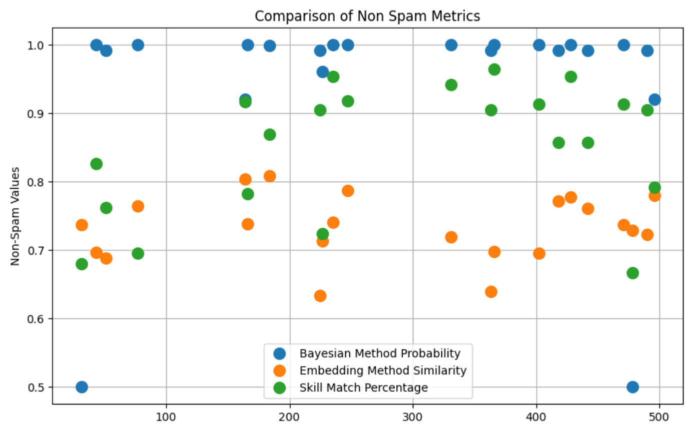

Figure 6: Comparison of metrics including Spam Probability, Cosine Similarity, and Skill Match Percentage: End of the Iteration.

data and basic computing floats. This highlights another advantage of the Bayesian method compared to embedding: it achieves better results with faster computing without causing memory issues associated with downloading large pre-trained model libraries. Although skill set matching with regular expressions is faster and less memory-intensive, the results are not nearly as accurate as those obtained with the Bayesian method.
This concludes our journey from worrying about spamming to loving the Bayes theory, we would also recommend that the company should stop worrying about spamming and start applying Bayes' Theory before diving into the latest and greatest machine learning methods that are popular on the internet and often confuse crowds.

## 5 Conclusion

In this project, we addressed the spamming problem faced by Dice.com and identified the necessary data to tackle the issue. One section was dedicated to data cleaning, emphasizing that our model's effectiveness depends on the quality of data provided. We queried and rendered HTML data into two components: job descriptions and candidate profiles. The data was then divided in a 90/10 split, using 400 data points for training and 100 for testing the models. We proposed three solutions to the problem, with a primary focus on Bayesian matching. The other two non-Bayesian methods are embedding and keyword matching Embedding, while offering the lowest accuracy rate, is adept at handling unknown skills, though it demands significant memory due to the use of pre-trained libraries. In contrast, keyword matching is effective only when specific words are already present in the job description—as was the case in our test samples—but performs poorly when these keywords are absent. Finally, we enhanced our Bayesian method by incrementally increasing the data sample and continuously updating the posterior for spam probability. Within 40 iterations, we observed that its results surpassed those of the other models.

## 6 Future Works

Besides debugging, the biggest constraint of this project was timing: we did not have sufficient time to fully explore other potential methods and observe their outcomes. Should time permit in the future, we would like to further investigate and complete the following approaches to explore the potential of Bayesian methods more thoroughly.

### Large Data set

There is a severe constraint on our labeling data; so far, we have utilized only around 400 data points for training, with an additional 100 for testing, all from a similar domain. This limitation leads us to wonder what would happen if we were to pool all our resources together by training on all available job postings with matching candidate profiles, potentially leveling up to 1 million data points. We can only imagine that such a comprehensive dataset would cover all tech skills, thereby eliminating any 0.5 outputs due to unknown skills. However, we would assume the computing requirements would be intense as well.

### PyMC

One approach we failed to successfully implement involved using the PyMC[2] model, where we employed Bayesian techniques to calculate the probability of spam versus non-spam,

[2]https://github.com/pymc-devs/pymc

while utilizing MCMC simulations to sample from the posterior for inferences. This attempt would highlight how TensorFlow can be a powerful tool for real-world text classification.

### Knowledge Graph

In the project proposal, we introduced a third dataset highly relevant to Bayesian analysis: our in-house developed knowledge graph. This has been converted into lists such as the following:

| **Job Title** | **Related Skills** |
|---|---|
| .Net Application Developer | Microsoft technologies; Software development; C#; HTML; Quality assurance; ASP.NET; Visual Basic .NET; .NET; Agile |

We believe with introduction of relevant skills, out matching would go beyond the previous result, but as in the result section, the bayes spam filter already achieved excellent result without even considering the knowledge graph, on top of that, by building a skill-set connector as shown in Figure 7, we can connect the skill with other candidate's skill based on conditional probability, to better predict spam outcome.

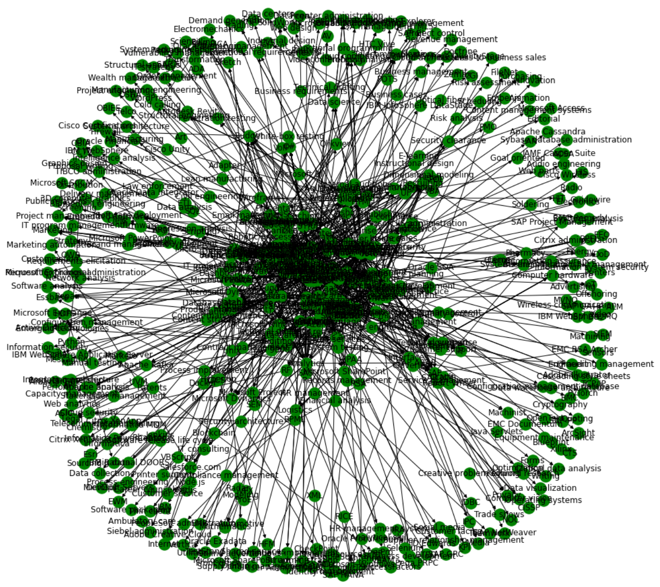
Figure 7: Overall connection of skillsets.

### Customized Bayes Spam filter

Another goal we were unable to achieve was building a customized spam filter for each candidate, e.g.: software developer vs data engineer, as outlined below:

```
customized_spam_non_spam_data =
'cadidate title': 'software developer',
'spam_non_spam_prob':
{
        'python': (0.9, 0.1),
        'java': (0.85, 0.15),
        'cooking': (0.01, 0.99),
        'cleaning': (0.01, 0.99),
'cadidate title': 'data engineer',
'spam_non_spam_prob':
{
        'python': (0.9, 0.1),
        'pyspark': (0.99, 0.01),
        'cooking': (0.01, 0.99),
        'cleaning': (0.01, 0.99)
        }
```

This was not only due to the lack of labeled data for each candidate, but also because there are hardly any job titles that we can associate with the skill sets, even with the help of the knowledge graph. In the future, we would like to prioritize exploring this solution further, using real-life labeled data.

## 7 Acknowledgment

We acknowledge and express our gratitude to DHI Group Inc. for allowing us to use their data for this project. The data provided remains the property of DHI Group Inc., and we are thankful for the opportunity to utilize such valuable resources to enhance our research and findings.

## References

[1] Brian J. Reich (2019), Bayesian Statistical Methods (Chapman & Hall/CRC Texts in Statistical Science) 1st Edition

[2] Andrew Gelman (2013), Bayesian Data Analysis (Chapman & Hall/CRC Texts in Statistical Science) 3rd Edition

[3] Naive Bayes spam filtering (wiki), `https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering`

[4] Christoph Wehner (2024), Interactive and Intelligent Root Cause Analysis in Manufacturing with Causal Bayesian Networks and Knowledge Graphs

[5] Ilya Sutskever (2013), Distributed Representations of Words and Phrases and their Compositionality

[6] Tomas Mikolov, (2013), Efficient Estimation of Word Representations in Vector Space