MARKET-BASKET ANALYSIS FOR JOB SKILLS USING LINKEDIN JOBS AND SKILLS DATASET

DARYA SAVITSKAYA

Contents

1. Introduction

The LinkedIn jobs and skills dataset, with over 1.29 million job postings, gives the possibility to gain insights into the skills needed for different job roles. Each entry has a job description and a list of skills, where the job_skills column is especially important. The goal of this project is to use market-basket analysis to find skill combinations that often appear together in job postings. I will treat each list of skills as a "basket" and each skill as an "item." By using frequent item-set discovery techniques I hope to identify groups of skills that are commonly requested together. This analysis could help better understand what skills are in demand and how they connect across different job roles, this approach could be used as a base for the job posting suggestion system.

2. Data Preprocessing

The dataset initially contained 1297332 job postings, i.e. two columns a link and a summary. After removing the null rows, the figure ended up being 1294346. After some inspection I have discovered that some skills are repeated with different levels of capitalisation of letters, for example in the dataset we have a skill 'Customer service' mentioned 166209 times, (the third most common skill in the dataset) and another one 'Customer Service' mentioned 110418 times. Obviously without removal this would result in the same skills considered as two different frequent skills. Similarly we have 'Time management' - 72481 times and 'Time Management' - 69769.

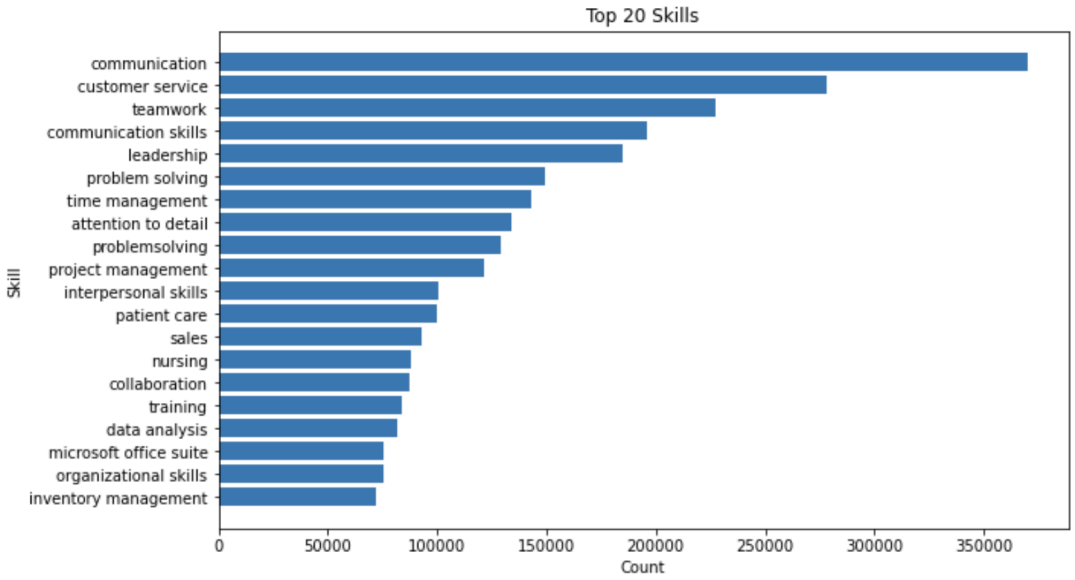

Figure 1. The most popular skills

In fact, the distribution of frequencies of the skills is highly positively skewed with mean equal to roughly nine and median to one. This means that a small number of skills appear very frequently, while the vast majority appear very infrequently - often just once. Given that 50% of the skills occur only once (median = 1), a support value much higher than 1 will drastically reduce the number of skills considered frequent. However, a support equal to 1 doesn't make much sense. We end up with the fact that half of the skills do not carry significant information for the frequent itemsets algorithm. In Figure 2 you can see the log-scaled distribution of the frequencies as proof.

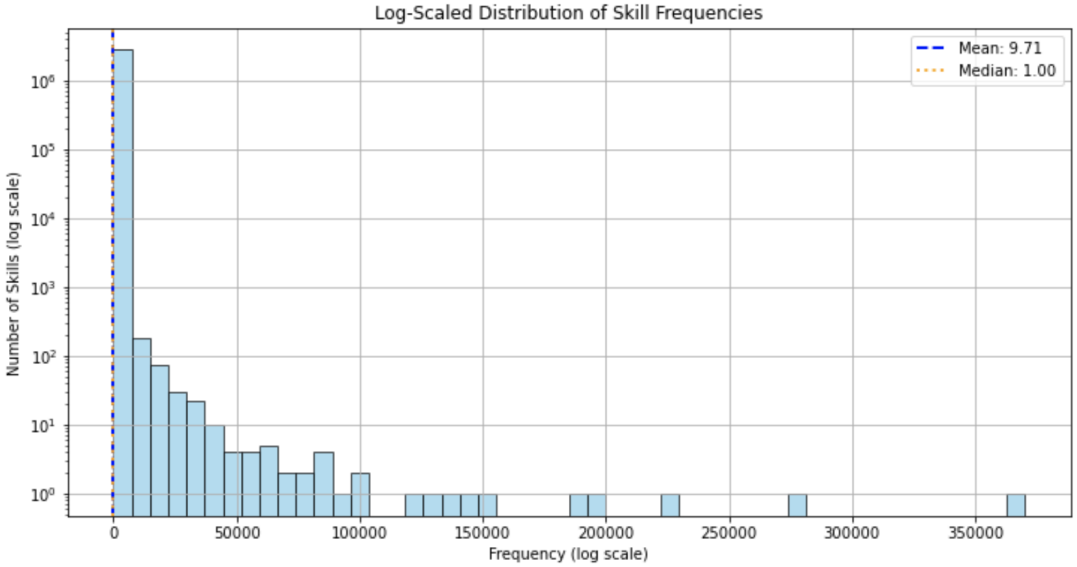

FIGURE 2. Skills distribution

3. Methodology: Algorithms and Implementation

3.1. **Algorithms.** In my analysis of job skills, the "basket" refers to the set of skills associated with a specific job listing. It's not literally a physical basket, but rather the data that represents which skills are required together for a particular job. These baskets are represented as sets of skills, with each skill being an entry in the dataset, for example "Python", "Data Analysis", "Machine Learning" is a basket.

By identifying frequently occurring sets of skills, we can reveal associations between different skills. This insight helps employers and job seekers understand which skills are commonly needed together, enabling better targeting of job ads, training programs, or career development strategies.

3.1.1. *A-priori algorithm.* The Apriori algorithm is a fundamental and in a sense basis approach used to identify frequent itemsets in large datasets. For the data at hand the algorithm would follow these steps:

First iteration: Counting the frequency of individual skills (1-itemsets) in all job descriptions. Any skill that appears less frequently than chosen support is discarded. Second iterations: Generating pairs of skills (2-itemsets) and counting their occurrences in the dataset. For an itemset to be considered frequent, all its subsets must also be frequent. If a pair (like Python, Cooking) is frequent, then both Python and Cooking must have already passed the support threshold. Following iterations: Continuing with larger sets (3-, 4-, ...) until no further frequent itemsets can be found. However, Apriori requires multiple passes over the dataset, which can be computationally expensive, especially as the size of itemsets increases. Indeed, the memory usage was rather excessive with the threshold that had some logical sense.

3.1.2. *PCY Algorithm.* The PCY algorithm is an enhancement of Apriori designed to reduce memory usage and computation time, making it more efficient for large datasets.

First iteration: Like Apriori, PCy counts the frequency of individual skills (1-item-sets). However, during this pass, it also uses a hash function to place pairs of skills into "buckets". These buckets are then counted to see how many skill pairs fall into each bucket. Bitmap Creation: After the first pass, a bitmap is created where each bit represents whether a bucket is frequent - exceeds the support. Second iteration: now counting only pairs of skills that hash to frequent buckets. Following iterations: Using the bitmap to filter out many non-frequent pairs early on, saving time and memory, counting with larger sets until no further frequent item-sets can be found.

At the end, the decision was made to employ PCY algorithm, mainly due to the memory efficiency. By using a hash-based approach to filter out non-frequent pairs early on, I have significantly reduced the number of candidate pairs that need to be counted in later passes. Similarly, the approach becomes more scalable in general. By using a bitmap and hashing, the PCY algorithm significantly reduces the number of candidate item-sets to be considered in each subsequent iteration, leading to faster convergence. I have noticed that PCY algorith ended up being a better choice in the matter of performance issues, it identified frequent datasets with a smaller number of iterations compared to A-priori and secures a resonable timeframe allowing a better performance for various support and busket thresholds.

3.1.3. *Specification.* I used Pyspark to efficiently process the large dataset, which contains over a million entries. PySpark enables to handle this volume of data by distributing the workload across multiple processors, making market busket algorithms scalable and fast. Without Pyspark, analysing such a large dataset would be slow and would likely exceed the memory limits of a single machine.

The following is the logic of the code used for the PSY algorithm. STarting by identifying single skills that occur frequently across all job listings by flattening the skills in each job entry and counting their occurrences. Only skills appearing more than a support number are kept. Next, it uses hashing to assign pairs of skills to buckets and creates a bitmap that indicates which buckets are frequent. The algorithm then iteratively searches for larger combinations of skills, starting with pairs, and filters these combinations using the previously generated bitmap and the frequent skills found in the previous iteration. Process continues, increasing the size of the item-sets in each iteration until no more frequent combinations are found. The use of Pyspark allows this analysis to be performed efficiently on large datasets by distributing the computation across multiple cores.

4. Results

For the PCY algorithm, with a support threshold of 1200 and 2000000 buckets, the results show that in the first iteration, there are 2200 frequent skills identified - these skills appear at least 1200 times across all job listings. For example, skills like 'waste management', 'attention to detail', and 'high school diploma' are among the most frequent, with counts of 1574, 133929, and 67244, respectively.

The choice of a support = 1200 was made to balance computational efficiency and the meaningfulness of the detected patterns. A lower support threshold would have included too many infrequent skills, increasing both memory usage and processing time, while also introducing noise into the analysis. A much higher threshold would risk filtering out potentially useful sets that could reveal important associations between job skills.

By setting the support threshold to 1200, we ensure that only skills that appear frequently enough across job postings are considered. This helps focus on skills that are truly common in the job market, making the findings more relevant for practical applications like job recommendations or market analysis. Additionally, this threshold was chosen based on an analysis of the skill distribution; it captures a sufficient number of frequent skills, 2200 in the first iteration, without overwhelming the system, allowing the algorithm to efficiently handle the dataset size while still extracting valuable insights. For calculation I used as a basis the number of the job postings - 1297332, i.e. the support is equal to around 0.009% of the whole dataset. Obviously very precise, the figure could've been made higher for even faster performance. However, the execution of the process by the PCY algorithm with such big data volume and small support says a lot about the scalability of this particular approach.

The bitmap has a size of 2000000, but only 11158 buckets are flagged, meaning that only a small fraction of the hash space is utilised to store frequent sets. This means that the algorithm is working well to compress the data while filtering out less frequent combinations.

To conclude the results section, this selection of support and bucket parameters demonstrates the effectiveness of the algorithm in handling large-scale datasets, focusing on an optimal balance between performance and the extraction of meaningful and actionable insights.

5. Conclusion

This project lays the basis for a scalable system capable of efficiently identifying frequently co-occurring skills in large datasets. By leveraging the PCY algorithm, we can optimise memory usage and performance while detecting patterns of skills that appear together in job descriptions. A model like this can be particularly beneficial for job platforms, where understanding common skill combinations can enhance recommendations for job postings or improve the match between candidates and job requirements. The insights gained can also support targeted training programs or skill development initiatives. Further additions, such as experimenting with different support thresholds or increasing the number of buckets, could better the accuracy and scalability of this system, making it a valuable tool for both job seekers and recruiters.

6. Appendix

References

[1] Francesco Bertolotti, *Market Basket Analysis*, `https://github.com/f14-bertolotti/labs/blob/main/DSE/MarketBasket/MarketBasket.ipynb`. Accessed: 2024-11-15.

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work, and including any code produced using generative AI systems. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.