

---

# Finding similarities between software development skills

---

Master of Science in Technology  
Thesis  
University of Turku  
Department of Computing  
Software Engineering  
2023  
Kristian Koskinen

UNIVERSITY OF TURKU  
Department of Computing

KRISTIAN KOSKINEN: Finding similarities between software development skills

Master of Science in Technology Thesis, 57 p., 11 app. p.  
Software Engineering  
April 2023

---

As software development becomes an increasingly important part of modern society, understanding the similarities and differences between different software development skills is critical for improving the effectiveness and efficiency of software development teams.

In this thesis we explore the use of user skill level data to identify similarities in software development skills. Using data from an application that a company uses to gather information about the competence of their personnel. In this thesis we try to explore the possibility of finding similar skills and measuring this similarity.

The initial results from this data do not give clear clusters of similar skills however further processing of this data gives results that hold great potential. By providing a measure of similarity between different skills we can better match candidates to job requirements and training programs improving the overall effectiveness and efficiency of software development teams.

Keywords: semantic search, similarity, software development

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                                 | <b>1</b>  |
| <b>2</b> | <b>Semantic search fundamentals</b>                 | <b>4</b>  |
| 2.1      | Machine learning . . . . .                          | 4         |
| 2.2      | Supervised learning . . . . .                       | 5         |
| 2.3      | Unsupervised learning . . . . .                     | 7         |
| 2.3.1    | Clustering . . . . .                                | 7         |
| 2.3.2    | Clustering evaluation . . . . .                     | 9         |
| 2.3.3    | K-means algorithm . . . . .                         | 9         |
| 2.4      | Semantics and semantic search . . . . .             | 10        |
| <b>3</b> | <b>Search engine and Elasticsearch fundamentals</b> | <b>12</b> |
| 3.1      | Brief history of search engines . . . . .           | 12        |
| 3.2      | Elasticsearch . . . . .                             | 13        |
| 3.3      | Searching in Elasticsearch . . . . .                | 16        |
| <b>4</b> | <b>Case Competence management system</b>            | <b>19</b> |
| 4.1      | About the company . . . . .                         | 19        |
| 4.2      | Demand for semantic search and use cases . . . . .  | 19        |
| 4.2.1    | Other solutions . . . . .                           | 21        |
| 4.3      | Examples of the data . . . . .                      | 21        |

|          |   |            |
|----------|---|------------|
| <b>5</b> | <b>Processing the data</b>                              | <b>26</b>  |
| 5.1      | Preprocessing . . . . .                                 | 26         |
| 5.1.1    | Formatting . . . . .                                    | 26         |
| 5.1.2    | Validating . . . . .                                    | 27         |
| 5.1.3    | Separating skills . . . . .                             | 27         |
| 5.2      | Processing . . . . .                                    | 29         |
| 5.2.1    | First clustering . . . . .                              | 30         |
| 5.2.2    | Standardizing and normalizing the data . . . . .        | 32         |
| 5.2.3    | Using the Principal Component Analysis . . . . .        | 34         |
| 5.2.4    | Conclusion from clustering . . . . .                    | 36         |
| 5.2.5    | Using dot product to create similarity matrix . . . . . | 37         |
| 5.2.6    | Cosine similarity . . . . .                             | 40         |
| 5.2.7    | Presence of unnamed skills . . . . .                    | 40         |
| 5.3      | Conclusion from processing . . . . .                    | 42         |
| <b>6</b> | <b>Validation</b>                                       | <b>43</b>  |
| 6.1      | Evaluating with experiments/interviews . . . . .        | 43         |
| 6.2      | Validation methods . . . . .                            | 44         |
| 6.3      | Interview materials . . . . .                           | 45         |
| 6.4      | Results from the interview . . . . .                    | 50         |
| <b>7</b> | <b>Conclusion</b>                                       | <b>54</b>  |
| 7.1      | Answers to the research question . . . . .              | 54         |
| 7.2      | Discussion . . . . .                                    | 55         |
|          | <b>References</b>                                       | <b>58</b>  |
|          | <b>Appendices</b>                                       |            |
| <b>A</b> | <b>Code</b>   | <b>A-1</b> |



# List of Figures

|      |  |    |
|------|--|----|
| 2.1  | Local minimum [3]  | 6  |
| 2.2  | Different clustering results [6]   | 8  |
| 4.1  | Top 50 most common skills, right with skills with no name (red) included   | 24 |
| 4.2  | Graph about skill occurrences  | 25 |
| 5.1  | Histogram on skill distribution (x-axis is the amount of skills per user and y-axis is the number of users with this number of skills) | 29 |
| 5.2  | Histogram on level distribution (x-axis is the skill level and the y-axis is the number of skills with that level)                     | 30 |
| 5.3  | Histogram on interest distribution (x-axis is the user interest in skill and the y-axis is the number of skills with that interest)    | 30 |
| 5.4  | Cluster size with default data   | 32 |
| 5.5  | Cluster size with normalized data  | 33 |
| 5.6  | Cluster size with standardized data  | 33 |
| 5.7  | Cluster size with normalized and standardized data   | 34 |
| 5.8  | Boxplot from different PCA dimension count with percentage of variance explained by each of the selected components                    | 35 |
| 5.9  | Cluster size with using PCA  | 36 |
| 5.10 | Visualizing PCA result   | 36 |
| 5.11 | Dot product ranking normal data and results  | 37 |

|      |   |    |
|------|---|----|
| 5.12 | Dot product ranking normal and standardized results . . . . .   | 38 |
| 5.13 | Dot product ranking standardized data and normal results . . . . .  | 38 |
| 5.14 | Dot product ranking standardized data and results . . . . .   | 39 |
| 5.15 | Dot product ranking with python . . . . .   | 39 |
| 5.16 | Cosine similarity with JavaScript . . . . .   | 41 |
| 5.17 | Cosine similarity with an unnamed skill . . . . .   | 42 |
| 6.1  | Interview materials showing cluster size and sample from first three<br>clusters on what skills they contain. . . . . | 46 |
| 6.2  | Interview materials showing two lists from the similarity matrix . . .  | 47 |
| 6.3  | Interview materials showing four lists with different scaling . . . . .   | 48 |
| 6.4  | Interview materials two list to compare the order . . . . .   | 48 |
| 6.5  | Interview material showing cosine similarity with "AWS" . . . . .   | 49 |
| 6.6  | Interview material showing cosine similarity with a skill with no name  | 50 |

# List of Tables

|     |   |    |
|-----|---|----|
| 3.1 | Example of forward index . . . . .              | 17 |
| 3.2 | Example of inverted index . . . . .             | 17 |
| 4.1 | Example of unprocessed skill data . . . . .     | 21 |
| 4.2 | Example of unprocessed userskill data . . . . . | 22 |
| 5.1 | Example of processed data . . . . .             | 28 |
| 5.2 | Example of data used in clustering . . . . .    | 28 |



# 1 Introduction

We are all familiar with keyword search which is often used by search applications in our own computer and throughout the internet. In keyword searching, you enter some text, and the search engine returns documents containing the text that you entered. These results are then ranked, based on different criteria.

Although this type of keyword search generally does a good job in ranking web pages, most of us know that this kind of search completely fails in other contexts. For example, searching your own computer for a document by relying on keywords can be very frustrating. Not to mention searching a data store the size of your corporate intranet.

Rather than blindly returning anything that contains the text you typed into the search bar, semantic search takes into account the context of your search as well as the underlying meaning of the documents to be searched. For example, searching with the keyword java, it might mean the programming language, the island, or a drink. There are multiple different meanings for one word, so it is quite hard to know what the user is actually trying to achieve with just one word. If you are using that particular search engine for the first time, usually it just ranks them based on popularity. Later the search engine can learn more about the user from its search history and give more relevant answers based on that. If those are not suitable then users can just add descriptions like “java island” if they meant the island.

Semantic search denotes search with meaning. It seeks to improve search ac-

curacy by understanding the intent and the contextual meaning of the search by ranking the results based on similarity. When it comes to software development there are a huge amount of different skills. All the tools, languages, methods of programming and managing the development are just a few to mention. Given the proper background knowledge we can tell which skills are similar. But it is impossible for one person to know all the skills in the world and rank them against each other based on similarity.

The application in this thesis has data on the user's own measurement of their skill level as well as interest on that specific skill in a 0-5 scale. For example, if person is interested in Java, maybe they would also like C#, because many other C# users are interested or skilled in similar skills.

**RQ1: What is the best method to process data of user skill levels to achieve the best possible result in similarity?**

Measuring this similarity can be very difficult. Does there exist any standardized metrics currently that can be used? Or do we just trust on basic instinct on the similarity between the skills?

**RQ2: How can we measure similarity between skills in software development?**

At the moment, the application users do not have a specific process to find suitable candidates for new projects. They just search with terms they think are relevant. Currently the application uses Elasticsearch, and one of the applications of this similarity data would be to integrate semantic search to this using a similarity score found by the model for the similarities.

In this thesis we are tackling a problem that includes creating a semantic search for different expertise in the software development business, like what programming languages one has experience in or what kind of development methodologies are used. What if some people only list programming languages they have used and then

---

someone would like to find a person with experience in “web technologies”? Using just a regular search would not show people who have not added web technologies into their expertise, even though they have multiple years of experience in the field. This is the problem that this thesis is trying to solve. If this kind of similarity can be found using this type of data, it should be easy to scale this into much larger data set to get even more precise results and add more skills.

The Chapter 2 goes through the necessary background information required to understand what semantic search is and how it works. It also goes through some other relevant information regarding the processing of the data and methods used in it. The Chapter 3 takes a look at the current situation in search engines and a deeper look into specific search engine used by the application in this thesis. The Chapter 4 tells about the company and their need for the application. It explains how the current application is used and why does the company have this need for this new approach. The Chapter 5 goes through different methodologies used in this thesis and tries to argument why they are used. It also goes through the results and tries to reflect on those whether they succeeded or not. The Chapter 6 goes through the validation of the results from the processing. And then finally the conclusions.

## 2 Semantic search fundamentals

As stated before semantic search denotes search with a meaning. But where does this meaning come from? How can a machine know the meaning behind some text that is given to the search engine? There are different ways of constructing the logic behind the semantic search. Usually they use artificial intelligence and machine learning techniques like natural language processing or clustering to achieve this kind of similarity score.

### 2.1 Machine learning

Machine learning is a subject that studies how to use computers to simulate human learning activities, and to study self-improvement methods of computers that to obtain new knowledge and new skills, identify existing knowledge, and continuously improve the performance and achievement.[1]

If the machine is given one simple task then the learning can also be done easily and the machine learns quickly. But if one increases the complexity of tasks and/or increases the amount of tasks this learning process becomes increasingly more difficult.

The most important part of machine learning is the data the machine is used to train and test with. When it comes to the amount of data one obviously wants to use all of the available data. But the trick is to use the least amount of possible data in order to achieve the result wanted. This is because the more data one has

to teach its machine the more confident one can be with the results. But this comes with a cost of time and complexity.

The result of machine learning is called a *model*. Model is trained with the given data and its performance can be measured during and after the training. There are ever increasing ways to perform this learning on the model. Some ways of learning are better suited for specific tasks and the best way to teach the model should always be chosen given the data and the task at hand.

As Wang et. al stated: "Generally speaking, the machine learning algorithm comes down to solving the optimization problem" [2]. Getting a number to represent this optimization problem usually happens with *loss function* (or sometimes called cost or error function). This function maps values of one or more variables into a number that represents the "cost" of that specific event. Job for the machine learning algorithm is to minimize this cost. But depending on where the initial position is the algorithm may end up in a local minimum instead.[2]

The figure from Wikipedia illustrates this very well (Figure 2.1). In the figure we can see that if we start on the right side of the local maximum then the algorithm may start diverting to the right and end up in the local minimum. One solution to this problem is to have multiple runs with different initialization to have possible different results and thus reducing the risk of ending in the local minimum.

## 2.2 Supervised learning

There are many different approaches to the usage of data in machine learning. One is called *supervised learning* where the algorithm gets to use labeled data as an input. Because we already know the answer in this dataset, we can use it to supervise the algorithm into giving more accurate results. In this the data is usually divided into training and test sets. The training set is used to train the model to increase its performance. And the test set is used to verify the results and improve the

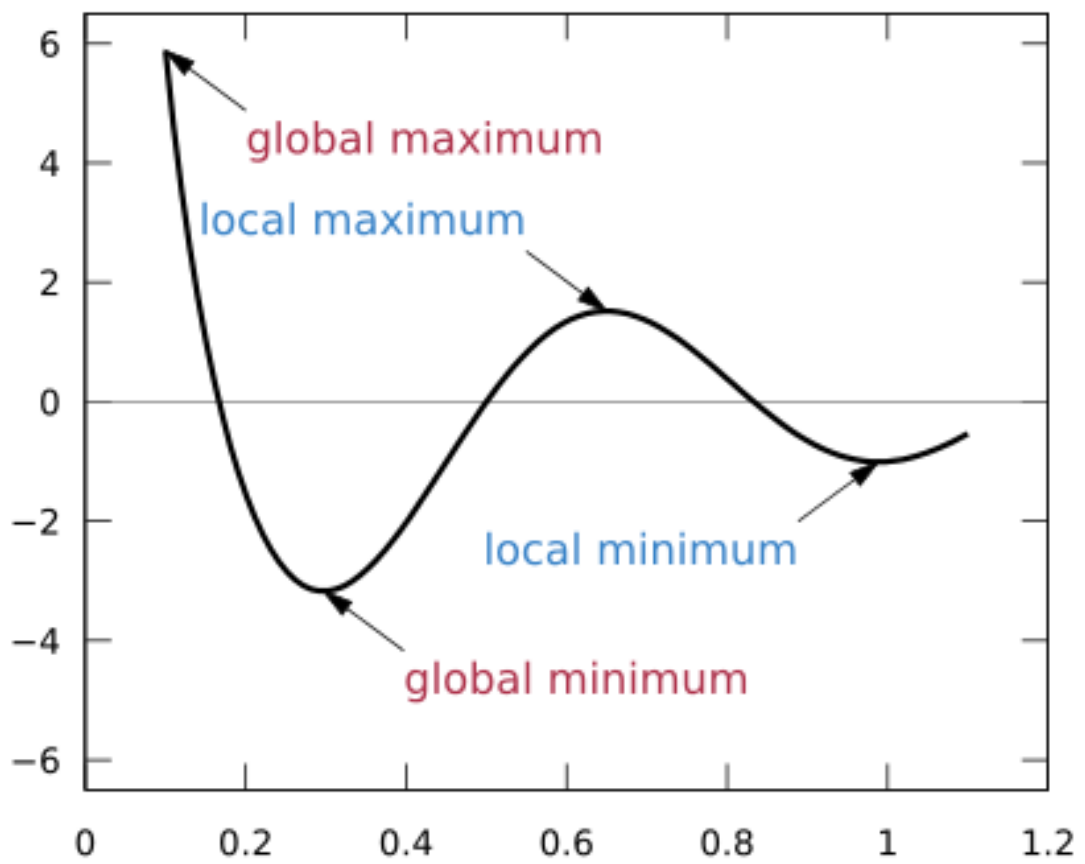


Figure 2.1: Local minimum [3]

performance of the model based on feedback. It is important to not use the same data for training and verification as this will cause unbiased results.[4]

Supervised learning can be divided into two different types: *classification* and *regression*. Classification problems use an algorithm to accurately assign test data into specific categories, such as separating apples from oranges. Or, in the real world, supervised learning algorithms can be used to classify spam in a separate folder from your inbox. Evaluating classification can be done in different ways. One way is calculating *precision* and *recall*. Recall is the proportion of real positive cases that are correctly predicted positive. And recall is the proportion of predicted positive cases that are correctly real positives. [5]

Regression is another type of supervised learning method that uses an algorithm to understand the relationship between dependent and independent variables. Regression models are helpful for predicting numerical values based on different data points. [4]

## 2.3 Unsupervised learning

Another one that we are more interested in in this thesis is called *unsupervised learning*. In this type of learning we do not have the labeled data as the model only gets inputs without labels, categories or classes. The model is not given any feedback and instead tries to find similarities/patterns, or the lack of those in the given data without human interaction. [4]

### 2.3.1 Clustering

Clustering is one of the most common types of unsupervised learning. In this type of learning the model tries to split the given input data into subsets that are called clusters. This splitting happens so that each data point in a cluster is similar with

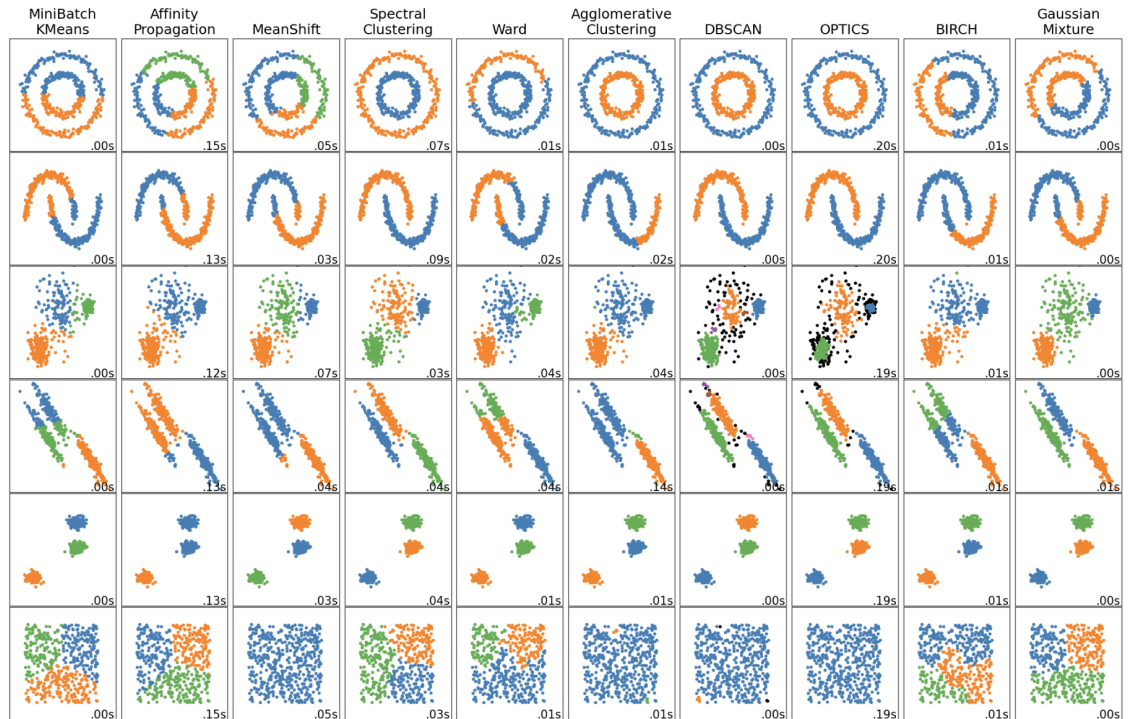


Figure 2.2: Different clustering results [6]

each other point in the cluster with one or more predefined criteria. There are multiple ways to define the criteria for the given tasks and they give very different results depending on what criteria is used. The example from the Scikit-learn website illustrates this very well (Figure 2.2). Looking at the figure one can see how different these methods perform in multiple situations. As said before, all of these come with their own pros and cons. As for an example, looking at the first row one can see how some clustering methods divide this into halves or into thirds and some can actually separate the circles into separate clusters. This same thing happens again with different datasets with these clustering techniques performing very differently.



### 2.3.2 Clustering evaluation

Evaluating the performance of a clustering algorithm is not as trivial as counting the number of errors or the *precision* and *recall* of a supervised classification algorithm.

Evaluation of clustering results can happen in different ways. The best method is usually chosen based on if one knows the true labels or not. [7]

In this thesis we are more interested in the methods where the labels are not known because we do not know for sure what labels we should use for different clusters. When the labels are unknown the evaluation must be performed on the model itself. And usually try to give a number on how similar the clusters are compared to other clusters. One of these methods is called *Silhouette Coefficient* (or silhouette score) which is designed to calculate how well defined the clusters are. The Silhouette Coefficient  $s$  of a single sample is calculated using

$$s = \frac{b - a}{\max(a, b)}$$

Where the  $a$  is the mean distance between sample  $a$  and all other points in the same class. And  $b$  is the mean distance between sample  $a$  and all other points in the next nearest cluster. Then the Silhouette Coefficient of the model is the mean of all of the samples. [7]

### 2.3.3 K-means algorithm

We are going to take a deeper dive into k-means clustering as this is going to be used in this thesis as the clustering method. K-means clustering algorithm tries to cluster data by separating data samples into a given number of samples with equal variance. As said this algorithm requires the number of clusters given as an input before the clustering. One benefit of this algorithm is that it scales very well to large number of samples and it has been widely used in multiple different applications in many different fields. K-means divides the dataset into clusters with each cluster

described with its mean of the samples in the cluster. These mean points do not necessarily belong to the original dataset and are usually called *centroids*. [8]

The algorithm itself has three steps. First is the initialization where the algorithm randomly chooses points in dataset to use as the first centroids of the clusters. The second step is when the algorithm calculates the distance between all of the points and the centroids and assigns points to the cluster. Then in the third step the algorithm calculates new mean from all of the points inside the cluster and assigns this as the new centroid. The algorithm also calculates the distance between the old and new centroid. The second and third step is repeated until the centroids no longer move over the given threshold and stable configuration is found. These are then the final clusters that are returned. [8]

Given enough time the k-means always converges but this can happen inside a local minimum This result is highly dependent on the initialization of the centroids so multiple runs are recommended for best results. One can also try to use initialization scheme called *k-means++*. K-means++ ensures that the first centroids are far apart from each other and lowering the chance of local minimum and improving the results.

## 2.4 Semantics and semantic search

Semantics is the philosophical and scientific study of meaning in natural and artificial languages. The word semantics comes from Greek verb *sēmainō* meaning "to mean" or "to signify". [9] Semantics studies the meaning of words, phrases, sentences, and other linguistic entities, not the meanings of actions or phenomena. [10]

Semantic web is an extension to current web where information is given well-defined meaning. This enables better cooperation between computers and humans. Semantic web also contains information about persons, places, events and so on, not only media on the internet that contains objects like web pages, images and videos. Even further the semantic web contains more than just hyperlink between these

objects instead it can contain multiple relations between the resources mentioned before. In short semantic search is an application of semantic web to search. [11]

# 3 Search engine and Elasticsearch fundamentals

## 3.1 Brief history of search engines

As the number of web sites increased quickly in the 90s more search engines started to appear to help people to find information they wanted quickly and more easily. A search engine is a program that when given a search query it will look up from its database for a corresponding result based on given parameters. A web search engine is a search engine that is used to search web pages. [12]

In today's life the search engines especially the web search engines have become part of our daily life. We constantly use them to look for news, funny images, recipes or find answers to questions. The amount of information available to us today is something that is very hard to visualize.

During the early development of the web, there was a list of web servers edited by Tim Berners-Lee and it was hosted on the CERN web server. As the number of pages that went online increased the central list could not keep up. [12]

The very first tool used to search the Internet was called "Archie". The name came from the word "archive". It was created by a group of computer science students at McGill University in Montreal in 1990. The program downloaded the directory listings of all the files located on public anonymous FTP (File Transfer

Protocol) sites, creating a searchable database of file names. However Archie did not index the contents of these sites and since the amount of data was so limited it could be readily searched manually. [12]

There were some new tools used to search that incrementally improved from the previous. But the next major improvement was the first full text crawler-based search engines was called WebCrawler. It came out in 1994 and unlike its predecessors, it let users search for any word in any web page, which has become the standard for all major search engines since. It was also the first one to be widely known by the public. [12]

Around 2000, Google's search engine rose to prominence. The company achieved better results for many searches with an innovation called PageRank. This iterative algorithm ranks web pages based on the number and PageRank of other web sites and pages that link there, on the premise that good or desirable pages are linked to more than others. Google also maintained a minimalist interface to its search engine. Unlike many of its competitors embedded a search engine in a web portal. [12]

## 3.2 Elasticsearch

We are going through some basic information about Elasticsearch engine, because the application already has a search application built on top of the Elasticsearch engine.

Shay Banon created the precursor for Elasticsearch called Compass to help his wife to search from her growing list of food recipes. After few iterations he wanted to create more scalable solution and renamed it Elasticsearch and made the project open source. The response from community was good and few people joined together and created company around the Elasticsearch.[13]

The main point of Elasticsearch compared to others is that it is according to

Banon: "a solution built from the ground up to be distributed and used a common interface, JSON over HTTP, suitable for programming languages other than Java as well" [14].

As Andhavarapu describes Elasticsearch: "Elasticsearch is a highly scalable open source search engine. Although it started as a text search engine, it is evolving as an analytical engine, which can support not only search but complex aggregations. Its distributed nature and ease of use makes it very easy to get started and scale as you get more data" [15].

### **Document**

Elasticsearch stores the data in the *JSON* format like most NoSQL databases. This is because JSON format is flexible and readily understandable by humans. Andhavarapu describes why JSON is used: "Reading similar information without the JSON structure would also be difficult as the information would have to be read from multiple tables. Elasticsearch allows you to store the entire JSON as it is. For a database table, the schema has to be defined before you can use the table. Elasticsearch is built to handle unstructured data and can automatically determine the data types for the fields in the document. You can index new documents or add new fields without adding or changing the schema. This process is also known as dynamic mapping." [15]

### **Index**

An index is similar to a database. The term index should not be confused with a database index, as someone familiar with traditional SQL might assume. Your data is stored in one or more indexes just like you would store it in one or more databases. The word indexing means inserting/updating the documents into an Elasticsearch index. The name of the index must be unique and typed in all lowercase letters.

[15]

### **Type**

A type can be compared to a table in a database, and an index may include one or more types. Essentially, a type serves as a means of logically separating different categories of data. It is also possible to establish relationships between different types of data. For instance, one could define a parent/child relationship between articles and comments, such that an article serves as the parent and can have one or more comments as its children. [15]

### **Cluster and node**

Elasticsearch is a distributed system, meaning it consists of one or more nodes that operate as a single application which allows it to handle larger workloads than a single server can handle. Each node contains a portion of the data and you can start with a single node and expand the cluster by adding more nodes as your data grows. The nodes work together to manage all the indexing and query requests on the data and each cluster and node has a unique name. If a name is not defined in the configuration Elasticsearch automatically assigns a unique name to each node. Elastic search also enables you to add or remove nodes on the go without interrupting the application. This process is handled seamlessly by Elasticsearch. [15]

### **Shard**

In Elasticsearch an index refers to a grouping of one or more shards. Each index's data is spread across multiple shards which enables Elasticsearch to store vast amounts of data that cannot be accommodated by a single server. To index and query data Elasticsearch employs Apache Lucene as an internal tool. Essentially a shard in Elasticsearch is an instance of Apache Lucene. [15]

### Handling documents

The primary way of interacting with Elasticsearch is with the *REST API*. The Elasticsearch itself provides JSON-based REST API over *HTTP*. Handling the documents in Elasticsearch happens with simple *CRUD* (Create Retrieve Update Delete) operations. Elasticsearch will automatically index new documents added to the system and also create the indices if they are missing. Updating the document is quite expensive as under the hood the Elasticsearch would retrieve the old document, apply changes and then insert it as a new document to the system. [15]

## 3.3 Searching in Elasticsearch

As the computation power is increasing and cost of storage is decreasing, the amount of day-to-day data we deal with is growing exponentially. But without a way to retrieve the information and to be able to query it, the information we collect does not help. [15]

Information retrieval systems are very important to make sense of the data. Imagine how hard it would be to find some information on the Internet without Google or other search engines out there. Information is not knowledge without information retrieval systems.

When it comes to text search engines in *forward index* each word in the document is stored in the index of the document. In *inverted index* instead each word contains the information on what documents contain that particular word. [15]

Using the same example as Andhavarapu to explain the forward and inverted index. Imagine if we have three documents each containing a quote from Yoda from Star Wars. These quotes are "Fear leads to anger", "Anger leads to hate" and "Hate leads to suffering". Forward index for these types of documents would be created like this 3.1. Using the same documents to create an inverted index would create



the following index 3.2.

| Document   | Terms                   |
|------------|-------------------------|
| Document 1 | fear,leads,to,anger     |
| Document 2 | anger,leads,to,hate     |
| Document 3 | hate,leads,to,suffering |

Table 3.1: Example of forward index

| Term      | Document |
|-----------|----------|
| fear      | 1        |
| leads     | 1,2,3    |
| to        | 1,2,3    |
| anger     | 1,2      |
| hate      | 2,3      |
| suffering | 3        |

Table 3.2: Example of inverted index

Understanding how the index works in Elasticsearch is important as it is one of the main steps in configuring the search engine. By default, Elasticsearch indexes all data in every field and each indexed field has a dedicated, optimized data structure. Elasticsearch also has the ability to be schema-less, which means that documents can be indexed without explicitly specifying how to handle each of the different fields that might occur in a document. When dynamic mapping is enabled, Elasticsearch automatically detects and adds new fields to the index. This default behavior makes it easy to index and explore your data. [16]

Elasticsearch is much more than just an index-based search engine. It can also support complex queries with vector search or even embedding a machine learning model directly to determine the ranking of the results.

Elasticsearch describes vector search as: "Vector search transforms text, audio, and images into numeric representations and leverages deep learning and machine learning to interpret the meaning, intent, and context of these representations to serve up much more relevant search results." [17]

With vectors we represent word in a numeric vector. This vector tries to capture the semantic properties of the word. These vectors are ranging from dense with 10 to 1000 dimensions to sparse with over 50 000 dimensions. Comparing the semantic properties, or similarity, between these vectors is usually done with *Cosine similarity*. [18]

Cosine similarity is a measure of similarity between two sequences of numbers, in this case, vectors. The cosine similarity is defined as the cosine angle between the vectors in an inner product space. It is calculated using the dot product of the vectors divided by the product of the lengths of the vectors. The cosine similarity will always be between -1 and 1. Two proportional vectors have a cosine similarity of 1, orthogonal vectors have 0 and two opposite vectors have similarity of -1. [19]

# 4 Case Competence management system

## 4.1 About the company

There are two companies involved in this thesis, the company X that develops this application and the company Y actually uses and has the data that is used in this thesis. The company Y is quite large and has many employees with multiple different skills and interests. Their main income is to sell the expertise of these persons as consultants so it is very important for them to know what everyone wants and can do.

Currently the application that is the target of this thesis is not publicly available and is only used internally by the company Y. The company Y uses this application to allow its employees to fill out their profiles containing information for example about their skills, interests, education and previous projects. Skills that the user fills in their profile are ranked on 0-5 scale on how good the user thinks they are with this specific skill. We are focusing on those skills in this thesis.

## 4.2 Demand for semantic search and use cases

The demand for this semantic search feature that allows user to search candidates for new projects more easily comes from the company X that wants to improve the

product. We looked different approaches and this semantic search feature was the most interesting and we thought this would bring the most value of all of the other proposals.

In the application when a new project is starting and it is known what types of technologies would be used in the new project, then the person in charge of finding people to the new project uses this application to find people suitable for this project based on their skills and vacancy. Of course the exact match for all the requirements is hard to find so usually the person finding these people just takes one skill at the time and tries to find a suitable person for the job.

Having a large number of people with different skill levels it has become increasingly challenging to identify the best employees for new projects. With a large and diverse workforce, it can be difficult to determine which employees have the skills and experience needed to excel in a new project.

The intended use case of the new search would be that the person searching people from the application could just list all of the skills required for the new project and the application would then return best possible matches based on similarity and vacancy. Of course exact match would be ideal, but in real world these rarely happen if the list of required skills is long and contain multiple different skills from different categories. Usually the new people to the project are not just a person, but multiple persons that would form the team that would then fulfill the requirements for the new project. But using this search currently happens only for a single position in the team, for example back-end developer and skill needed for that. Of course if the new project also has a front-end developer then obviously those skills are not needed from the back-end developer.

### 4.2.1 Other solutions

I was unable to find other solutions that would fit in the same category which is creating this similarity data based on peoples own perception of skill levels. Papers that came somewhat close were mostly comparing programming languages based on their syntax or other methods using the text needed to write the code. I was looking something that would result in some kind of similarity data that could be used with search engine. This result would then be used in the application to help these people find more easily the people they need for their new projects.

## 4.3 Examples of the data

The data in its unprocessed form had two different tables. One containing the skills and the other skills of the users. The following tables are just examples of given data and do not contain any information about the contents of the real data. All of the ids are simplified for readability but are actually using the 16-octet version of UUID in the real data. The simplest of the tables contains the information about the skills (Table 4.1). It only contains the id of the skill and its given name. This table does not contain any other information. I think that this table does not require any further processing to be able to use it. We are only interested in this data because it would be very beneficial to know what skills we are working with as all of the actual processing later will only be done with the ids so knowing what skill is what would be nice. In the example there is the id and the display name of the skill.

| id   | name    |
|------|---------|
| 321  | Angular |
| 332  | NodeJs  |
| 9019 | React   |

Table 4.1: Example of unprocessed skill data

The next table contains the actual data that we are interested in. This table contains the skills of all of the users (Table 4.2). Each row in this table contains one skill of a single user. Starting from the `id` field of the data, this actually is the id of the user. Then we have the `skillId` which is the id of the skill. Next `skillInterest` which is users own interpretation of their interest level on that skill on a scale of 0 to 100. The next field is `expInYears` which is just as the name suggests the experience of the user in years with that particular skill. The next field is the most interesting for us, the `skillLevel`, this is users own interpretation of their skill level on a 1-5 scale. Note that in this data there are bigger numbers than 5, this is because there was an older system that used scale of 0-100. The new system maps the new values to the old scale. Lastly we have the `category` field. This is users own input on how they want to categorise these skills in their own profile. I believe that this is not usable for the clustering later as it can contain anything and would not give enough value for the result to be included.

| id | skillId | skillInterest | expInYears | skillLevel | category           |
|----|---------|---------------|------------|------------|--------------------|
| 12 | 321     | 100           | 5          | 64         | Front End          |
| 12 | 332     | 100           | 2          | 14         | Back-end / Backend |
| 23 | 9019    | 80            | 4          | 100        | JavaScript         |

Table 4.2: Example of unprocessed userskill data

The Figure 4.1 shows us a list of the 50 most common skills found in the information we have. It gives us a quick look at the most frequent skills that are present. There are 2 different lists of skills. One is without the unnamed skills and the other with the unnamed skills. The unnamed skills are just skills that do not have a name attached to them. These are explained later in Chapter 5 that goes through the processing and handling of these skills.

Each skill is followed with a number that represents the count of users that have

that particular skill in their profile. Given that the dataset contains almost 1000 different users that means that half of them know JavaScript, which is a quite large portion.

We also have a graph that shows how common each skill is in the dataset. The Y-axis is the times that skills appeared in the dataset. And the X-axis is the ranking of that skill based on the amount of times it appears in the dataset. We can clearly see how quickly they drop to only single digit appearances. This is something that gives me worries about the possibilities of the data I am using. If most of these skills only appear once or twice in the dataset, they only give little to no value when trying to cluster based on similarity. I am hoping I can prove myself wrong in the next chapter where I go through the processing of the data and get some initial results.

|                                  |                          |
|----------------------------------|--------------------------|
| javascript 492                   | javascript 492           |
| git 445                          | git 445                  |
| java 436                         | java 436                 |
| python 363                       | python 363               |
| css 344                          | css 344                  |
| scrum 264                        | unnamedskill-8 299       |
| mysql 262                        | scrum 264                |
| xml 259                          | mysql 262                |
| jenkins 259                      | xml 259                  |
| c++ 254                          | jenkins 259              |
| x html 247                       | c++ 254                  |
| docker 247                       | x html 247               |
| postgresql 245                   | docker 247               |
| mariadb 244                      | postgresql 245           |
| node.js 240                      | mariadb 244              |
| react.js 234                     | node.js 240              |
| sass 220                         | react.js 234             |
| scss 218                         | sass 220                 |
| amazon web services 216          | scss 218                 |
| aws 216                          | amazon web services 216  |
| linux 208                        | aws 216                  |
| svn 208                          | linux 208                |
| sql 207                          | svn 208                  |
| microsoft .net c# 199            | sql 207                  |
| subversion 195                   | unnamedskill-85 201      |
| typescript 193                   | microsoft .net c# 199    |
| jquery 190                       | subversion 195           |
| mongodb 189                      | typescript 193           |
| microsoft sql server 186         | unnamedskill-3 192       |
| atlassian jira 179               | jquery 190               |
| windows 166                      | mongodb 189              |
| unix 163                         | microsoft sql server 186 |
| php 161                          | unnamedskill-57 180      |
| azure 159                        | atlassian jira 179       |
| angularjs 159                    | windows 166              |
| robot framework 158              | unix 163                 |
| c 156                            | php 161                  |
| agile 154                        | azure 159                |
| spring framework 146             | angularjs 159            |
| oracle 133                       | robot framework 158      |
| bash 119                         | unnamedskill-24 157      |
| kanban 118                       | c 156                    |
| microsoft visual studio 117      | agile 154                |
| etc. 117                         | unnamedskill-40 147      |
| unix and linux shell scripts 114 | spring framework 146     |
| vue.js 113                       | unnamedskill-4 137       |
| angular 113                      | unnamedskill-17 137      |
| clojure 108                      | oracle 133               |
| sh 108                           | unnamedskill-10 130      |
| ksh 108                          | bash 119                 |

Figure 4.1: Top 50 most common skills, right with skills with no name (red) included



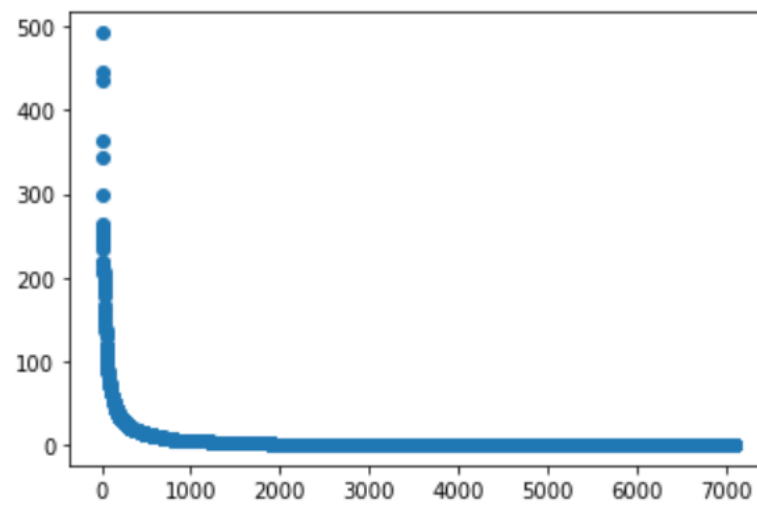


Figure 4.2: Graph about skill occurrences

# 5 Processing the data

## 5.1 Preprocessing

### 5.1.1 Formatting

Before data can be processed it requires to take a look more closely and trying to clean it as best as one can. The received data needs to be formatted so it can be processed by the selected programming language with ease. Luckily the scale of the skill level is limited so there should not be any outliers above or below the predetermined scale. But this is something that also should be checked and fixed if any are found.

The data was packed so that it came in multiple pieces so first I had to combine the data together. And doing so I get some first look at the contents. Javascript was used to combine these datasets as they were all in JSON format and the dataset was not that large that it would have required a more performing language to process. The dataset contained two tables, first one, call it skill table, containing the skill ids and their names, and the other one, call it the user skill table, containing user id, skill id and user skill level with that skill as a number from 0 to 100 as well as their interest towards that skill. After the first run it was found that there are 5624 skills in the skill table and 35120 user skills in the user skill table. But looking closely the user skill table had 7426 different skill ids so that means about 1800 skills are missing from the skill table.

### 5.1.2 Validating

It is also important to check for missing values and try to figure why those are missing and if possible try to deduce what those missing values could be. Knowing that this dataset also contains data from older versions of this application and knowing that the data has gone through several migrations it is highly possible that some values are invalid or unusable. It is important to check how much of the data is possibly affected by those migrations and try to fix them.

Like said before about 1800 skills have their ids missing from the skill table. After trying to figure out why they are missing and finding those missing skills, I came to the conclusion that they are nowhere to be found. Most likely scenario was that they used some old id that was removed from one of those migrations where the old data was migrated to support the new system. But the migration failed to clean those same ids from other places leaving these traces of old data.

We are trying to find similarities between skills so leaving these skills in the data can actually be helpful for the results. Even though these skills have no name, they still can help find the similarities between skills. So I decided to just leave those skills there and named them just as unnamed skills.

### 5.1.3 Separating skills

Looking more closely on the skills that are in the table, it was found that they contain a lot of old data that still allowed users to fill their skill with text field, meaning that they can contain anything. I found that many users had used one field to list multiple skills and on top of that, different users used different separators. Looking more closely I decided to separate the skill names into separate skills if they contained any of the following characters: comma, ampersand, slash or bracers. Again Javascript was used to separate these names as the data was in JSON format. See code 1 and 2. After separation, if we find a skill that already exists, we do not

create a new skill but instead add this same skill to the user to keep the amount of duplicates low and save the similarity data.

There is also a lot of skill names containing the word "and" but looking more closely to those, I decided that most of those actually require the word "and" between them. After this separation we now have 41415 user skills and 8983 different skills. The result from preprocessing is two files, the first one containing the skills and the other contains the user skill in JSON-format.

As to give examples on how the data looks after the processing I present you some example with the Table 5.1. Each row contains the information of one skill of a single user. The fields are the same as before, but now the skill display name is embedded to the information about the skills of the users.

| id | skillId | skill   | skillLevel |
|----|---------|---------|------------|
| 12 | 321     | Angular | 64         |
| 12 | 332     | NodeJs  | 14         |
| 23 | 9019    | React   | 100        |

Table 5.1: Example of processed data

And lastly here is an example of what kind of data I am using in the clustering (Table 5.2). I created a matrix where each row is a skill and each column is a user. And the values are the skill levels of the user with that skill. This gives us nice vectors on user skill levels which we can then compare on the similarity to create clusters.

|      | 12 | 23  |
|------|----|-----|
| 321  | 64 | 0   |
| 332  | 14 | 0   |
| 9019 | 0  | 100 |

Table 5.2: Example of data used in clustering

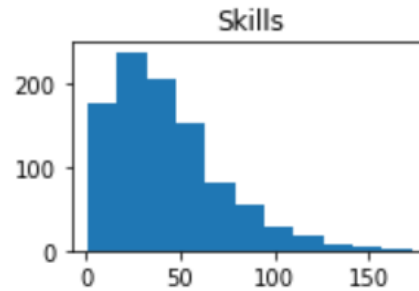


Figure 5.1: Histogram on skill distribution (x-axis is the amount of skills per user and y-axis is the number of users with this number of skills)

## 5.2 Processing

Processing is going to start from getting a look of the data and doing some diagrams on how the data is distributed and can we detect something from those distributions.

I used jupyter notebook and python libraries like pandas to create the dataset from the tables and matplotlib to draw the diagrams. The histograms were drawn using the matplotlib histogram method with the Sturges' formula on the bin width. The first figure shows the distribution of the amount of skills per user. I am expecting something like 10 skills per user. As we can see that most users have around 50 skills, which is still quite a bit more than I expected. I think that this will help us find the similarities because the user has more skills in their profile than expected thus expanding the possible matches for the skill similarities. (Figure 5.1).

The second figure shows the distribution of levels that users have rated their skill level with that particular skill. Although users currently can only use zero to five scale, it used to be 0-100. I am expecting to see these levels well distributed between all skill levels. As seen from the figure (Figure 5.2) the levels are evenly distributed. There are some peaks on the spots that map the current system to the old one. For example the new scale of 4 is located near the 75 of the old system.

The third figure shows the distribution of interest in that skill rated by the user.

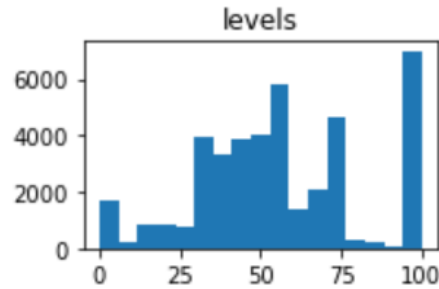


Figure 5.2: Histogram on level distribution (x-axis is the skill level and the y-axis is the number of skills with that level)

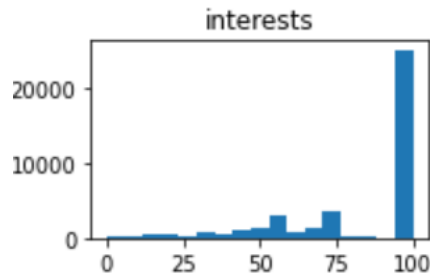


Figure 5.3: Histogram on interest distribution (x-axis is the user interest in skill and the y-axis is the number of skills with that interest)

Clearly most of the skills have an interest of 100 so this data does not help us that much (Figure 5.3).

### 5.2.1 First clustering

Clustering is the important part here as I am interested to find if we can group these skills together in clusters that well define certain types of skills. Clustering is done only with the skill levels because as seen from the histograms earlier the interest level was heavily sided on the top end. For example I expect to find clusters that one contains the front-end skills and the other back-end skills. Maybe some cluster could contain the general skills in software development like using the *git* and some may contain about managing the software development for example tools like JIRA

or using agile.

K-means clustering is going to be used as it is general clustering algorithm that is used in various applications. At least it is going to be a good starting point for the initial results. I am using the Scitkit learn toolkit for their K-mean algorithm and start by using default values and try to change those around after the first results to see if I can improve the results somehow. The most important default parameters used aside from cluster count are the number of time the k-means algorithm will be run with different centroid seeds, which is in this case 10, maximum number of iterations per run, which is in this case 300, and the tolerance on which the cluster is considered converged, which is in this case 1e-4.

The clustering is calculated based on the user skill levels. One vector contains user skill levels on all of the skills that are present in this dataset.

With just eight clusters, there are some good clusters that in my opinion contains some similar skills, but it is hard to determine if those are actually the closest skills together. We clearly had front-end skills in one group and skills that are used to manage software development in another group. But there is also always one cluster that contains "everything else". It does not matter if the cluster count is five, ten or even fifty, all other clusters are between two and twenty and one always has the rest containing over 6000 skills. Based on these results, the clustering did not achieve the results wanted.

Next I tried to change the default values to try to improve the results. The most notable difference came from changing the maximum number of iterations to 1000. This notably increased the run duration but also improved the results in my opinion. Clusters had less skills that I think that did not belong to that cluster. But still however there is still the one cluster with over 6000 skills and there are still a lot of skills that I would move to another cluster.

Increasing the maximum number of iterations over 1000 did not change the result

```
Default data
Cluster : 0, Length: 34
Cluster : 1, Length: 6854
Cluster : 2, Length: 48
Cluster : 3, Length: 7
Cluster : 4, Length: 11
Cluster : 5, Length: 81
Cluster : 6, Length: 2
Cluster : 7, Length: 4
Cluster : 8, Length: 37
Cluster : 9, Length: 10
```

Figure 5.4: Cluster size with default data

and the run time would also remain the same meaning the clusters did manage to converge before that. Changing the tolerance or the amount of runs per different centroid seed did not seem to have much effect on the results for the clustering.

Conclusion from the first clustering is that it did not succeed like I expected. Maybe the data needs some more refining in order to get some clear clusters out of it. I understand that the big over 6000 skill cluster is there to catch all of the "outlier" that are present with only a couple of users. But suspect that these skills would still belong to some other cluster and I would really like to get rid of that huge cluster. See Figure 5.4.

Next I am going to try to refine the data even further to see if the results improve. I am approaching this with normalizing and standardizing the data and then comparing the results between the previous results and results after the refinement.

### 5.2.2 Standardizing and normalizing the data

Normalizing and standardizing is important. This ensures that all of the values have the same importance in the model. There are numerous ways to achieve the normalization and/or the standardization of the data. One of these is called *z-score* standardization that standardizes features by removing the mean and scaling to unit



```
Normalized data
Cluster: 0, Length: 6987
Cluster: 1, Length: 50
Cluster: 2, Length: 4
Cluster: 3, Length: 2
Cluster: 4, Length: 8
Cluster: 5, Length: 6
Cluster: 6, Length: 14
Cluster: 7, Length: 2
Cluster: 8, Length: 5
Cluster: 9, Length: 10
```

Figure 5.5: Cluster size with normalized data

```
Standardized data
Cluster: 0, Length: 28
Cluster: 1, Length: 1344
Cluster: 2, Length: 1602
Cluster: 3, Length: 295
Cluster: 4, Length: 44
Cluster: 5, Length: 81
Cluster: 6, Length: 30
Cluster: 7, Length: 74
Cluster: 8, Length: 67
Cluster: 9, Length: 3523
```

Figure 5.6: Cluster size with standardized data

variance. This is done with z-score which is calculated by subtracting each value with the mean value and dividing the result with standard deviation.

After normalizing the data with the Scikit-learn Normalizer method the clusters did not change like was expected (Figure 5.5).

I used the Scikit-learn StandardScaler method that uses z-score to standardize the data and then try clustering with that data. The cluster size got way more even, but it is hard to know if they contain similar skills as they contain so many skill and looking what they contained there were always some skills that in my mind they did not belong to the same cluster.

Normalization brought the standardization result with even more evenly dis-

```
Normalized and standardized data
Cluster: 0, Length: 2451
Cluster: 1, Length: 329
Cluster: 2, Length: 1029
Cluster: 3, Length: 23
Cluster: 4, Length: 142
Cluster: 5, Length: 9
Cluster: 6, Length: 2021
Cluster: 7, Length: 31
Cluster: 8, Length: 1048
Cluster: 9, Length: 5
```

Figure 5.7: Cluster size with normalized and standardized data

tributed clusters (Figure 5.7). At a glance it is very hard to determine if these clusters properly contain similar skills due to the amount of skills in them. But the sizes seem quite promising, but we still have those huge clusters with multiple thousand skills and those with only few.

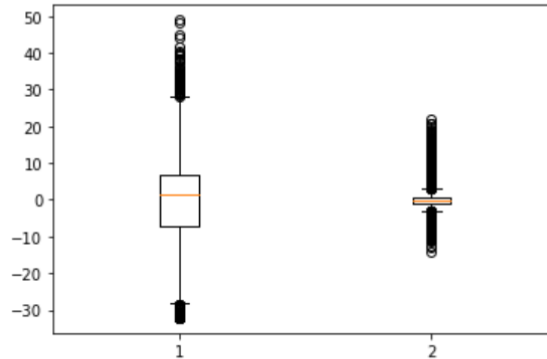
Currently the data used in this has high amount of features or dimensions. Next I am trying to apply *PCA* (Principal Component Analysis) to the data to see if it can improve the results.

### 5.2.3 Using the Principal Component Analysis

PCA lowers the amount of dimensions or features in the data. In this case we are comparing thousands of skills against each other and each skill is its own feature so the data has several thousands of features. Using the PCA can greatly speed up the processing and it can also be used to remove or lower the impact of features or dimensions that are not important for the results.

Using Principal Component Analysis to project the normalized data into fewer dimensions and trying the clustering with that data. I tried using different amounts of dimensions to determine the best result to keep the distribution of skill similar. I visualized the results using Matplotlib Boxplot to see the distribution of each dimension and to see how much they affect the final result by looking at the outliers and where the majority of skills are located. From the Figure 5.8 we can see that it

PCA with 2 dimensions explained variance ratio:  
 [0.144713 0.00519422]



PCA with 10 dimensions explained variance ratio:  
 [0.144713 0.00519424 0.00423593 0.00320724 0.00309625 0.00287263  
 0.00269321 0.00259305 0.00246091 0.00237518]

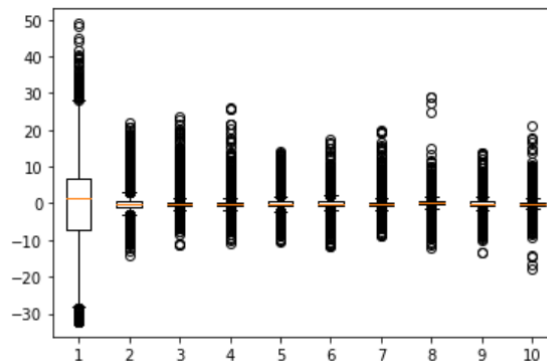


Figure 5.8: Boxplot from different PCA dimension count with percentage of variance explained by each of the selected components

seems that only the first feature is important for the distribution. The rest of the features are located near the center meaning they might be able to reduce to fewer features. I did more testing with even bigger sizes of features but the results were the same that only the first feature seemed to have any impact.

I used PCA to lower the dimensions to only 2 dimensions so I can try to visualize the result properly. The cluster size again got way more evenly distributed as we can see from Figure 5.9. Now all of the clusters got so big it is extremely hard to determine if all of the skills are properly similar.

Visualizing the PCA clustering result we can see some clearly defined clusters

```

PCA to 2 dimensions with standardized and normalized data
Cluster: 0, Length: 973
Cluster: 1, Length: 626
Cluster: 2, Length: 1150
Cluster: 3, Length: 1845
Cluster: 4, Length: 283
Cluster: 5, Length: 298
Cluster: 6, Length: 247
Cluster: 7, Length: 1027
Cluster: 8, Length: 106
Cluster: 9, Length: 533

```

Figure 5.9: Cluster size with using PCA

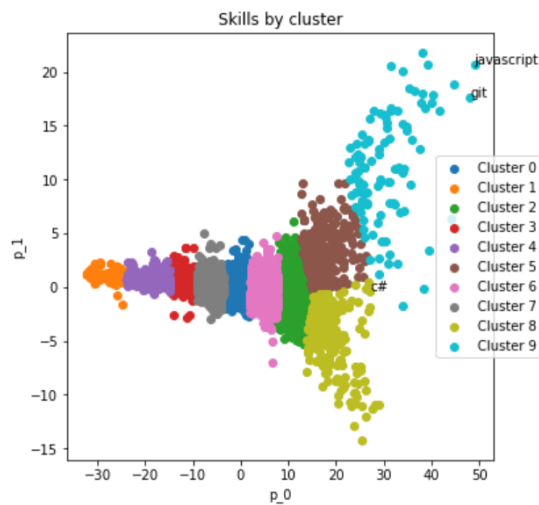


Figure 5.10: Visualizing PCA result

(Figure 5.10). All of the most common skills are located on the top right section of the plot. I also included some skills in the visualizing to present that most of the popular skills are located in the right side.

#### 5.2.4 Conclusion from clustering

The clustering did not achieve the result that was wanted. Maybe the data does not have that strong connections between the skills of same type, but instead quite loose connection with multiple types. This makes the clustering this type of dataset quite hard. I believe that trying to refine this clustering even further does not improve the results significantly. I think that it would not be worth trying to pursue with

| skill      |           |
|------------|-----------|
| javascript | 1601203.0 |
| css        | 1010231.0 |
| git        | 996183.0  |
| java       | 857761.0  |
| scss       | 691003.0  |
| node.js    | 674725.0  |
| sass       | 673344.0  |
| xml        | 670820.0  |
| react.js   | 665641.0  |
| x html     | 647477.0  |

Figure 5.11: Dot product ranking normal data and results

this approach any further.

Instead I think the next step would be to trying to actually compare skills against each other. This way we can create a sort of ranking system to determine which skills are similar. Using dot product on the dataset we can create a similarity matrix that tries to represent a numeric value of the similarity between two types of skills.

### 5.2.5 Using dot product to create similarity matrix

Taking a dot product from the skills we can create a matrix on how similar skills are compared to each other. I used normal dot product calculation for dot product and Scikit-learn StandardScaler with z-score to standardize values. I compared four different results. We use JavaScript as the target in all of these and show the top ten of most similar skills, based on the dot product. Note that all of the "unnamed skills" have been filtered from the results. The first image shows the results when the data before dot product and the dot product are not standardized (Figure 5.11). As we can see in the results there are some skills that I think are quite close to JavaScript, for example the "css" and "node.js". But some are not and I think they appeared here only because they are very common amongst all users.

In the second image we can see the results when the data is not standardized but

|                     |                  |
|---------------------|------------------|
| <b>skill</b>        |                  |
| <b>javascript</b>   | <b>31.761283</b> |
| <b>react</b>        | <b>28.423816</b> |
| <b>typescript</b>   | <b>27.037276</b> |
| <b>graphql</b>      | <b>25.903543</b> |
| <b>jquery</b>       | <b>25.746951</b> |
| <b>node.js</b>      | <b>25.736796</b> |
| <b>reactjs</b>      | <b>25.649805</b> |
| <b>react native</b> | <b>25.154997</b> |
| <b>vue.js</b>       | <b>25.081358</b> |
| <b>webpack</b>      | <b>24.983641</b> |

Figure 5.12: Dot product ranking normal and standardized results

|                   |                    |
|-------------------|--------------------|
| <b>skill</b>      |                    |
| <b>javascript</b> | <b>6132.543301</b> |
| <b>node.js</b>    | <b>2991.358258</b> |
| <b>typescript</b> | <b>2848.817175</b> |
| <b>css</b>        | <b>2769.578613</b> |
| <b>react.js</b>   | <b>2741.500133</b> |
| <b>jquery</b>     | <b>2619.633720</b> |
| <b>scss</b>       | <b>2347.961633</b> |
| <b>sass</b>       | <b>2158.183616</b> |
| <b>git</b>        | <b>1993.042284</b> |
| <b>vue.js</b>     | <b>1955.138913</b> |

Figure 5.13: Dot product ranking standardized data and normal results

the dot product is standardized (Figure 5.12). This completely changes the results but these still seem like good results, all seem to be related to JavaScript, but that may be because it is a common skill.

The third image show the results when the data for dot product is standardized but the result of dot product is not (Figure 5.13). Here everything again is related to JavaScript, but I would say that the order is better than in results before.

The fourth and final image show the results when the data and the result of the dot product are standardized. These results again change the order quite significantly, and again some of these results are quite off. (Figure 5.14)

In my opinion the result with standardized data for dot product but not stan-

|                   |           |
|-------------------|-----------|
| skill             |           |
| javascript        | 42.746369 |
| react js          | 28.540476 |
| microsoft tfs     | 27.453182 |
| typescript        | 26.233114 |
| react query       | 25.876800 |
| react context api | 25.876800 |
| node.js           | 25.331292 |
| kcml              | 25.059614 |
| jquery            | 24.870981 |
| apollo client     | 24.739246 |

Figure 5.14: Dot product ranking standardized data and results

|                  |             |                         |           |
|------------------|-------------|-------------------------|-----------|
| skill            |             | skill                   |           |
| python           | 935496.0    | python                  | 35.179502 |
| git              | 577842.0    | django                  | 26.655642 |
| java             | 530847.0    | pandas                  | 26.571061 |
| javascript       | 508081.0    | machine learning        | 25.397181 |
| jenkins          | 381588.0    | flask                   | 24.918074 |
| docker           | 331364.0    | matlab                  | 23.924658 |
| linux            | 299833.0    | r language              | 22.936473 |
| robot framework  | 293827.0    | numpy                   | 21.074551 |
| postgresql       | 292896.0    | robot framework         | 19.741578 |
| c++              | 288097.0    | scripting               | 19.588894 |
| skill            |             | skill                   |           |
| python           | 5889.236588 | python                  | 56.196011 |
| robot framework  | 1823.618735 | django                  | 30.323536 |
| jenkins          | 1537.730327 | communication protocols | 30.265829 |
| django           | 1508.199297 | selenium library        | 30.246215 |
| pandas           | 1408.226799 | electronics design      | 30.095445 |
| selenium library | 1268.956545 | robotics                | 30.095445 |
| matlab           | 1254.108062 | embedded developer      | 30.095445 |
| machine learning | 1214.300651 | embedded testing        | 28.917449 |
| scikit-learn     | 1186.540328 | pytest                  | 28.179493 |
| numpy            | 1156.159821 | pandas                  | 27.723482 |

Figure 5.15: Dot product ranking with python

standardized result of the dot product presented the best results. Here in the case of JavaScript we can still see some similarities, but if we change to some more uncommon skill, the standardized results have drastically different results to the ones without standardizing the results. This requires more testing to make sure this is consistent among most of the skills. Here are the results using "python" (Figure 5.15). Here the top row has normal data and bottom row has standardized data and the left column has normal results and right standardized results. As we can see from the python example that it reinforces the claim on the JavaScript case.

### 5.2.6 Cosine similarity

Just using the similarity matrix we can easily see how different skills are ranked against each other. But not necessarily how similar they are between each other. We can use cosine similarity to calculate how similar the vectors in the matrix are. This can give us a number on how similar the skills are based on the data we currently have. I am comparing different cosine similarity results from using data without standardization or normalization, data with only normalization, data with only standardization and data with both standardization and normalization.

I am expecting the results varying between the different types of data, but I think that the same result from previous test also apply here, so that the result with just standardized data would give the best results.

I used Scikit-learn cosine similarity method to calculate the cosine similarity between the skills. I tested this with different skills but to highlight the results I am presenting to you the results using JavaScript as an example.

From the Figure 5.16 we can see that results are somewhat similar to each other. Some skills are ranked differently in each of the results. The main difference here is that each of the results except the result with standardized data is containing skills "Git" and "Java" in them. They are both very common skills and so is JavaScript which is why I think they appear so high in the similarity. However the result with the standardized data does not include the mentioned skills in them. Also the skills contained in this list are in my opinion very close to JavaScript or JavaScript environments in general.

### 5.2.7 Presence of unnamed skills

The earlier mentioned unnamed skills are filtered out from the results but they are still included in the data. If we bring back the discussion on what to do about the unnamed skills we can start using this similarity data to try to figure out what skill



```

----- Default -----
skill
javascript    1.000000
node.js       0.965752
jquery        0.958324
react.js      0.957220
git           0.954046
css           0.953256
typescript    0.943159
angularjs     0.941215
mongodb       0.932659
java          0.931446
Name: javascript, dtype: float64

----- Standardized data -----
skill
javascript    1.000000
node.js       0.738164
react.js      0.721449
jquery        0.685429
css           0.668091
typescript    0.650856
scss          0.610214
vue.js        0.605398
sass          0.600626
angularjs     0.578543
Name: javascript, dtype: float64

----- Normalized data -----
skill
javascript    1.000000
node.js       0.962179
jquery        0.951188
react.js      0.950265
css           0.949903
git           0.946724
typescript    0.941055
java          0.931134
angularjs     0.930746
mongodb       0.927376
Name: javascript, dtype: float64

----- Norm and std data -----
skill
javascript    1.000000
node.js       0.915048
typescript    0.908090
git           0.884685
mongodb       0.862249
postgresql    0.851385
react.js      0.846056
java          0.843247
php           0.833809
vue.js        0.833080
Name: javascript, dtype: float64

```

Figure 5.16: Cosine similarity with JavaScript

they actually may be.

Looking at the Figure 5.17 we can see how the skills are ranked when comparing to "unnamedskill-8". I chose this skill as it is the most common skill of the ones that does not have a name. From the figure we can see that it is considered to be close to Git, Subversion and SVN which all are some sort of version controls. This means that with high probability this skill with no name might actually be some of these and almost certainly it is linked to version control. Using the same method for the second most common skill of the ones that does not have a name, "unnamedskill-85", I can say with high probability that it is either mariadb or mysql, because they both are considered similar with the unnamed skill. At least the skill has something to do with similar databases as those two.

But they are not a exact match. This means that they also might contain some other similarity data that the others do not. So for now I am going to leave them as is and maybe later if needed might run this type of conversion to the closest match

```
----- Standardized data -----  
skill  
unnamedskill-8      1.000000  
git                 0.767190  
subversion          0.515306  
svn                 0.500697  
unnamedskill-22    0.497560  
java                0.492231  
unnamedskill-10    0.481320  
unnamedskill-4     0.455975  
jenkins             0.453290  
unix                0.452429  
Name: unnamedskill-8, dtype: float64  
-----
```

Figure 5.17: Cosine similarity with an unnamed skill

and see if the results of the similarity change drastically.

### 5.3 Conclusion from processing

The clustering did not succeed like expected. The clusters were not bound strongly by single type of skill instead they contained random skills that can be explained by the data. Like said before I believe that the data contains loose connection between multiple skills and they are not focused on the same types of skills.

However using the dot product to create similarity matrix and cosine similarity brought some great results. The ranking and cosine similarity showed very promising results and the next chapter will go through the validation of these results.

# 6 Validation

This chapter goes through the evaluation of the results from the processing. Now that I have the similarity data I need to validate the results. This is not something as easy as to just comparing the results to some other data to see how well it performed. In this chapter I also discuss the possibilities of the similarity data and how it can be used in applications.

Validating the results from the processing is not an easy task. There is no ready database to compare the results to verify. Instead we need to use people to give their take on how similar the skills are. Of course when we incorporate humans to give their opinion, it is never fully objective. This is why we need multiple experiments or interviews with different people to achieve the best possible result on the validation. Due to the limitations of current situation, only way to evaluate the results is with an interview.

## 6.1 Evaluating with experiments/interviews

There are many ways to try to validate these results with data from experiments. One way to approach this would be to try to create the same kind of similarity matrix from interviewing different people. But with dataset of this size it is quite difficult and cumbersome to make people rank all of the skills with all of the other skills. Not to mention that the person ranking would need some information about all of the skills in order to know where to place it. Of course it would be more

feasible to only include the most common ones, both from the processing and for the people to rank. This way we could compare these two together and it would give us a number on how well the processing of the data performed. This would give confidence on the similarity matrix from the processing that it is on to something if the results would be similar.

Another way to approach this is to have people go through the similarity matrix from the processing and give their own take on how well it performed. This could happen by interviewing people and presenting them with different results from the processing. One option could be to present the four different results from the processing when the data and the result was scaled or not and ask them which one of these would they consider the most similar. Another option would be to give for example the same top ten results for one skill in random order and ask them to select the ones that are similar to the given skill. Another variation to this would also be to ask them to rank them based on similarity and then comparing this to the actual result.

## 6.2 Validation methods

For the interview I need to make sure that the person is well qualified to speak on the topic. The most optimal person would be to use someone that was involved in assigning this topic of creating this semantic search as they would most likely have some picture in their mind as of what the end result should contain. But unfortunately this is not possible so this adds a limitation to validating these results. But I believe that anyone who knows the field is well suited to give their take on the results in order to give the results more validation.

During the interview I would go through the methods used in the thesis so that we can validate if the thesis produced good results and maybe if they can be improved somehow. In the interview I will use around 10 skills from the top 100

most common skills. I will select these skills myself so that I think these skills would give the best overview of the results. I will show results of the clustering for the interviewee to validate those results if they were a success or not. I will also show results of the dot product ranking and the cosine similarity score on the skills with the variations mentioned in the thesis. These variations included standardizing and not standardizing the data to see how the results would change. I also include 2 skills with no name with the cosine similarity to present how this data can be used as a tool to help determine what these skills could be.

### 6.3 Interview materials

The whole interview is presented in the Appendix B. The interview is going to be semi-structured, meaning the actual order and phrasing of the questions are not predetermined. But the overall theme and content for the questions are determined beforehand. There also might be some questions that are not presented here in the materials but those will be well documented in the results section. This section shows the highlights of its content. It is possible for the interviewee to not know all the possible skills that might be shown so it is allowed to look up what those skills are.

First in the interview I will go through the results from the clustering. I will explain how the clustering was done and show the sizes of the cluster as well as a sample from each cluster of what skills are inside. See Figure 6.1. With this part I want to know what they think about the clustering, whether it was a success or not and why they think so.

**Q1: Do you think the clustering was a success or not and why?**

After that I will explain how the similarity matrix was created and show different results from there. First I will show just a list of skills and ask if they are similar to the top skill. I would also like to ask if one would change any of these skills to

```

Cluster: 0, Size: 3473
Cluster: 1, Size: 67
Cluster: 2, Size: 2230
Cluster: 3, Size: 55
Cluster: 4, Size: 34
Cluster: 5, Size: 1036
Cluster: 6, Size: 44
Cluster: 7, Size: 54
Cluster: 8, Size: 53
Cluster: 9, Size: 42

```

First cluster

```

Index(['"can do" attitude', '1', '10', '14', '17',
      '1st - 3rd line support for applications and workstations',
      '200 kantataulua', '2003', '2003 - 2016', '2007',
      ...
      'ylläpito', 'yocto project', 'yup', 'zachman framework', 'zef',
      'zend framework', 'zephyr for jira',
      'zepto and other popular libraries', 'zoom', 'zustand'],
      dtype='object', name='skill', length=3473)

```

Second cluster

```

Index(['apple time machine', 'bpfftrace', 'collectd', 'debian-installer',
      'dhcpd', 'dovecot', 'dtrace', 'embedded linux', 'gdb', 'gnu autotools',
      'hiera', 'hp procurve switches', 'ida pro', 'illumos zones',
      'infrastructure as code', 'ip networking', 'iptables', 'ipv6',
      'isc bind', 'juniper', 'junos switches', 'ldap', 'linux kvm',
      'linux mdraid', 'linux system administration', 'lvm', 'lxc', 'make',
      'mdb', 'meson', 'microsoft hyper-v', 'netfilter', 'nsd', 'ollydbg',
      'openbsd pf', 'openbsd vmm', 'openembedded', 'opensmtpd', 'packaging',
      'pam', 'pki', 'postfix', 'public key infrastructure', 'puppet', 'pxe',
      'receive based systems', 'rrdtool', 'solaris', 'systemd',
      'systemd-nspawn', 'tcpdump', 'traditional unix visibility tools',
      'unbound', 'unix programming', 'unnamedskill-1099', 'unnamedskill-601',
      'unnamedskill-767', 'unnamedskill-786', 'unnamedskill-803',
      'unnamedskill-835', 'unnamedskill-928', 'unnamedskill-996', 'wireshark',
      'xen', 'zabbix', 'zfs', 'zfs send'],
      dtype='object', name='skill')

```

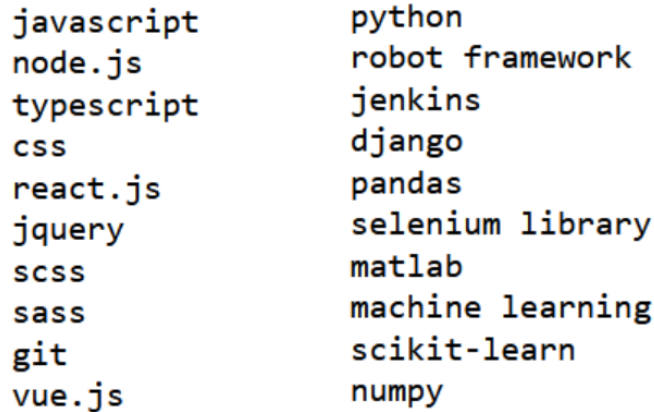
Third cluster

```

Index(['+ browser dev tools', '-toimiston tuki', '10 mobile',
      '1st line support for cash registers',
      '1st line support for pda devices and mobile devices', '2', '2.0',
      '3-tier models', '3.5', '3d design',
      ...
      'xpages', 'xrebel', 'y2k', 'yagni', 'yeesql', 'yeoman', 'yourkit',
      'zed attack proxy', 'zendesk', 'zkoss'],
      dtype='object', name='skill', length=2230)

```

Figure 6.1: Interview materials showing cluster size and sample from first three clusters on what skills they contain.



```
javascript      python
node.js        robot framework
typescript     jenkins
css            django
react.js      pandas
jquery        selenium library
scss          matlab
sass          machine learning
git           scikit-learn
vue.js        numpy
```

Figure 6.2: Interview materials showing two lists from the similarity matrix

the other list. I will use "JavaScript" and "Python" as the reference skills here. See Figure 6.2. The purpose of this question is to get feedback on the similarity matrix that produces similar skills.

**Q2: Do you think that these lists contain similar skills?**

Next in the interview is comparing the different scaling methods and how they affect the results. I will show a list of results with different scaling methods and ask which of these lists look like they have the most similar skill together. I will show these lists as "Java" and "Git" as reference. See Figure 6.3 showing these lists with Java as reference. The goal of this part is to verify which of these methods give the best results as was discussed in the earlier chapter.

**Q3: Which one of these lists do you consider containing the most similar skills compared to the top most one?**

Next I will show lists from the similarity matrix using "Scrum" and "MySQL" as reference. Here I will ask if the lists are properly ordered based on similarity. See Figure 6.4 The goal of this step is to get feedback if the similarity is correctly weighed on a certain skill so that the order of the skills based on similarity is good.

**Q4: Do you think the order in these lists is good?**

Next in the interview I will go through some results from the cosine similarity

|                  |                  |
|------------------|------------------|
| java             | java             |
| spring boot      | spring framework |
| kotlin           | spring boot      |
| spring framework | apache maven     |
| vaadin framework | junit            |
| junit            | hibernate        |
| hibernate        | mysql            |
| scala            | postgresql       |
| apache maven     | mariadb          |
| android studio   | javascript       |
|                  |                  |
| java             | java             |
| spring boot      | javascript       |
| jakarta ee       | git              |
| spring framework | python           |
| vaadin framework | css              |
| kotlin           | mysql            |
| guice            | spring framework |
| junit            | postgresql       |
| hibernate        | mariadb          |
| spring portlets  | jenkins          |

Figure 6.3: Interview materials showing four lists with different scaling

|                          |                  |
|--------------------------|------------------|
| scrum                    | mysql            |
| kanban                   | mariadb          |
| safe                     | postgresql       |
| agile                    | java             |
| atlassian jira           | mongodb          |
| agile project leadership | php              |
| sitemfinity cms          | oracle           |
| agile project management | javascript       |
| scrum master             | apache tomcat    |
| lean                     | spring framework |

Figure 6.4: Interview materials two list to compare the order



| skill                 |          |
|-----------------------|----------|
| aws                   | 1.000000 |
| amazon web services   | 0.999513 |
| docker                | 0.518207 |
| gcp                   | 0.481041 |
| google cloud platform | 0.462584 |
| postgresql            | 0.438887 |
| kubernetes            | 0.415993 |
| jenkins               | 0.394069 |
| azure                 | 0.390455 |
| aws dynamodb          | 0.390392 |

Figure 6.5: Interview material showing cosine similarity with "AWS"

and how we can measure the similarity between skills. I will show examples using "C", "Docker", "AWS" and "Linux" as reference points and ask their opinion about if they agree or not about the similarity score with cosine similarity between the skills. See Figure 6.5 for an example with "AWS". The goal is to get feedback if this is a good way to measure the similarities between the skills.

**Q5: Do you think that this is a good way to measure similarity?**

Lastly I will go through the presence of skills with no name. I will show two examples of skills with no name and their cosine similarities from the dataset. See Figure 6.6 for an example. The goal is to get feedback if this is something that can be used to determine what these skills with no name could be. And maybe use this tool to automatically convert those skills into corresponding skills to improve the quality of the dataset.

**Q6: Do you think that this can be used as a tool to find what those skills with no names could have been?**

|                        |                 |
|------------------------|-----------------|
| <b>skill</b>           |                 |
| <b>unnamedskill-8</b>  | <b>1.000000</b> |
| <b>git</b>             | <b>0.767190</b> |
| <b>subversion</b>      | <b>0.515306</b> |
| <b>svn</b>             | <b>0.500697</b> |
| <b>unnamedskill-22</b> | <b>0.497560</b> |
| <b>java</b>            | <b>0.492231</b> |
| <b>unnamedskill-10</b> | <b>0.481320</b> |
| <b>unnamedskill-4</b>  | <b>0.455975</b> |
| <b>jenkins</b>         | <b>0.453290</b> |
| <b>unix</b>            | <b>0.452429</b> |

Figure 6.6: Interview material showing cosine similarity with a skill with no name

## 6.4 Results from the interview

Even though I got some results back from the first interview and immediately had some ideas on how to improve the interview I decided that I will keep the interview the same for all of the interviews so that they all would have the same materials. I interviewed three people for this interview. All of them were more than suited for this interview. First of them had previously worked for this same project and now worked as a senior developer. The second person works as the CIO of the company Y that actually uses this application. This person has some good insight on the application as he has worked with it more than I have. And the last person has good knowledge on multiple different skills and is currently working as a freelancer. I will present the results from the interview from question to question and share the main points that came up during the interview.

**Q1:** The list that I presented definitely should have been cleaned a bit more. The lists contained a lot of information that was only not needed but actually hindered reviewing these results. The lists were sorted alphabetically, which was not the

correct solution. It would have been very helpful to sort these lists so that they would be in order of how many times these skills were present in the data. This would then show the most relevant skill for that cluster first and give the interviewee better understanding what are the common skills that bind these all together. One good point that also was raised is that these lists should have been filtered so that skills with less than some amount of occurrences in the data would be filtered away. All-in-all some of the clusters seemed very balanced in terms of size. Others had the majority skill in them and were kind of trash bins containing all sorts of things. One thing that was raised is that the smaller the cluster the more concise it was. Especially the fifth cluster contained very similar or relevant skills. Few suggestions for the future were that the most insignificant skill would be filtered away so that the data actually had skills that were being used.

**Q2:** The main points that were said here were the fact that maybe these skills in some cases are not that similar but rather relevant to each other. In the JavaScript list all of the other skills are related to web development except Git. I believe that Git raises here because JavaScript and Git are both in the top of the list of skills when it comes to occurrences. One point that was said was that this list could be used to help find suitable people for educating new skills. And it was said about the list with Python as reference point that most of these skills do not even use Python as a language if they are a framework or a tool. Instead this list tells that if a person knows Python, they most likely are into machine learning and DevOps. But in the end both of these lists were said that they looked very good and contained similar skills.

**Q3:** This question showed how differently these lists can be made even with small changes like standardizing the data. All three interviewees though the same ones were the best ones, these were the top-left in the first example and top-right in

the second example. However there were some tough choices. Everyone said that the bottom-right was in both cases just full of random skills so we can clearly rule that version out. Result of this question gave me a thought that maybe some small variations in the data, for example standardizing it, are better with other skills than others. From my own testing I usually found that the top-right was the clear winner but in the case of Java the top-left is clearly better. So maybe this has something to do with how many occurrences these skills and similar skills to those have.

**Q4:** When it comes to the order of these lists there is still a lot of improvement to do. Although they contained similar or relevant skills the ordering was not optimal. For example if you are looking for a person to do Scrum you would not take someone with experience with Jira over someone who is considered Scrum master. This repeats the same problem that was found before that if you are looking for someone with experience in a specific skills this list does not guarantee that the person has any skill in that even if those skills are considered similar. Clearly this way of ordering the skills does not tell how close some skills are compared to others. Instead it tells how likely these skills are relevant to each other.

**Q5:** All interviewees agreed that showing a number is a better way to compare the relevancy between skill than just showing an ordered list. This gives a good estimation on how similar this model thinks these skills are. Even though the same problems from before are present here like some skills are higher up than others when it should be the other way around. Some points that were noted were about the values in the list. In the C language example the C++ was quite close but everyone thought that it should have been closer. And the values in the list dropped way quicker than compared to other skills. This was thought to be caused by the fact that people with C skill do not have many other skills. Other point that was noted was on how close the "aws" and "amazon web services" were because they are the

same thing and it was impressive to find that the model also found that. When it comes to the example with operating systems all of the operating systems are in the top part. But it was quite interesting to find the "unix and linux shell scripts" in the list so high. It was thought that just "shell scripts" should have been higher and that was way too specific to be so high in the list. This number representation was said to be a good representation of the possibility of that person knowing the other skill also. And this would be a great tool to help recruiters to rank suitable candidates on a new project based on a number rather than just looking through a list of skills.

**Q6:** When it comes to presence of these unnamed skills this number representation would certainly help to find what those missing skills are linked to. Even though they are not as close as the "AWS" and "Amazon Web Services" so that it can be said with almost certain accuracy that this unnamed skill is the skill closest to it but it can be clearly seen that skill is linked to the same area. In the example of the "unnamedskill-8" it can be clearly seen that it is linked to version control but it cannot be said for certain that it is one of those listed version control tools.

**Other** Some other comments that were given were about how this could be a great tool for recruiting even if the person in charge of the recruitment is not familiar with the skills. Also this would be used to find new people for training new skills so that a company could just query with the skill and the application would give the best possible match of people who would be most likely interested in that particular skill. In the end the interview was a great success and it gave me some good feedback on the results.

# 7 Conclusion

## 7.1 Answers to the research question

**RQ1: What is the best method to process data of user skill levels to achieve the best possible result in similarity?** Unfortunately there were no similar studies for this type of assignment so there was not much comparing to do between results from this thesis and other papers. Although there were studies that tried to find similarities between programming languages but those used very different approaches.

Through the process of the trial and error of processing the data I can say that there is no definite answer to this question. Different processes produced different results and the process to use in some specific case should always be tailored to that specific problem. Even a small change in the process can drastically change the results. For example the simple act of standardizing the data gave very different results from the normal. And as was seen from the interview results that these small tweaks changes what the results look like and based on those results one should decide what type of processing to use. And the results may even differ depending on different characteristics of the skill in the data for example how common it is and does it have many links to other skills.

In conclusion while this study could not draw upon similar research it provides valuable insights into the processing of skill data. The trial and error approach used

in this study highlights the importance of tailoring the processing approach to the specific problem at hand. It also emphasizes the need to consider the characteristics of the skills in the data to obtain accurate and meaningful results.

**RQ2: How can we measure similarity between skills in software development?** The results of this study and the feedback from the interviewees suggest that measuring the similarity or relevance between two skills using a numerical score holds great potential. Measuring the similarity or relevancy between two skills with a number gives great advantages over for example using just a list with skills in order of similarity. When the similarity score is between zero to one it is very easy to compare how similar it actually is compared to the skill we want.

In case of this thesis I used cosine similarity to calculate how similar two vectors were and as it was seen from the results and from the interview it seemed to work very well. This kind of tool could be found useful for recruiters trying to find suitable candidates to hire or for companies trying to find suitable people to educate new skill to increase their competence with that skill.

In conclusion the use of similarity scores to measure the relevance of skills holds significant potential as demonstrated by the results of this study and feedback from the interviewees. This approach could prove to be a valuable tool for recruiters and companies seeking to identify the most suitable candidates for specific roles or training programs.

## 7.2 Discussion

In this thesis we explored the possibility of finding similarities between software development skill only using user skill levels. While there were no similar studies for this type of assignment the process of data processing revealed valuable insights into tailoring the processing approach to the specific problem at hand. Furthermore we

found that measuring the similarity or relevancy between two skills using a numerical score holds great potential offering significant advantages over using just a list of skills in order of similarity. Also the unnamed skills could be defined with this type of similarity score but I believe that this would require more data to be certain about the similarities.

The data that was used in this thesis only contained people working in a single company. Of course these people had some experience prior to joining this company but I still believe that company policies and culture would still affect how common certain skill are presented in the data. Not to mention how the data was structured and migrated multiple times making the data not so coherent. Users had previously been able to write in text field their skill and those got migrated as such.

I believe that if this same process would have been done with data from multiple companies and many more users then we would have seen even better results. Around 35 000 user skills is nothing small but this number could have been far greater. Even with data like this the results were very promising and more data would be very easy to produce.

Unfortunately due to the limitations of current situation I was not able to create the semantic search application that was the goal for the results from this thesis. However I believe that once we have this similarity data it is not that difficult to create semantic search in Elasticsearch using vectors from the matrix as the basis for vector search.

Like said before these results could be improved with even more data. Also cleaning up the data used in this by the company using this could also improve these results as that would reduce the amount of those skills that would appear only one time. One could use this matrix as the corpus for some machine learning application that tries to guess skill given similar skills and their similarity scores with that skill. In conclusion with data that was this simple to produce and still



gave such promising results holds great potential moving forward finding similarities between software development skills.

# References

- [1] H. Wang, C. Ma, and L. Zhou, “A brief review of machine learning and its application”, in *2009 International Conference on Information Engineering and Computer Science*, 2009, pp. 1–4. DOI: [10.1109/ICIECS.2009.5362936](https://doi.org/10.1109/ICIECS.2009.5362936).
- [2] Q. Wang, Y. Ma, K. Zhao, and et al., “A comprehensive survey of loss functions in machine learning”, vol. 9, 2022, pp. 187–212. DOI: <https://doi.org/10.1007/s40745-020-00253-5>.
- [3] Wikipedia. “Maxima and minima”. (2022), [Online]. Available: [https://en.wikipedia.org/wiki/Maxima\\_and\\_minima](https://en.wikipedia.org/wiki/Maxima_and_minima). (accessed: 16.11.2022).
- [4] IBM. “Supervised vs. unsupervised learning: What’s the difference?” (2022), [Online]. Available: <https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>. (accessed: 30.10.2022).
- [5] D. M. W. Powers, “Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation”, 2011.
- [6] Scikit-learn. “Overview of clustering methods”. (2022), [Online]. Available: <https://scikit-learn.org/stable/modules/clustering.html#overview-of-clustering-methods>. (accessed: 19.10.2022).
- [7] Scikit-learn. “Clustering evaluation”. (2022), [Online]. Available: <https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation>. (accessed: 19.10.2022).

- 
- [8] Scikit-learn. “Clustering - k-means”. (2022), [Online]. Available: <https://scikit-learn.org/stable/modules/clustering.html#k-means>. (accessed: 19.10.2022).
- [9] “Semantics | definition & theories | britannica”. (2022), [Online]. Available: <https://www.britannica.com/science/semantics>. (accessed: 16.11.2022).
- [10] S. Löbner, “Understanding semantics”, Routledge, 2013.
- [11] R. Guha, R. McCool, and E. Miller, “Semantic search”, in *Proceedings of the 12th International Conference on World Wide Web*, ser. WWW '03, Budapest, Hungary: Association for Computing Machinery, 2003, pp. 700–709, ISBN: 1581136803. DOI: 10.1145/775152.775250. [Online]. Available: <https://doi.org/10.1145/775152.775250>.
- [12] T. Seymour, D. Frantsvog, and S. Kumar, “History of search engines”, 4, vol. 15, 2011, pp. 47–58. DOI: <https://doi.org/10.19030/ijmis.v15i4.5799>.
- [13] Elasticsearch. “Our story”. (2022), [Online]. Available: <https://www.elastic.co/about/history-of-elasticsearch>. (accessed: 16.11.2022).
- [14] S. Banon. “The future of compass & elasticsearch”. (2022), [Online]. Available: [https://thedudeabides.com/articles/the\\_future\\_of\\_compass](https://thedudeabides.com/articles/the_future_of_compass). (accessed: 16.11.2022).
- [15] A. Andhavarapu, *Learning Elasticsearch*. Packt Publishing, 2017, ISBN: 9781787129917. [Online]. Available: <https://books.google.fi/books?id=2nc5DwAAQBAJ>.
- [16] Elasticsearch. “Data in: Documents and indices”. (2022), [Online]. Available: <https://www.elastic.co/guide/en/elasticsearch/reference/master/documents-indices.html>. (accessed: 30.11.2022).

- 
- [17] Elasticsearch. “5 reasons it leaders need vector search to improve search experiences”. (2022), [Online]. Available: <https://www.elastic.co/blog/why-technology-leaders-need-vector-search>. (accessed: 05.12.2022).
- [18] Elasticsearch. “Text similarity search with vector fields”. (2022), [Online]. Available: <https://www.elastic.co/blog/text-similarity-search-with-vectors-in-elasticsearch>. (accessed: 05.12.2022).
- [19] Wikipedia. “Cosine similarity”. (2022), [Online]. Available: [https://en.wikipedia.org/wiki/Cosine\\_similarity](https://en.wikipedia.org/wiki/Cosine_similarity). (accessed: 05.12.2022).

# Appendix A Code

---

**Listing 1** Function to split skill name into an array of skill names.

---

```
{javascript}

function splitName(name) {

  name = name.toLowerCase().replace("(", " ").replace(")", " ");

  let oldSplit = [name];

  let newSplit = [];

  const splitters = [",", "&", "/"]

  for (const splitter of splitters) {

    for (const n of oldSplit) {

      if (n.includes(splitter)) {

        newSplit = newSplit.concat(n.split(splitter));

      }

      else {

        newSplit = newSplit.concat(n);

      }

    }

    oldSplit = [...newSplit]

    newSplit = [];

  }

  return oldSplit.map(e => e.trim()).filter(e => e.length > 0)

}
```

---

---

**Listing 2** Function to separate skill name into multiple skills.

---

```
{javascript}

function separate() {
  const combined = JSON.parse(fs.readFileSync("./combined.json", "utf8"));
  const skills = JSON.parse(fs.readFileSync("./skills/result.json", "utf8"));
  const fixedSkills = [];
  const fixed = [];
  const skillsByName = {};
  const splittedById = {};
  for (const skill of skills) {
    const idx = skill.id;
    const splitted = splitName(skill.displayName);
    splittedById[idx] = splitted;
    for (const name of splitted) {
      const id = uuidv4()
      skillsByName[name] = id;
      fixedSkills.push({id: id, name: name})
    }
  }
  for (const userSkill of combined) {
    const splitted = splittedById[userSkill.skillId]
    if (splitted && splitted.length > 0) {
      for (const split of splitted) {
        const id = skillsByName[split];
        fixed.push({...userSkill, skillId: id, skill: split})
      }
    }
  }
  fs.writeFileSync("fixed.json", JSON.stringify(fixed), {encoding: "utf8"});
  fs.writeFileSync("fixedSkills.json", JSON.stringify(fixedSkills), {encoding: "utf8"});
}
```

---

# Appendix B Interview materials

Questions:

**Q1:** Do you think the clustering was a success or not and why?

**Q2:** Do you think that these lists contain similar skills?

**Q3:** Which one of these lists do you consider containing the most similar skills compared to the top most one?

**Q4:** Do you think the order in these lists is good?

**Q5:** Do you think that this is a good way to measure similarity?

**Q6:** Do you think that this can be used as a tool to find what those skills with no names could have been?



```

Cluster: 0, Size: 3473
Cluster: 1, Size: 67
Cluster: 2, Size: 2230
Cluster: 3, Size: 55
Cluster: 4, Size: 34
Cluster: 5, Size: 1036
Cluster: 6, Size: 44
Cluster: 7, Size: 54
Cluster: 8, Size: 53
Cluster: 9, Size: 42

```

First cluster

```

Index(['can do" attitude', '1', '10', '14', '17',
      '1st - 3rd line support for applications and workstations',
      '200 kantataulua', '2003', '2003 - 2016', '2007',
      ...,
      'ylläpito', 'yocto project', 'yup', 'zachman framework', 'zef',
      'zend framework', 'zephyr for jira',
      'zepto and other popular libraries', 'zoom', 'zustand'],
      dtype='object', name='skill', length=3473)

```

Second cluster

```

Index(['apple time machine', 'bpftrace', 'collectd', 'debian-installer',
      'dhcpd', 'dovecot', 'dtrace', 'embedded linux', 'gdb', 'gnu autotools',
      'hiera', 'hp procurve switches', 'ida pro', 'illumos zones',
      'infrastructure as code', 'ip networking', 'iptables', 'ipv6',
      'isc bind', 'juniper', 'junos switches', 'ldap', 'linux kvm',
      'linux mdraid', 'linux system administration', 'lvm', 'lxc', 'make',
      'mdb', 'meson', 'microsoft hyper-v', 'netfilter', 'nsd', 'ollydbg',
      'openbsd pf', 'openbsd vmm', 'openembedded', 'opensmtpd', 'packaging',
      'pam', 'pki', 'postfix', 'public key infrastructure', 'puppet', 'pxe',
      'receive based systems', 'rrdtool', 'solaris', 'systemd',
      'systemd-nspawn', 'tcpdump', 'traditional unix visibility tools',
      'unbound', 'unix programming', 'unnamedskill-1099', 'unnamedskill-601',
      'unnamedskill-767', 'unnamedskill-786', 'unnamedskill-803',
      'unnamedskill-835', 'unnamedskill-928', 'unnamedskill-996', 'wireshark',
      'xen', 'zabbix', 'zfs', 'zfs send'],
      dtype='object', name='skill')

```

Third cluster

```

Index(['+ browser dev tools', '-toimiston tuki', '10 mobile',
      '1st line support for cash registers',
      '1st line support for pda devices and mobile devices', '2', '2.0',
      '3-tier models', '3.5', '3d design',
      ...,
      'xpages', 'xrebel', 'y2k', 'yagni', 'yeesql', 'yeoman', 'yourkit',
      'zed attack proxy', 'zendesk', 'zkoss'],
      dtype='object', name='skill', length=2230)

```

## Fourth cluster

```
Index(['ad (active directory)', 'apache camel', 'apache jmeter', 'archi',
      'archimate', 'aris', 'arquillian', 'aws cloudfront',
      'aws serverless application model', 'business architecture', 'hudson',
      'iaf', 'ibm websphere mq', 'ide', 'information systems security',
      'integrated architecture framework', 'java application servers',
      'jax-rs', 'jhs 179 enterprise architecture planning and development',
      'lamda functions', 'libressl', 'mule esb', 'okta',
      'open source smart card tools and middleware', 'openmq', 'opensc',
      'openssl', 'qpr process designer', 'saml', 'single sign on', 'sso',
      'technology architecture', 'the grinder', 'togaf architecture framework',
      'tomcat', 'unnamedskill-1200', 'unnamedskill-1444', 'unnamedskill-1480',
      'unnamedskill-1510', 'unnamedskill-1530', 'unnamedskill-1561',
      'unnamedskill-1680', 'unnamedskill-1686', 'unnamedskill-1725',
      'unnamedskill-1729', 'unnamedskill-1763', 'unnamedskill-1802',
      'unnamedskill-212', 'unnamedskill-254', 'unnamedskill-30',
      'unnamedskill-52', 'unnamedskill-594', 'unnamedskill-812', 'weblogic',
      'websphere'],
      dtype='object', name='skill')
```

## Fifth cluster

```
Index(['business leadership', 'business models', 'business strategy',
      'compensations models', 'digital marketing', 'digital strategy',
      'financial reporting', 'internal tools', 'key account management',
      'kpi's', 'kpis', 'management reporting', 'okrs', 'processes',
      'product strategy', 'productization', 'public speaking', 'reporting',
      'resourcing', 'sales lifecycle', 'sales management', 'sales operations',
      'sales training and coaching', 'solution sales',
      'stakeholder management', 'strategic planning', 'strategy',
      'talent strategy', 'targets', 'team', 'team development',
      'user acquisition', 'user retention', 'vision crystallisation'],
      dtype='object', name='skill')
```

## Sixth cluster

```
Index(['2000', '2010', '2013', '3.11', '3d', '3d modeling', '3ds max', '7',
      '95', '98',
      ...,
      'xaml', 'xcode', 'xml', 'xml technologies', 'xp', 'xunit', 'yaml',
      'yarn', 'zbrush', 'zeplin'],
      dtype='object', name='skill', length=1036)
```

## Seventh cluster

```
Index(['cordoba', 'developer', 'embarcadero rad studio series', 'etc',
      'hybrid mobile', 'idm systems', 'intel xdk', 'java native mobile',
      'js frameworks', 'kiss', 'knockout', 'lamp', 'microsoft server', 'mim',
      'ms sql stack', 'oracle forms', 'patterns and principles',
      'quest software db tools', 'reports', 'require', 'solid',
      'support manager', 'ubisecure', 'unnamedskill-1064',
      'unnamedskill-1086', 'unnamedskill-1248', 'unnamedskill-1445',
      'unnamedskill-1465', 'unnamedskill-1497', 'unnamedskill-1517',
      'unnamedskill-1538', 'unnamedskill-1540', 'unnamedskill-1545',
      'unnamedskill-1606', 'unnamedskill-1610', 'unnamedskill-1613',
      'unnamedskill-1687', 'unnamedskill-1727', 'unnamedskill-1735',
      'unnamedskill-1747', 'unnamedskill-1790', 'vbox', 'wamp stack',
      'web servers'],
      dtype='object', name='skill')
```

Eight cluster

```
Index(['acquaia', 'acquaia cloud', 'acquaia cloud site factory', 'akamai',
      'amazon cloudsearch', 'apache solr', 'api integrations', 'backstop.js',
      'behat', 'bitbucket pipelines', 'bodybuilding',
      'client support and communications', 'composer', 'dev desktop',
      'documenting', 'drone', 'drupal 8', 'drupal 9', 'fitness',
      'front-end development', 'id', 'infrastructure management', 'js',
      'junior training', 'lando', 'migrate api', 'modern front-end',
      'module development', 'performance testing', 'php 7.4+',
      'planning and estimation', 'platform.sh', 'powerlifting',
      'regression testing', 'site building', 'teamwork', 'theme', 'trainee',
      'twig', 'unnamedskill-102', 'unnamedskill-114', 'unnamedskill-159',
      'unnamedskill-181', 'unnamedskill-205', 'unnamedskill-226',
      'unnamedskill-361', 'unnamedskill-395', 'unnamedskill-440',
      'unnamedskill-558', 'unnamedskill-622', 'varnish', 'virtual machines',
      'vue', 'wraith'],
      dtype='object', name='skill')
```

Ninth cluster

```
Index(['amazon elastic container service', 'api connect', 'api management',
      'apiops', 'app connect', 'architectural patterns',
      'architecture design', 'azure analysis services',
      'azure api management', 'azure data lake', 'azure iot hub',
      'azure network', 'azure storage', 'bapi', 'cloud foundry', 'compute',
      'containers', 'data', 'dell boomi', 'ecs', 'enterprise service bus',
      'financial management', 'integration',
      'iterative and incremental development', 'materials management',
      'microsoft visual studio team services', 'personality insights',
      'product development', 'purchase-to-pay', 'rds', 'recognize', 'rfc',
      'serverless', 'sql data warehouse', 'sqs queue', 'streams and events',
      'tone analyser', 'unnamedskill-1011', 'unnamedskill-1014',
      'unnamedskill-1040', 'unnamedskill-1073', 'unnamedskill-1078',
      'unnamedskill-1087', 'unnamedskill-46', 'unnamedskill-805',
      'unnamedskill-809', 'unnamedskill-820', 'unnamedskill-876',
      'unnamedskill-887', 'unnamedskill-976', 'vision - face', 'vpn', 'vsts'],
      dtype='object', name='skill')
```

Tenth cluster

```
Index(['azure app service', 'azure functions', 'azure key vault',
      'azure pipelines',
      'communication skills - successful in building strong co-operative relationships with key clients and decision-makers.',
      'company objectives.', 'crystal reports', 'ibm websphere studio 5.1.2',
      'microsoft access', 'microsoft visual basic .net',
      'microsoft visual sourcesafe', 'mks',
      'planning and organizing - proven track record of effectively prioritizing multiple tasks and assignments in a fast-pace
d work environment to efficiently meet departmental',
      'problem solving ability - regarded as a resourceful problem solver evident in the successful development and implementa
tion of new policies and procedures.',
      'rational rose', 'razor',
      'self-starter - considered a highly motivated employee with the capacity to learn quickly and take responsibility for my
own development.',
      'sql stored procedure', 'team work', 'unnamedskill-1179',
      'unnamedskill-1258', 'unnamedskill-1259', 'unnamedskill-1326',
      'unnamedskill-1413', 'unnamedskill-238', 'unnamedskill-244',
      'unnamedskill-250', 'unnamedskill-421', 'unnamedskill-462',
      'unnamedskill-476', 'vb 6', 'vb.net', 'vbscript', 'web api',
      'webmatrix', 'websites', 'wincvs', 'windows 7', 'windows 8',
      'windows nt', 'windows xp', 'winrunner'],
      dtype='object', name='skill')
```

|            |                  |
|------------|------------------|
| javascript | python           |
| node.js    | robot framework  |
| typescript | jenkins          |
| css        | django           |
| react.js   | pandas           |
| jquery     | selenium library |
| scss       | matlab           |
| sass       | machine learning |
| git        | scikit-learn     |
| vue.js     | numpy            |

|                  |                   |
|------------------|-------------------|
| java             | java              |
| spring boot      | spring framework  |
| kotlin           | spring boot       |
| spring framework | apache maven      |
| vaadin framework | junit             |
| junit            | hibernate         |
| hibernate        | mysql             |
| scala            | postgresql        |
| apache maven     | mariadb           |
| android studio   | javascript        |
| java             | java              |
| spring boot      | javascript        |
| jakarta ee       | git               |
| spring framework | python            |
| vaadin framework | css               |
| kotlin           | mysql             |
| guice            | spring framework  |
| junit            | postgresql        |
| hibernate        | mariadb           |
| spring portlets  | jenkins           |
| git              | git               |
| javascript       | svn               |
| java             | docker            |
| css              | subversion        |
| scrum            | gitlab            |
| python           | jenkins           |
| docker           | bitbucket         |
| jenkins          | gitlab ci         |
| node.js          | python            |
| sass             | unit testing      |
| git              | git               |
| docker           | bolt cms          |
| svn              | cybersecurity     |
| subversion       | aws architecting  |
| javascript       | symfony framework |
| java             | docker            |
| jenkins          | gherkin           |
| node.js          | svn               |
| typescript       | subversion        |
| linux            | embedded c        |

|                          |                  |
|--------------------------|------------------|
| scrum                    | mysql            |
| kanban                   | mariadb          |
| safe                     | postgresql       |
| agile                    | java             |
| atlassian jira           | mongodb          |
| agile project leadership | php              |
| sitefinity cms           | oracle           |
| agile project management | javascript       |
| scrum master             | apache tomcat    |
| lean                     | spring framework |

**skill**

|            |          |
|------------|----------|
| c          | 1.000000 |
| c++        | 0.591730 |
| qt         | 0.384074 |
| unix       | 0.327741 |
| linux      | 0.314773 |
| svn        | 0.291183 |
| subversion | 0.289168 |
| python     | 0.265980 |
| java       | 0.258093 |
| perl       | 0.236696 |

|                     |          |
|---------------------|----------|
| skill               |          |
| docker              | 1.000000 |
| kubernetes          | 0.619477 |
| postgresql          | 0.550793 |
| git                 | 0.544604 |
| jenkins             | 0.536666 |
| ansible             | 0.523036 |
| aws                 | 0.518207 |
| amazon web services | 0.516680 |
| terraform           | 0.507377 |
| gitlab ci           | 0.499299 |

|                       |          |
|-----------------------|----------|
| skill                 |          |
| aws                   | 1.000000 |
| amazon web services   | 0.999513 |
| docker                | 0.518207 |
| gcp                   | 0.481041 |
| google cloud platform | 0.462584 |
| postgresql            | 0.438887 |
| kubernetes            | 0.415993 |
| jenkins               | 0.394069 |
| azure                 | 0.390455 |
| aws dynamodb          | 0.390392 |

|                              |          |
|------------------------------|----------|
| skill                        |          |
| linux                        | 1.000000 |
| unix                         | 0.864532 |
| windows                      | 0.746124 |
| git                          | 0.429370 |
| os x                         | 0.422612 |
| bash                         | 0.421132 |
| subversion                   | 0.413837 |
| svn                          | 0.404941 |
| unix and linux shell scripts | 0.403668 |
| jenkins                      | 0.399135 |

|                 |          |
|-----------------|----------|
| skill           |          |
| unnamedskill-8  | 1.000000 |
| git             | 0.767190 |
| subversion      | 0.515306 |
| svn             | 0.500697 |
| unnamedskill-22 | 0.497560 |
| java            | 0.492231 |
| unnamedskill-10 | 0.481320 |
| unnamedskill-4  | 0.455975 |
| jenkins         | 0.453290 |
| unix            | 0.452429 |

|                  |          |
|------------------|----------|
| skill            |          |
| unnamedskill-85  | 1.000000 |
| mariadb          | 0.862804 |
| mysql            | 0.832578 |
| unnamedskill-4   | 0.615275 |
| unnamedskill-16  | 0.556666 |
| spring framework | 0.553494 |
| java             | 0.546869 |
| apache tomcat    | 0.526623 |
| apache maven     | 0.516044 |
| unnamedskill-49  | 0.515884 |