Identifying Roles of Software Developers from their Answers on Stack Overflow

Dean Power

Technological University Dublin

Follow this and additional works at: https://arrow.tudublin.ie/scschcomdis

Part of the Computer Engineering Commons, and the Computer Sciences Commons

Recommended Citation


This Dissertation is brought to you for free and open access by the School of Computer Sciences at ARROW@TU Dublin. It has been accepted for inclusion in Dissertations by an authorized administrator of ARROW@TU Dublin. For more information, please contact arrow.admin@tudublin.ie, aisling.coyne@tudublin.ie, gerard.connolly@tudublin.ie, vera.kilshaw@tudublin.ie.
Identifying Roles of Software Developers from their Answers on Stack Overflow

Dean Power

A dissertation submitted in partial fulfilment of the requirements of Technological University Dublin for the degree of M.Sc. in Computer Science (Data Science)

16th June 2021
Declaration

I certify that this dissertation which I now submit for examination for the award of M.Sc. in Computer Science (Data Science), is entirely my own work, has not been taken from the work of others and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of Technological University Dublin and has not been submitted in whole or part for an award in any other university or institute.

The work reported on in this dissertation conforms to the principles and requirements of the Institute’s guidelines for ethics in research.

Signed: 

Date: 16th June 2021
Abstract

Stack Overflow is the world’s largest community of software developers. Users ask and answer questions on various tagged topics of software development. The set of questions a site user answers is representative of their knowledge base, or “wheelhouse”. It is proposed that clustering users by their wheelhouse yields communities of similar software developers by skill-set. These communities represent the different roles within software development and could be used as the basis to define roles at any point in time in an ever-evolving landscape of software development. A network graph of site users, linked if they answered questions on the same topic, was created. Eight distinct communities were identified using the Louvain method. The modularity of this set of communities was 0.46, indicating the presence of community structure that is unlikely to occur randomly. This partition was validated with the results of previous research that used data from the same time period. By extracting the top 5 tags from each identified community, the harmonic F1-score between the communities and the external dataset was found to be 0.75. It was statistically proven with 95% confidence that the communities identified were not identical to the results from the previous research. Nonetheless, there exists a strong similarity to the previous research. Hence, it was suggested that Stack Overflow data could be used to identify and define roles within software development. Upon applying this method to 2021 data, a previously unknown community of experts in R, C and Rust was identified. The method used in this research could be applied directly to any of the 177 Stack Exchange sites and could be used to form the basis of job roles for a wide range of industries.

Keywords: Network Science, Stack Overflow
Acknowledgements

I would like to extend my gratitude to all of my lecturers at TU Dublin over the past two years, namely: Deirdre Lawless, David Leonard, Brendan Tierney, John Gilligan, Emma Murphy, Cathy Ennis, Luca Longo and Robert Ross. Each of you have contributed to this work through your teaching. A special thanks to Pierpaolo Dondio for both introducing me to network science and for your consultation in the early stages of this project.

I would also like to thank my supervisor Basel Magableh. Your guidance gave me the confidence in my ideas to proceed with them to completion.

To my family.
# Contents

Declaration .......................... I

Abstract ................................ II

Acknowledgments ......................... III

Contents ................................ IV

List of Figures ........................... VII

List of Tables ............................ IX

List of Variables ......................... X

1 Introduction ............................. 1
  1.1 Background .......................... 1
  1.2 Research Project/Problem .......... 2
  1.3 Research Objectives ................. 3
  1.4 Research Methodologies .......... 4
  1.5 Scope and Limitations .......... 5
  1.6 Document Outline .................... 6

2 Review of Existing Literature ...... 7
  2.1 Network Science ..................... 7
     2.1.1 Network Variants .......... 9
     2.1.2 Similarity Measures .... 10
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2 Problem Definition</td>
<td>53</td>
</tr>
<tr>
<td>5.3 Design/Experimentation, Evaluation &amp; Results</td>
<td>54</td>
</tr>
<tr>
<td>5.4 Contributions and Impact</td>
<td>55</td>
</tr>
<tr>
<td>5.5 Future Work &amp; Recommendations</td>
<td>55</td>
</tr>
</tbody>
</table>

References  

A Additional Content  

VI
## List of Figures

2.1 Graph of Facebook dataset (Danowski, 2012). ............................................. 8
2.2 An example of a bipartite network of users and tags, weighted by the number of questions answered. .......................................................... 9
2.3 The unipartite user-user projection of the weight bipartite network, with cosine similarity as edge weights. $u_1$ and $u_2$ are identical as they share the same tags and tag frequencies (Figure 2.2). ........................................... 11

3.1 Phases of the CRISP-DM Process Model for Data Mining (Wirth, 2000). 20
3.2 Cumulative percentage of questions by tag popularity. .............................. 28
3.3 Number of questions answered by users. .................................................. 28
3.4 Number of different tags answered by users. ............................................ 29
3.5 Distribution of percentage of total questions the user answered per tag, and how many total tags they answered. ............................................. 30
3.6 Ground-truth skills per software development role. ...................................... 32
3.7 An example of the Louvain Method (Barabási, 2015). ................................. 35

4.1 Modularity scores from 1000 bootstrapped samples of 1000 nodes from the Stack Overflow user network. Also plot (blue) is a theoretical normal distribution of the same mean and standard deviation. ......................... 40
4.2 Harmonic F1-scores to the ground-truth dataset from 1000 bootstrapped samples of 1000 nodes from the Stack Overflow user network. Also plot (blue) is a theoretical normal distribution of the same mean and standard deviation. ................................................................. 42
4.3 Quantile-quantile plot of bootstrap harmonic F1-score against theoretical normal distribution. ....................................................... 44

4.4 Stack Overflow user network graph (2018 data) with labelled communities (ForceAtlas2 in Gephi). Edges are not shown. .................. 46

4.5 Network projection of the identified Stack Overflow communities. . . . 48

4.6 Network projection of the ground-truth roles. ............................... 48

4.7 Stack Overflow user network graph (2021 data) with labelled communities (ForceAtlas2 in Gephi). ................................................. 50
List of Tables

3.1 Scrum sprints, stories and subtasks for this project. .................. 21
3.2 Sample of raw data from the Stack Exchange Data Explorer SQL query builder. .................................................. 26
3.3 Software development roles and associated skills taken from job postings (Gurcan & Sevik, 2019). This will form the ground-truth clustering for this study. .................................................. 31
3.4 Long-form dataset (example) prior to manipulation to edge list network format. .................................................. 33
3.5 Dataset (example) after manipulation to edge list network format. . . 34
4.1 Results from community detection using the Louvain Method. ...... 39
4.2 Stack Overflow Communities and their best-matching role in the ground-truth dataset by F1-score. ................................. 41
4.3 Details of communities and their best-matching ground-truth roles after filtering steps. .................................................. 42
4.4 Results from community detection between 1/1/2021 and 30/4/2021. . 50
4.5 A comparison between the communities identified in 2018 vs. 2021, and their best-matching roles. ................................. 52
List of Variables

\begin{itemize}
\item \( E \) \hspace{1cm} Number of edges in a network
\item \( F1h \) \hspace{1cm} Harmonic F1-score
\item \( Mod \) \hspace{1cm} Modularity of a network
\item \( n \) \hspace{1cm} A particular node in a network
\item \( N \) \hspace{1cm} Number of nodes in a network
\item \( \sigma_{n,n_j} \) \hspace{1cm} Smallest number of edges connecting two nodes in a network
\item \( q \) \hspace{1cm} A particular answered question on Stack Overflow
\item \( Q \) \hspace{1cm} A set of questions answered on Stack Overflow
\item \( SE \) \hspace{1cm} Standard error
\item \( t \) \hspace{1cm} A particular tag on Stack Overflow
\item \( T \) \hspace{1cm} The set of all tags on Stack Overflow
\item \( u \) \hspace{1cm} A particular Stack Overflow user who answered a question
\end{itemize}
Chapter 1

Introduction

1.1 Background

Stack Overflow is the world’s largest software development forum, where users answer each other’s questions relating to all things software development. As of May 2021, there are 31.5M answers to 21.2M questions about a myriad of topics in software development, from the usage of early technologies right through to the state-of-the-art tools and techniques.

A typical scenario exists in software development where a professional has a gap in their knowledge about a particular topic or problem and requires support or input from their peers when attempting to form a solution. The professional can post a question on Stack Overflow and tag the question with keywords that help other users find and answer their question. Other Stack Overflow users then contribute their answers and the user can up-vote, reply and/or accept them.

As questions and answers are time-stamped, Stack Overflow is a temporal representation of the software development industry. At any point since its creation in 2008, questions are tagged with the relevant topics and technologies. Site users can search for and browse through unanswered questions by tag. The tagging system is an effective method of connecting potential answerers to the questions they know about. Users should only be able to answer a question if they have knowledge about that particular area of software development represented by the tag. Therefore, the set of
CHAPTER 1. INTRODUCTION

tags answered by each user should represent their “wheelhouse”, or area of expertise. It should follow that if a user answers more questions on a particular topic, they have a greater depth of knowledge on that topic than other topics they answer questions on less often. Hence, each user has a “fingerprint” in tag-space that is characteristic of their area of expertise in software development. By grouping similar “fingerprints” together, distinct roles in software development could be emergent.

One would speculate the quality of content in such an anonymous online forum. However, the motivation to answer questions with high quality comes from a gamified system of earned badges, up/down-voting and question closing, along with the natural satisfaction of sharing one’s expertise with the world. Similar systems have seen success on other sites such as Reddit, where users self-regulate the quality of content through a system of up-voting and down-voting. Regardless of the quality of individual questions and answers, the sheer volume of material posted ensures that the site is indicative of current trends in software development.

1.2 Research Project/Problem

Software development is ever-changing. New technology often comes in waves of hype, adoption and/or dismissal (Strawn, 2021). With this comes a turbulent landscape to navigate. Once an attempt is made to understand the different areas or roles in the software development industry, a paradigm shift occurs and radically alters the space. From the perspective of industry, companies must make a conscious effort to keep up to date with the latest trends in technology or risk becoming obsolete, with potential financial implications. An employer may have a requirement to fill a particular software developer role, but they may be out of touch with current trends in tools and technology professionals in that role may use. Conversely, the employer may require a specialist in particular technologies, but may not be able to adequately define the title of the role they are looking to fill. With an ever-changing technology landscape, there is a disparity between software development roles and the topics/tools they span. Turning to opinion-based articles found on the internet may offer some
clarity, but may be out of date.

This can also pose a challenge for educators when designing curricula that best suit the needs of industry. Although some educators accept that the material taught in a current curriculum may be obsolete (Hewitt, 2006), others require their material to be at the cutting-edge. This is especially important for computer science, data science and other information technology-related courses. Four years to complete a degree accommodates a lot of change in the industry.

However turbulent the software development landscape may be, contained within Stack Overflow data is a historically accurate proxy for the relative popularity of trends. It is proposed that a set of distinct software developer roles can be identified from Stack Overflow at any point in time, with an industry sample size large enough for the roles identified to act as the “gold-standard” definition of roles within software development.

1.3 Research Objectives

The primary objective of this research project is to show that Stack Overflow can be used as a way to identify and define distinct roles in software development. To achieve this, several secondary objectives must also be met:

- To collect, store and cleanse Stack Overflow data.
- To manipulate the data into a format that is suitable for unsupervised clustering.
- To perform unsupervised clustering on the data such that software development roles are emergent and reproducible.
- To statistically assess the quality of the identified clusters.

These objectives are laid out in detail in a proposed project timeline in Table 3.1 (page 21).
1.4 Research Methodologies

This project was structured according to the principles outlined in the CRoss-Industry Standard Process for Data Mining (CRISP-DM) and Scrum. These are explained in more detail in section 3.1 (page 19). These methodologies were chosen as they suit an iterative, code-heavy project such as this. They are designed to support continuous improvement through iterative development, evaluation and readjustment of goals. Jira software was used to track the progress of each sprint as per the Scrum methodology.

The type of research undertaken in this project is secondary data analysis, as the data had already been collected by Stack Overflow. The objective of this research is quantitative, as there is a definite hypothesis that can be tested through statistical analysis. The form of this research is primarily empirical as the primary objective of this project is to quantitatively form a new theory about software development roles, and, by inductive reasoning, their emergence from the behaviours of Stack Overflow users. However, if it is not assumed that the ground-truth is unequivocally true, then this research takes a more constructive form which does not require the strict acceptance of the hypothesis through a statistical test. If one assumes the ground-truth dataset to be potentially true and the experimental result of this project is in general agreement, one could argue that the experimental result could also be potentially true. Regardless, any claims made around the formation of new theory regarding emergent roles in software development will be thoroughly examined and caveated in chapter 4 (page 38).

Although there were multiple techniques with which to perform this analysis, network science was chosen as the main technique. Unsupervised clustering, when performed on a network, is called community detection. The Louvain Method was chosen as the community detection algorithm for this project.

---

1https://deanpower.atlassian.net/secure/RapidBoard.jspa?projectKey=MP
1.5 Scope and Limitations

There are a number of fundamental assumptions and limitations to this research project.

- This project uses network community detection, an unsupervised clustering method. To externally evaluate unsupervised methods, a “ground-truth” is required for comparison. This is a source of information that is taken to be true, and the results of the unsupervised method are compared to the ground-truth using a specified metric. If a ground-truth cannot be identified for the problem, any experimental result cannot be proven to be true. A further assumption that must be taken is that the ground-truth is actually true; if the source of truth is from another experiment, the results of that experiment must be taken to be true.

- It is assumed that the distribution of answers from Stack Overflow users is an accurate representation of their wheelhouse of skills. The assumption that their answers reflect their entire wheelhouse is not required; any missing skill in a user’s wheelhouse would be statistical noise given the large sample size.

- It is assumed that the relative frequency of topics answered by a user is a proxy to their relative knowledge in those topics. This is a significant assumption if a user’s answer count is low.

- It is assumed that the ground-truth will represent a particular period in software development.

- If the ground-truth identified comes from a particular period and it is proven that the unsupervised community detection method returns a result equivalent to the ground-truth within a statistical tolerance using Stack Overflow data from that period, it is assumed that the unsupervised method returns a true result at any point in time using data from any period.
• There exists a trade-off when choosing the length of the period to look at. If the period is too small, the sample size will be too small to provide a statistically representative result. If the period is too large, there may be significant paradigm shifts in software development during that time which will be blended together.

• It is assumed that all users of Stack Overflow are representative of the software development industry. This may not be the case with some test or throw-away accounts that may be present in the data.

These assumptions and limitations were kept in mind throughout the project, from the design steps through to conclusions.

1.6 Document Outline

This project begins with a literature review on the topics of network science, Stack Overflow and roles within software development in chapter 2 (page 7).

A description of the experiments performed, their motivations and methods of evaluation are outlined in chapter 3 (page 19).

The results of the experiments and a discussion around each finding is available in chapter 4 (page 38).

This project concludes with chapter 5 (page 53) which summarises the key findings and outlines future work.
Chapter 2

Review of Existing Literature

This literature review is divided into three main parts; Network Science, Stack Overflow and Roles of Software Developers.

- The network Science section will introduce the reader to the main concepts of networks and network analysis before exploring recent advancements in the areas of network similarity measures, community detection algorithms and community evaluation methods.

- The Stack Overflow section will outline the main areas of research on the popular software development forum.

- The Roles of Software Developers section will outline previous research conducted that relates to the identification of roles within software development.

2.1 Network Science

To preface this section, most general information about network science (unless specified otherwise) comes from Barabási (2015), regarded by many as the “gold standard” textbook on network science.

Network science is the study of relationships between entities as part of the whole. The internet or social media might come to mind when one thinks about networks.
CHAPTER 2. REVIEW OF EXISTING LITERATURE

Networks, in the traditional sense, consist of entities called nodes connected pairwise by edges which represent any type of relationship that may exist between the nodes. Taking Facebook as an example; a conceivable network would have users as nodes and friendships as edges. It is possible to draw a graph of this network to help visualise the user’s relationships (Figure 2.1).

![Figure 2.1: Graph of Facebook dataset (Danowski, 2012).](image)

Network science enriches this view with a multitude of properties about how the network behaves as a whole. What is the average number of friends? How many degrees of separation are two typical users apart? Does the network form communities such as different political stances or professions? Are they distinct, polarised groups or is there considerable overlap in these communities? Network science, particularly social network analysis, enables these relatively complex questions about social dynamics to be answered which could not be easily done using traditional tabular data analysis methods.

One such example of a useful network science measure is betweenness centrality (Freeman, 1977). The betweenness centrality $b$ of a node $n$ in a network is the proportion of “shortest paths” $\sigma$ between all pairs of nodes that pass through that node.

$$b(n) = \sum_{m \neq n \neq o} \frac{\sigma_{mo}(n)}{\sigma_{mo}}$$  \hspace{1cm} (2.1)
Nodes with a higher betweenness centrality tend to lie on the boundaries between communities as a bridge and are important in the effective flow of information from one community to another.

### 2.1.1 Network Variants

Nodes can have more than one edge connecting them to another node. This is usually represented as a single edge with a weight. Nodes can also be of more than one type, with edges connecting nodes of different types. This is called a bipartite network.

Consider a Stack Overflow user $u$ who answers $Q^u_t$ questions containing tag $t$. The relationship between the users and the tags can be visualised as a weighted bipartite network (Figure 2.2). The user nodes are connected to the tag nodes by the number of questions the user answered with that tag.

![Figure 2.2: An example of a bipartite network of users and tags, weighted by the number of questions answered.](image)

Bipartite networks have two unipartite projections, a user-user projection and a tag-tag projection in this example. As we are interested in finding communities of users, the user-user projection is the relevant projection. If the bipartite network were
unweighted, the edge weights of the resulting projections would be a straightforward sum of the number of different tags per user and users per tag respectively. However, the problem becomes more involved as the bipartite network in Figure 2.2 is already weighted and so a simple sum of unique tags per user will not retain all information about the network. Therefore, a similarity measure must be derived between users that captures both the specific tags answered and the frequency of those tags.

In general, it is possible for a network edge to connect more than two nodes together. N-way hypergraphs can be used to capture non-binary relationships between entities. Hypergraphs pose their challenges both from a mathematical perspective and through difficulty in visualisation and will not be considered for this project, although this is an area of great interest and promise (Benson, Gleich, & Higham, 2021; Higham & de Kergorlay, 2021).

2.1.2 Similarity Measures

When comparing users based on their tag wheelhouse, it is important to take their frequency of tag use into account. For example, if a user answers 10 questions on Python and 1 on Java, it could be said that the user is more comfortable answering questions on (and therefore has a greater depth of knowledge on) Python than Java. In the example, although user \( u_2 \) shares the same tags as \( u_1 \) and \( u_3 \), \( u_2 \) is more similar to \( u_1 \) as their tag frequencies are more similar. Each user can be represented as a vector \( \vec{Q}_u \) in \( T \)-dimensional tag-space, where \( T \) is the total number of different tags available. This allows pairwise user similarity to be quantified with a vector similarity measure. Common similarity measures include Jaccard similarity (Jaccard, 1912) and the more commonly used cosine similarity. Cosine similarity has applications in document matching, where vectors of word/term frequencies are compared (M. Mohammed, Jacksi, & R. M. Zeebaree, 2021). The cosine similarity of two vectors is simply the cosine of their angle, which ranges from 0 (completely dissimilar, vectors are orthogonal) to 1 (completely similar, vectors are parallel):
CHAPTER 2. REVIEW OF EXISTING LITERATURE

\[
similarity(Q^u_i, Q^u_j) = \cos(\theta) = \frac{Q^u_i \cdot Q^u_j}{\|Q^u_i\| \|Q^u_j\|} = \frac{\sum_{t=1}^{T} Q^u_i t Q^u_j t}{\sqrt{\sum_{t=1}^{T} Q^u_i t^2} \sqrt{\sum_{t=1}^{T} Q^u_j t^2}} \tag{2.2}
\]

The cosine similarity is computed for every pair of users in the network. This can then be used as the edge weight in a user-user network, as done by Cao, Bu, Gao, and Tao (2016) who used cosine similarity as edge weights for a modularity maximisation community detection algorithm. Chessa, Crimaldi, Riccaboni, and Trapin (2014) also used cosine similarity for the edge weights in a bipartite network prior to community detection. For our example, \( u_1 \) and \( u_2 \) have a cosine similarity of 1 as they share the same tags and tag frequencies, while \( u_3 \) has a cosine similarity of 0.824 to the other two users - quite a high score as they share the same tags but different tag frequencies. The unipartite user-user projection can now be created (Figure 2.3).

Figure 2.3: The unipartite user-user projection of the weight bipartite network, with cosine similarity as edge weights. \( u_1 \) and \( u_2 \) are identical as they share the same tags and tag frequencies (Figure 2.2).
2.1.3 Community Detection in Networks

Two main approaches to community detection in networks have been developed: bottom-up and top-down. Bottom-up approaches begin with every node as its own community and agglomerates into fewer, larger communities, while top-down approaches begin with a single community which divides into increasingly smaller communities.

Farkas, Ábel, Palla, and Vicsek (2007) proposed a bottom-up community detection algorithm, k-clique percolation with weights. Communities are identified as a series of overlapping k-node cliques. Edge weights can be considered, and communities can be overlapping. K-clique percolation is relatively fast for large networks with complexity $O(N)$, where $N$ is the number of nodes in the network. A challenge with this method is the optimisation of $K$, the clique size, especially if the expected number of clusters is not known.

Another bottom-up community detection algorithm was proposed by Blondel, Guillaume, Lambiotte, and Lefebvre (2008), a modularity maximization method called the Louvain Method. Each node must be assigned to exactly one community, edge weights can be considered and no parameters need optimisation. The Louvain method is a greedy algorithm and complexity scales linearly with the number of communities left after each iteration. It is also the most widely used community detection method today and is the default method in network analysis software Gephi (Bastian, Heymann, & Jacomy, 2009).

Ramalingeswara Rao, Ghosh, and Goswami (2020) proposed a method to detect user-user communities from a weighted bipartite network through projection using a novel similarity measure and a bottom-up community detection algorithm called label propagation. This research could be applied directly to Stack Overflow data but requires a bespoke distributed architecture. This project aims to achieve the same result with a significantly less complex implementation.

Top-down network community detection approaches were proposed, such as Weighted Community Clustering (Prat-Pérez, Dominguez-Sal, Brunat, & Larriba-Pey, 2012) and an overlapping community detection algorithm by D. Chen, Shang, Lv, and Fu
(2010). However, top-down approaches seem to scale exponentially with network size (O(E^2) and O(N^2) respectively, where E is the number of edges in the network), deeming them impractical for large networks such as the Stack Overflow user network in this project.

Community detection in hypergraphs is currently an emerging field of study (Kamiński, Prałat, & Théberge, 2021), but will not be used for this project.

2.1.4 Evaluating Communities

Community detection is an unsupervised learning problem, meaning community detection algorithms do not refer to pre-defined community labels but rather create communities from other network attributes. This creates an issue in evaluating the performance of the algorithm; How can it be proven that the detected communities represent real-world communities? Community detection algorithms can be evaluated in two ways, internally and externally.

**Internal Evaluation**

Internal evaluation evaluates the quality of communities based on their structural properties alone, without comparison to labels or a ground-truth set of communities. The most commonly used internal measure is modularity (Newman & Girvan, 2004). Characteristics of networks with high modularity are densely connected nodes within communities and sparsely connected between communities.

Modularity is defined as:

\[
Mod = \frac{1}{2E} \sum_{n_i,n_j} \left[ A_{n_i,n_j} - \frac{k_{n_i}k_{n_j}}{2E} \right] \delta(c_{n_i}, c_{n_j})
\] (2.3)

Where \( E \) is the number of edges in the network, \( \frac{k_{n_i}k_{n_j}}{2E} \) is the probability of an edge existing between nodes \( n_i \) and \( n_j \) of degree \( k_{n_i} \) and \( k_{n_j} \) respectively, and \( A_{n_i,n_j} = 1 \) if an edge exists between the nodes and 0 otherwise. \( \delta(c_{n_i}, c_{n_j}) \) is the Kronecker Delta and is 1 if \( n_i \) and \( n_j \) are members of the same community and 0 otherwise. Random networks should have a modularity of 0. Perfect community structure has modularity
equal to 1. It is also possible for modularity to be negative if the communities are worse than random.

The current state of the art (Chakraborty, Dalmia, Mukherjee, & Ganguly, 2017) internal evaluation measure for overlapping communities is the extended modularity density proposed by M. Chen, Kuzmin, and Szymanski (2014). The communities of Stack Overflow users are not expected to overlap, however, and will require neither an overlapping community detection algorithm nor an internal evaluation measure that can handle overlap.

**External Evaluation**

External evaluation compares the communities to a ground-truth set of communities. This is necessary with any unsupervised learning method as the communities have no intrinsic meaning without comparison to a reference set of communities.

One particularly useful external evaluation measure is the average F1-score between the communities identified and their matching ground-truth set. However, the resultant communities from unsupervised clustering methods are unlabelled. This poses a challenge when using the F1-score as the communities must be matched up. As a remedy, the communities can be paired with their best-matching ground-truth by simple F1-score, and vice-versa. The average of both sets of best matches (communities to ground-truth and ground-truth to communities) can be taken.

To elaborate on the above points:

- The harmonic F1-score is the harmonic mean of a) the weighted F1-scores of the best matching ground-truth communities to the formed clusters and b) the best matching formed clusters to the ground-truth communities (Lutov, Khayati, & Cudré-Mauroux, 2019).

- In the context of comparing a cluster of tags to skills in a ground-truth role, precision is the number of correctly classified tags divided by the number of classified tags (Manning, Raghavan, & Schutze, 2008). Using set notation:
\[
\text{Precision} = \frac{|\{\text{classified}\} \cap \{\text{ground-truth roles}\}|}{|\{\text{classified}\}|}\quad (2.4)
\]

- Recall is the number of correctly classified tags divided by the number of tags that should have been classified:

\[
\text{Recall} = \frac{|\{\text{classified}\} \cap \{\text{ground-truth roles}\}|}{|\{\text{ground-truth roles}\}|}\quad (2.5)
\]

- The F1-score is defined as the harmonic mean of precision and recall if giving equal importance to both:

\[
F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}\quad (2.6)
\]

- Therefore, given the set of identified communities \(c_i \in C\) and ground-truth roles \(c'_i \in C'\), the harmonic F1-score is defined as:

\[
F_{1h}(C', C) = 2 \cdot \frac{F_{C', C} \cdot F_{C, C'}}{F_{C', C} + F_{C, C'}},\quad (2.7)
\]

where

\[
F_{X,Y} = \frac{1}{|X|} \sum_{x_i \in X} F_1(x_i, g(x_i, Y)),\quad (2.8)
\]

and

\[
g(x, Y) = \{\text{argmax}_y F_1(x, y) | y \in Y\}.\quad (2.9)
\]

### 2.2 Stack Overflow

General research on Stack Overflow has mainly focused on:

2. **Question/answer quality** (Beyer et al., 2020; Cagnoni et al., 2020; S. Wang, Chen, & Hassan, 2018; Chatterjee et al., 2020; Chua & Banerjee, 2015; Campos & De Almeida Maia, 2014)

3. **Code snippet detection and analysis** (Cagnoni et al., 2020; Chatterjee et al., 2020; Wu et al., 2019; Odiete et al., 2017)

4. **Topic modelling** (Beyer et al., 2020; H. Chen, Coogle, & Damevski, 2019; Zou, Xu, Yang, Zhang, & Yang, 2017; Joorabchi, English, & Mahdi, 2015, 2016; Barua et al., 2014)

Looking at user behaviour, the focus of research has mainly been on finding topic experts. Anusha, Rekha, and Sivakumar (2015) used X-means (Pelleg & Moore, 2000) and expectation maximisation (Dempster, Laird, & Rubin, 1977) clustering algorithms to cluster Stack Overflow users into four groups: “naive users”, “surpassing users”, “experts”, and “outshiners”. The study made use of various user metrics on the site.

(C. Chen & Xing, 2016) created a knowledge graph of co-occurring tags on Stack Overflow questions. This approach yielded a network of tags linked if they appeared on the same question. This can be interpreted as a tag synonymity network as Stack Overflow questions should be focused and contain a question about a single technology. The study did not consider linking tags by their use by users. Furthermore, technology communities were successfully identified using the Louvain method (Blondel et al., 2008). A similar approach was applied to other Stack Exchange sites by Fu, Yu, and Benson (2021), highlighting the transferability of Stack Overflow analyses to other areas and professions.

The annual Stack Overflow Developer’s Survey\(^1\) is a good reference of aggregated data from the site and survey results of users, outlining current industry trends and software development roles. It does not break down roles by their top skills but is a good reference nonetheless.

---

\(^1\)https://insights.stackoverflow.com/survey/2020
2.3 Roles of Software Developers

Identifying top skills of technology roles has been previously investigated by a few studies. Aken, Litecky, Ahmad, and Nelson (2010) scraped nearly 250,000 computing job advertisements from various sites and used text analysis to extract the most common skills. K-means clustering (Lloyd, 1982) was performed on these skills and 20 clusters were identified as having the lowest average mean centroid distance per cluster. The clusters were manually labelled with their roles.

Gurcan and Kose (2017) used latent Dirichlet allocation (LDA) (Blei, Ng, & Jordan, 2003), a natural language processing technique, to identify 20 main roles for software engineers and their core skills and responsibilities from postings on the Stack Overflow Careers site\(^2\). Their motivation was to allow for an innovative academic curriculum for software engineering education to be designed consistent with the emerging needs and trends in the software industry.

Gurcan and Sevik (2019) also used LDA in their approach to identify 20 main roles for software development, listing 5 in-demand skills for each role. The dataset was collected from 1,385 job postings related to software development from Indeed\(^3\), a popular job search site, from April to August 2018. This is the most up to date and relevant study of skills required by the software development industry, and may be suitable for use as a ground-truth dataset for this project.

Research on identifying software development roles has entirely focused on scraping job postings from websites and using text analysis to extract the most common skills required in the posting. No known attempt has been made to leverage Stack Overflow user data of any kind to infer software development roles.

2.4 Summary of Literature Review

Through comprehensive literature review of techniques in network science, research utilising Stack Overflow data and identification of roles within software development,
a number of conclusions were reached.

- Data science approaches to defining software development roles have been limited to textual analysis of job postings.

- Stack Overflow is an under-utilised resource of rich information about the behaviours and interactions of a large sample of the software development community.

- Network science is a growing field of research with a multitude of applications. One such application is unsupervised community detection. Results from adopting a network science-based approach are comparable to results obtained through classical machine learning techniques. Community detection is applicable to the problem of clustering Stack Overflow users by the questions they answer, revealing their areas of expertise as a proxy to their particular role as software developers. To achieve this without loss of information, a weighted user-user network must be created, with a tag-based user-user similarity measure as weights. A commonly used similarity measure that delivers state-of-the-art results is the cosine similarity.

- A fast and commonly implemented community detection method for weighted networks is the Louvain Method. To evaluate communities, a convenient external measure is the harmonic F1-score as the communities do not need to be matched up to the ground-truth communities.

- The most universally accepted internal community evaluation measure is modularity.

- A dataset was identified as suitable to be used as ground-truth in this project (Gurcan & Sevik, 2019) that contains a list of 20 distinct roles and the top 5 skills in each role. Detected communities could be compared to this dataset to assess their quality.
Chapter 3

Experiment Design and Methodology

This chapter details the design of this project, from the overall methodology to the specific tools and techniques used. Some derivations are also outlined in this chapter that build on concepts introduced in the literature review.

3.1 Project Structure

This project adhered to the CRoss-Industry Standard Process for Data Mining (CRISP-DM) (Wirth, 2000) and Scrum\textsuperscript{1}. The sections of this chapter reflect the stages of these methodologies.

CRISP-DM is the most widely used process for data mining projects as it is independent of both industry and technology used by design. It outlines the logical order in which a data science project should be conducted, and which steps inform other steps. It is a process that cycles until the result satisfies the business objective. CRISP-DM can be modified for use in data science research projects by replacing the Business Understanding step with a comprehensive literature review (chapter 2, page 7) and definition of a research question.

Scrum is an iterative framework that defines two-week sprints of work at a time.

\textsuperscript{1}https://www.scrum.org/resources/what-is-scrum
At the end of each sprint, the state of the project is evaluated and the next sprint is planned. In this manner, the trajectory of the project can be constantly re-adjusted as needed. It is based on the idea of producing a "minimum viable product" in the shortest possible time, with additional sprints adding further complexity and features. Jira\(^2\) is a software product from Atlassian that helps manage Scrum projects. This project was managed on Jira and was divided into 10 sprints of 2 weeks (Table 3.1). Once a task was completed it was marked as complete. The sprint was marked as complete once all tasks in that sprint were complete, and the next sprint would then commence.

\(^2\)https://www.atlassian.com/software/jira
Table 3.1: Scrum sprints, stories and subtasks for this project.

<table>
<thead>
<tr>
<th>Sprint</th>
<th>Story</th>
<th>Subtask</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Environment setup</td>
<td>Create local file storage system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set up R development environment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Integrate Git versioning</td>
</tr>
<tr>
<td></td>
<td>Data collection</td>
<td>Query Stack Exchange Data Explorer for the required data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Store queried data in local file system</td>
</tr>
<tr>
<td>2</td>
<td>Data understanding</td>
<td>Load data into R and perform initial exploratory analysis</td>
</tr>
<tr>
<td></td>
<td>Data pre-processing</td>
<td>Data cleansing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mutate data into required network format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compute network parameters and perform exploratory analysis</td>
</tr>
<tr>
<td>3</td>
<td>Data modelling</td>
<td>Community detection on bootstrapped samples of the network</td>
</tr>
<tr>
<td>4</td>
<td>Model evaluation</td>
<td>Evaluate community detection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Perform t-test on evaluation measures</td>
</tr>
<tr>
<td>5</td>
<td>Deployment</td>
<td>Display detected software development roles</td>
</tr>
<tr>
<td>6-10</td>
<td>Report writing</td>
<td>Report writing</td>
</tr>
</tbody>
</table>

3.2 Problem Understanding

The goal of this project is to find a set of software development roles and their most relevant skills from the questions answered by users on Stack Overflow. When viewed through a data science lens, this is a typical example of unsupervised machine learning. Specifically, the goal is to assign labels (roles) to a set of unlabelled data points (Stack Overflow users) based on the attributes of the data points (questions they answer). This is called unsupervised clustering. Data with similar attributes are more likely to be assigned to the same cluster. There are multiple approaches to this result through the use of data science techniques. This section compares approaches through classical machine learning techniques to network science approaches.
3.2.1 Classical Machine Learning Approaches

When one considers unsupervised clustering techniques, k-means clustering (MacQueen, 1967) always makes the list. K-means is a centroid-based clustering algorithm, which assigns data points to one of \(k\) clusters based on a distance measure, usually the squared Euclidean distance to heavily penalise far distances to the centroids. Although the principle behind k-means clustering is straightforward, the problem of choosing \(k\) remains. This is not trivial if there is no expected number of clusters, as is the case with this problem. Various techniques for optimising \(k\) exist, such as the elbow method (Thorndike, 1953). K-means can be extended to a fuzzy clustering algorithm where the probability of being in each cluster is returned (Bezdek, 1981). K-means is by far the most popular classical approach to unsupervised geometric clustering. As this problem does not require hierarchical clustering, other classical approaches that return hierarchical clusters will not be discussed. Another popular unsupervised clustering approach is to use deep learning. Recent advancements in using deep learning to cluster data have outperformed classical approaches (Zhang & Qian, 2021).

3.2.2 Network Science Approaches

Although not exhaustive, three specific approaches to this problem via network science were identified. Depending on the format of the resulting network to be clustered, either an overlapping or non-overlapping community detection algorithm should be used. Various community detection algorithms were outlined in subsection 2.1.3 (page 12).

- The first approach creates a network of tags. Edges are weighted by the number of users that answer questions with both tags. Communities are then found in this network which represent software development roles. This approach is simple to conceptualise as the skills within the roles are simply the nodes in the community. The main issue with this approach is that the communities of tags would be highly overlapping as tags are not exclusive to a single software development role. Another issue would be the loss of information as each user’s contribution is spread over multiple different edges. Overlapping communities
are both difficult to detect and evaluate.

- The second approach creates a hypergraph of tags. Edges in this case are weighted by entire user wheelhouses as edges are no longer just pairwise node connections, but connect any number of nodes together. This method retains all information about user wheelhouses as they are not flattened to a pairwise representation. However, research on community detection in hypergraphs has only recently emerged (Kamiński et al., 2021) and would be difficult to implement in practice.

- The third approach creates a network of users. Edges in this case are weighted by the number of questions with a particular tag answered by both users. Communities are then found in this user-user network which represent software development roles. These communities will not overlap as each user should fall into a single software development role. The main issue with this approach is extracting the main skills for each role, as this information is not contained in the network properties but rather is a property of the nodes themselves - the user’s tag wheelhouse. A strategy for extracting the most relevant tags per community must be implemented, such as the top $n$ tags by answer count per community.

The third approach was chosen for this project by process of elimination. The first approach was investigated as it would not require the extraction of the top tags per community. However, it was found that it was not possible to detect communities in this format as the communities overlapped to such an extent that no communities were emergent using a clique percolation algorithm for detection of overlapping communities. Little was known about the implementation of the second approach and was deemed unnecessarily complex for this project. No visualisations of the network would be possible with the second approach.
3.3 Data Understanding

A crucial step in the data science process is thoroughly understanding the data source structure and quality. This section chronologically presents the steps taken from data sourcing to a cleansed data set that is suitable for pre-model feature selection and manipulation steps.

3.3.1 Data Source

Stack Overflow data are available through the Stack Exchange Data Explorer\(^3\), a SQL database of data from all sites in the Stack Exchange ecosystem. The data are stored in intuitively named tables associated with the category of data contained within, e.g. Posts. Each table has a unique primary key.

The primary tags on questions answered by users are required for this study. This requires data related to questions, answers, tags and users. Care must be taken when constructing the SQL query such that all table joins are made on primary keys to avoid many-to-many relationships. The time frame of data collected in the ground-truth study by Gurcan and Sevik (2019) was between April and August 2018, so answers should also be limited to this period.

Data were collected under the assumptions of quality outlined in section 1.5 (page 5).

3.3.2 Hypothesis

At this point, it is necessary to define a formal hypothesis to test through experiment. Definition of this hypothesis will inform subsequent steps such as data collection, modelling and evaluation methods. The hypothesis as defined was formulated with the knowledge gained from the identification of a research problem (section 1.2, page 2) and subsequent literature review (chapter 2, page 7). The evaluation techniques used in section 3.6 (page 36) must directly reject or fail to reject the null hypothesis.

\(^3\)https://data.stackexchange.com/stackoverflow/query/new
**H0:** Upon attempting to cluster a weighted network of Stack Overflow users (linked if the users have authored accepted answers on questions tagged with the same primary tag, and weighted by the cosine similarity of such users) using the Louvain Method, the harmonic F1-score between the clusters and the assumed ground-truth roles identified in Gurcan and Sevik (2019) is not statistically significantly equal to 1.

**HA:** Upon attempting to cluster a weighted network of Stack Overflow users (linked if the users have authored accepted answers on questions tagged with the same primary tag, and weighted by the cosine similarity of such users) using the Louvain Method, the harmonic F1-score between the clusters and the assumed ground-truth of the groups identified in Gurcan and Sevik (2019) is statistically significantly equal to 1. Hence, these clusters can form the basis for role names in software development.

### 3.3.3 Data Collection

The Stack Exchange Data Explorer interface limits the answer set to 50,000 rows returned, which equates to just over 1 month of user and tag data. Therefore, it was necessary to form 5 separate monthly SQL queries to fetch the data, which will be concatenated during data processing. A sample of the raw output from this step is shown in Table 3.2. Id is the answer Id, ParentId is the question Id, OwnerUserId is the Id of the answerer and Tags are the tags on the question ordered by popularity across the site at the time. Answer date was not returned as a field; it is irrelevant information as the query returned all answers from April and August 2018, which is sufficiently granular for this project.

The SQL query used to construct the dataset is available in Listing A.1 (page 64).
Table 3.2: Sample of raw data from the Stack Exchange Data Explorer SQL query builder.

<table>
<thead>
<tr>
<th>Id</th>
<th>ParentId</th>
<th>OwnerUserId</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>49594373</td>
<td>49594336</td>
<td>21009</td>
<td><code>&lt;qt&gt;&lt;qml&gt;&lt;qtquickcontrols2&gt;</code></td>
</tr>
<tr>
<td>49594398</td>
<td>49594368</td>
<td>9450991</td>
<td><code>&lt;python&gt;&lt;python-3.x&gt;&lt;numpy&gt;</code></td>
</tr>
<tr>
<td>49594404</td>
<td>49594360</td>
<td>711206</td>
<td><code>&lt;php&gt;</code></td>
</tr>
<tr>
<td>49594405</td>
<td>49594361</td>
<td>1953590</td>
<td><code>&lt;java&gt;&lt;overloading&gt;</code></td>
</tr>
<tr>
<td>49594438</td>
<td>49594421</td>
<td>9147721</td>
<td><code>&lt;javascript&gt;&lt;api&gt;&lt;fetch&gt;&lt;wikipedia-api&gt;</code></td>
</tr>
</tbody>
</table>

### 3.3.4 Data Quality and Cleansing

All data processing henceforth was performed using the R programming language\(^4\). R was chosen as it is more suited to the complex data manipulation required for this project than Python, which is more suited to machine learning tasks and production environments.

Although the data had, presumably, already been somewhat curated by Stack Exchange before storing in the SQL database, it is standard practice to conduct a basic data quality report and cleansing step.

- The first data processing task was to concatenate the 5 months of data fetched from the SQL interface. The initial raw input contained 344,154 rows, 92,425 users and 4,703 tags.

- Only the tag most popular on Stack Overflow was kept for each question, the primary tag. This dramatically reduced the size of the dataset but did not remove any useful information as each question should have only one topic. Secondary tags should be children of the primary tag, such as “Scikit-Learn” and parent “Python”.

\(^4\)https://www.r-project.org/
• 0.9% of rows contained missing data, so they were simply dropped as they would have negligible impact on the result.

• A noticeable trend in the tag data was the presence of hyphenated synonyms such as “Python” and “Python-3.x”. This presented an opportunity for further dimensionality reduction. The devised strategy was to replace any hyphenated tags with the text to the left of the hyphen, should that text be a tag itself. This reduced the number of tags by 25%.

• Some tags were present in the data that were irrelevant to the task of software development role clustering, such as “Twitter”, “Facebook”, and “string”. These tags were removed.

• Many users answered questions with only one tag. Nothing can be inferred from these users so they were removed. This reduced the user count by 74%.

• The final cleansed dataset contained 232,889 rows (-32%), 23,708 users (-74%) and 2,783 tags (-41%).

### 3.3.5 Data Exploration

A high-level exploratory data overview was conducted. The purpose of this was to obtain a high-level understanding of the dataset to help inform decisions during modelling.

#### Tag Popularity Distribution

The popularity distribution of tags was investigated. Figure 3.2 shows the cumulative percentage of questions by the percentage of tags ranked by question count.
The distribution is very top-heavy, with the top 7 most popular tags (0.25%) covering off over 50% of all questions. A very large proportion of the least popular tags could be removed without many issues if necessary due to computational constraints.

**User Answer Total**

Figure 3.3 shows a distribution of the total questions answered per user. As expected, when a user answers a question they are less likely to answer a subsequent question.
CHAPTER 3. EXPERIMENT DESIGN AND METHODOLOGY

Unique Tags per User

When comparing the user answer count (Figure 3.3) to the number of unique tags on answered questions (Figure 3.4), the latter drops off at a faster rate than the former. These distributions in combination suggest that users tend to be experts in a small number of tags.

![Figure 3.4: Number of different tags answered by users.](image)

Proportion of Total User Answers per Tag

To elaborate on the previous insight, Figure 3.5 shows the proportion of questions answered per tag for users of different total unique tags answered. The blue line $y = \frac{1}{x}$ shows what the distribution would look like if users answered the same number of questions for each tag they answer questions on. These users would not be experts in any particular tag. It can be seen from the graph that the median percentage of total user answers per tag (middle lines within the boxplots) is less than this line, indicating that users of all breadths of tags answered tend to favour a small number of tags, i.e. user tend to be experts. Interestingly, users who answered questions on 2 tags total tend to answer the same number of questions on each tag.
Figure 3.5: Distribution of percentage of total questions the user answered per tag, and how many total tags they answered.

Ground-Truth Roles

Before attempting to match clusters to ground-truth roles, the ground-truth dataset must be suitably cleansed to be compatible with the Stack Overflow data. The raw dataset is shown in Table 3.3.
Table 3.3: Software development roles and associated skills taken from job postings (Gurcan & Sevik, 2019). This will form the ground-truth clustering for this study.

<table>
<thead>
<tr>
<th>Expertise Role</th>
<th>In-Demand Skills</th>
<th>Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Engineer</td>
<td>Java, Python, C#, JavaScript, C++</td>
<td>12.5</td>
</tr>
<tr>
<td>Frontend Developer</td>
<td>JavaScript, Html5, css, Java, AngularJs</td>
<td>9.5</td>
</tr>
<tr>
<td>Software Developer</td>
<td>Java, C#, C++, JavaScript, Python</td>
<td>8.1</td>
</tr>
<tr>
<td>Mobile Developer</td>
<td>Android, Java, Objective-C, Python, C++</td>
<td>7.6</td>
</tr>
<tr>
<td>Full Stack Engineer</td>
<td>Java, Python, JavaScript, C++, C#</td>
<td>6.9</td>
</tr>
<tr>
<td>Backend Developer</td>
<td>Php, Java, MySQL, Python, Node.Js</td>
<td>5.4</td>
</tr>
<tr>
<td>DevOps Engineer</td>
<td>Linux, Amazon-Web-Services, DevOps, Puppet, Java</td>
<td>5</td>
</tr>
<tr>
<td>Python Developer</td>
<td>Python, Django, PostgreSQL, Php, Jquery</td>
<td>4.7</td>
</tr>
<tr>
<td>Java Developer</td>
<td>Java, Spring, Sql, Hadoop, JavaScript</td>
<td>4.4</td>
</tr>
<tr>
<td>Data Engineer</td>
<td>Sql, MySQL, Oracle, Hadoop, Java</td>
<td>4.3</td>
</tr>
<tr>
<td>Android Developer</td>
<td>Android, Java, Mobile, Python, C++</td>
<td>4.1</td>
</tr>
<tr>
<td>Cloud System Engineer</td>
<td>Cloud, Linux, Java, C#, Amazon-Web-Services</td>
<td>3.9</td>
</tr>
<tr>
<td>Web Developer</td>
<td>Html5, JavaScript, css, AngularJs, Ruby-on-Rails</td>
<td>3.7</td>
</tr>
<tr>
<td>Ruby on Rails Developer</td>
<td>Ruby-on-Rails, Ruby, Python, Node.Js</td>
<td>3.5</td>
</tr>
<tr>
<td>System Engineer</td>
<td>Linux, Windows, Python, Ruby, Java</td>
<td>3.2</td>
</tr>
<tr>
<td>IOS Developer</td>
<td>IOS, Objective-C, Swift, IPhone, Mobile</td>
<td>3</td>
</tr>
<tr>
<td>UI Developer</td>
<td>JavaScript, Html5, css</td>
<td>2.7</td>
</tr>
<tr>
<td>C++ Developer</td>
<td>C++, C#, .Net, Sql, C</td>
<td>2.7</td>
</tr>
<tr>
<td>Quality Assurance Engineer</td>
<td>Sql, Testing, Java, C++</td>
<td>2.5</td>
</tr>
<tr>
<td>PHP Developer</td>
<td>Php, MySQL, WordPress, JQuery, Oop</td>
<td>2.3</td>
</tr>
</tbody>
</table>

First, any skills that were not listed as tags or were very poorly populated were removed as any attempt to compare these tags to the Stack Overflow dataset would be nonsensical. This was to ensure a like-with-like comparison.
Next, any tags that were not present as skills in the ground-truth were removed from the Stack Overflow data, as these tags will never match the ground-truth and only add to noise when clustering. It would be sufficient to prove that the ground-truth roles arise from Stack Overflow data when restricted to the scope of skills from that study. The ground-truth study can then be built upon with the much richer Stack Overflow data.

To understand the resulting number of skills per role, a bar chart was created (Figure 3.6).

![Bar chart](image)

Figure 3.6: Ground-truth skills per software development role.

The number of skills per role varies from 3 to 5. This creates an issue when trying to match unlabelled clusters to their corresponding role, as the top tags per cluster are manually set. A solution to this was to determine the best matching role for each cluster, set the number of tags per cluster to the number of skills in that role by keeping the most popular 3, 4, or 5 tags in that cluster, and discard the remaining roles from the ground-truth. This ensured the same number of roles as clusters and the same number of tags as skills in the best matching role for each cluster. This method enables a fair comparison between clusters and roles without selectively pruning the skills per role in the ground-truth dataset.
3.4 Data Preparation

The last step before modelling was data preparation. In this project, this was a short but relatively involved data manipulation step. The dataset, at this point, was a long-format table of questions, answerers and tags (Table 3.4).

Table 3.4: Long-form dataset (example) prior to manipulation to edge list network format.

<table>
<thead>
<tr>
<th>User</th>
<th>Question</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Python</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>Java</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Python</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>R</td>
</tr>
</tbody>
</table>

The creation of a user-user-weight network format data frame was required. Each row in this data format represents an edge between two users and the weight of the edge. This form of network data is known as an edge list. Only one row is necessary to represent connections between two nodes as the graph is undirected. Therefore, the choice of source and target for the nodes of an edge is arbitrary.

Each user in the long-format data frame was vectorised in tag-space using the number of questions they answer for every tag.

The cosine similarity was calculated in a pairwise fashion between every user. The resulting data frame consisted of pairs of users and their cosine similarity (Equation 2.2, page 11). This similarity measure represents the weight of a connection between pairs of users.

The data set was now in an edge list format, representative of a weighted network of users (Table 3.5).
Table 3.5: Dataset (example) after manipulation to edge list network format.

<table>
<thead>
<tr>
<th>User A</th>
<th>User B</th>
<th>Cosine Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.1234</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.2345</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.3456</td>
</tr>
</tbody>
</table>

3.5 Modelling - Community Detection

From the network of Stack Overflow users weighted by their tag cosine similarity, communities (if present) represent users with common skill-sets. It is reasonable to conjecture that these communities represent different roles within software development. This section outlines the unsupervised community detection method used to identify communities in the network.

The Louvain Method was chosen as the community detection algorithm for this project. The method is described in subsection 2.1.3 (page 12) in the context of other community detection algorithms.

The following steps are illustrated in Figure 3.7.

- The algorithm begins by assigning each node to its own community, and for each node, calculating the change in modularity (Equation 2.3) if the node were to join any of its neighbour’s communities in turn.

- The algorithm will keep the change if there is a positive increase in modularity.

- The communities are then represented as single nodes. Self-loops represent edges within communities. The algorithm is repeated on this network projection.

- These steps are repeated until no further steps are possible.

- The result is a greedy maximisation of modularity through agglomerative clustering. It is a greedy algorithm as it begins with a random node and performs
these operations as they arrive. The result is fast convergence to local maximum modularity, but this may not be the partition that gives the optimal modularity.

Figure 3.7: An example of the Louvain Method (Barabási, 2015).

The Louvain Method performs a specific task in the context of unsupervised clustering; a set of communities of any size is returned such that the network modularity is maximised. One cannot ask the Louvain Method to return a specific number of communities. This means that the number of communities returned may not match the set number of roles in the ground-truth dataset. Furthermore, as the Louvain Method is a hierarchical clustering method, there may exist sub-communities within larger communities that could represent roles in the ground-truth dataset, such as Android Developer and iOS Developer both being Mobile Developers. The Louvain Method does not recognise these sub-communities if the network has higher modularity with them combined as Mobile Developers.

An R iGraph\(^5\) implementation of the Louvain Method was used to detect communities in the Stack Overflow user network. The edge list created in section 3.4 (page 33) was converted to an iGraph object before the algorithm was run. The Louvain Method

---

\(^5\)https://igraph.org/
was efficient enough to allow for the entire network to be used during community detection. A previous attempt at clustering using clique percolation (subsection 2.1.3, page 12) was found to be extremely slow to compute for the entire network and causes several issues with RAM.

3.6 Evaluation

Evaluation of results is as important as the results themselves. Without rigorous evaluation, one cannot claim the validity of their results or hypothesis.

Network statistics are commonly calculated, but much less common is the calculation of their standard errors. One of the main statistical assumptions, independence of events, does not necessarily hold for networks due to the complex interaction between nodes and edges. Therefore, standard approaches to estimating statistical variation do not hold. This is a current area of research emerging with network science. Calculation of standard errors of network statistics enables a quantification of the probability of the statistic occurring by chance alone. Statistical tests such as a Student’s t-test can then be conducted.

Snijders and Borgatti (1999) estimated the standard error of a network statistic by using a bootstrapping strategy. Nodes were sampled with replacement and the sample network retains edges as they exist between pairs of nodes in the original network. In the event of the same node being sampled more than once, an edge value was randomly drawn from a list of all existing edge values in the network. Sampling with replacement is both complex to implement and computationally expensive, so a simulation was performed to empirically estimate the number of duplicated nodes in a sample of 1000 nodes from a network of 23,000 nodes, the number of users in the Stack Overflow dataset. The result was that 2% of nodes would be duplicated upon sampling with replacement. This percentage becomes vanishingly small with increased network size. Therefore, a simple sampling without replacement was used for this project as a good approximation to bootstrap sampling for standard error estimation of the community detection algorithm. Edges between nodes in the sample network were
kept as they were in the full network. The sampled networks are regarded as networks that could have been observed instead of the actual network. This is a valid claim as each user of Stack Overflow is independent of other users.

The bootstrap standard error of a statistic is simply the standard deviation of the bootstrap distribution of that statistic. Therefore, to calculate the standard error of the community detection evaluation measure $F_1h$ from $M = 1000$ evaluation measures $F_1h^*$ from bootstrapped network samples of 1000 nodes each:

$$
\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{M}}
$$

$$
\therefore \text{SE}(F_1h) = \sqrt{\frac{\sum_{m=1}^{M=1000} (F_1h^*(m) - F_1h^*)^2}{999}} \tag{3.1}
$$

The standard error can be estimated for any network statistic. Modularity (Equation 2.3, page 13) was used as a measure of the quality of communities without relying on external sources of data. The standard error of the modularity was also calculated. Harmonic F1-score (Equation 2.7, page 15) was used to evaluate the accuracy of the communities to the ground-truth dataset created by Gurcan and Sevik (2019) and outlined in Table 3.3. The standard error of this score was also calculated.

A one-sample Student’s t-test was carried out to determine, at a confidence level of 95% ($p = 0.05$), if no difference exists between the experimental $F_1h$ score and the theoretical $F_1h$ score of 1 (the communities and ground-truth roles are identical), and the null hypothesis can be rejected. The Student’s t-statistic is the ratio of the difference between an estimated statistic $\hat{\beta}$ and the theoretical value $\beta_0$ to the standard error of the estimated statistic:

$$
t_{\hat{\beta}} = \frac{\hat{\beta} - \beta_0}{\text{SE}(\hat{\beta})}
$$

$$
\therefore t_{F_1h} = \frac{F_1h - 1}{\text{SE}(F_1h)} \tag{3.2}
$$

Consulting a table of critical values for the t-distribution, a t-statistic of greater than -1.66 is required to reject the null hypothesis ($p = 0.05, df = 999$).
Chapter 4

Results, Evaluation and Discussion

This chapter outlines the results obtained from modelling (section 3.5) and evaluation (section 3.6). These results will be discussed in the context of the research question.

4.1 Community Detection

The results from running the Louvain algorithm on the Stack Overflow user network data and selecting the top 5 most common tags per community is shown in Table 4.1. 8 distinct communities were identified.

There is significant overlap between the communities, with globally popular tags such as Javascript occurring in almost every community. This confirms the previous speculation of the existence of tag overlap and also restricts the choice of evaluation metric to one which deals with non-unique community membership, such as harmonic F1-score (Equation 2.7, page 15). The best-matching roles in the ground-truth must be identified for each community in order to calculate the harmonic F1-score, so labelling of the rest of the communities must be completed.
Table 4.1: Results from community detection using the Louvain Method.

<table>
<thead>
<tr>
<th>Community</th>
<th>Most Answered Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c++, c, linux, python, java</td>
</tr>
<tr>
<td>2</td>
<td>python, django, javascript, c++, java</td>
</tr>
<tr>
<td>3</td>
<td>c#, sql, .net, javascript, php</td>
</tr>
<tr>
<td>4</td>
<td>ios, swift, javascript, android, php</td>
</tr>
<tr>
<td>5</td>
<td>java, android, javascript, sql, c#</td>
</tr>
<tr>
<td>6</td>
<td>ruby-on-rails, ruby, sql, javascript, jquery</td>
</tr>
<tr>
<td>7</td>
<td>javascript, angular, html, jquery, css</td>
</tr>
<tr>
<td>8</td>
<td>php, javascript, sql, html, jquery</td>
</tr>
</tbody>
</table>

4.2 Internal Evaluation of Communities

The modularity of the community partition was found to be 0.46 (Equation 2.3, page 13). This positive-valued result indicates the presence of communities that would not occur randomly.

To validate this claim, a random network with the same number of nodes and edges as the user network was created using iGraph. The original edges of the user network were removed, and the same number of random connections were generated between the nodes. The modularity of this randomly generated network was 0, indicating no communities existed in the random network. Therefore, there exists, therefore, a non-random community structure in the user network as the modularity is 0.46.
The standard error (SE) of the modularity $Mod$ was calculated using evaluation measures $Mod^*$ from $M = 1000$ bootstrapped network samples of 1000 nodes each:

\[
SE(\text{Mod}) = \sqrt{\frac{\sum_{m=1}^{M} (\text{Mod}^*(m) - \overline{\text{Mod}}^*)^2}{M - 1}}
\]

\[
= \sqrt{\frac{\sum_{m=1}^{1000} (\text{Mod}^*(m) - 0.46)^2}{999}}
\]

\[
= 0.02
\]

Figure 4.1 shows the modularity scores for the 1000 bootstrapped samples, very closely following a normal distribution.

Figure 4.1: Modularity scores from 1000 bootstrapped samples of 1000 nodes from the Stack Overflow user network. Also plot (blue) is a theoretical normal distribution of the same mean and standard deviation.
4.3 External Evaluation: Comparison to Ground-Truth

For each detected community, the F1-score (Equation 2.6) of each ground-truth role was calculated to find the best-matching role for that community (Table 4.2).

Table 4.2: Stack Overflow Communities and their best-matching role in the ground-truth dataset by F1-score.

<table>
<thead>
<tr>
<th>Community</th>
<th>Best-Matching Ground-Truth Role</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>mobile developer</td>
<td>0.67</td>
</tr>
<tr>
<td>2</td>
<td>Full-Stack/Software</td>
<td>0.80</td>
</tr>
<tr>
<td>3</td>
<td>backend developer</td>
<td>0.60</td>
</tr>
<tr>
<td>4</td>
<td>ios developer</td>
<td>0.57</td>
</tr>
<tr>
<td>5</td>
<td>java developer</td>
<td>0.75</td>
</tr>
<tr>
<td>6</td>
<td>ruby on rails developer</td>
<td>0.67</td>
</tr>
<tr>
<td>7</td>
<td>frontend developer</td>
<td>0.80</td>
</tr>
<tr>
<td>8</td>
<td>php developer</td>
<td>0.75</td>
</tr>
</tbody>
</table>

The ground-truth dataset was then limited to those 8 roles. For each community, the 5 most answered tags were intentionally returned as the maximum number of skills for any role in the ground-truth dataset is 5. If the number of skills in a best-matched role was less than 5, the number of tags in the corresponding community was reduced to match (Table 4.3).
Table 4.3: Details of communities and their best-matching ground-truth roles after filtering steps.

<table>
<thead>
<tr>
<th>Community</th>
<th>Tags</th>
<th>Best-Matching Ground-Truth Role</th>
<th>F1-Score</th>
<th>Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>c++, c, linux, python</td>
<td>mobile developer</td>
<td>0.50</td>
<td>android, java, python, c++</td>
</tr>
<tr>
<td>2</td>
<td>python, django, javascript, c++, java</td>
<td>full-stack/software</td>
<td>0.80</td>
<td>java, python, c#, javascript, c++</td>
</tr>
<tr>
<td>3</td>
<td>c#, sql, .net, javascript, php</td>
<td>backend developer</td>
<td>0.60</td>
<td>php, java, sql, python, javascript</td>
</tr>
<tr>
<td>4</td>
<td>ios, swift</td>
<td>ios developer</td>
<td>1</td>
<td>ios, swift</td>
</tr>
<tr>
<td>5</td>
<td>java, android, javascript</td>
<td>java developer</td>
<td>0.67</td>
<td>java, sql, javascript</td>
</tr>
<tr>
<td>6</td>
<td>ruby-on-rails, ruby, sql, javascript</td>
<td>ruby on rails developer</td>
<td>0.75</td>
<td>ruby-on-rails, ruby, python, javascript</td>
</tr>
<tr>
<td>7</td>
<td>javascript, angular, html, jquery, css</td>
<td>frontend developer</td>
<td>0.80</td>
<td>javascript, html, css, java, angular</td>
</tr>
<tr>
<td>8</td>
<td>php, javascript, sql</td>
<td>php developer</td>
<td>0.75</td>
<td>php, java, sql, python, javascript</td>
</tr>
</tbody>
</table>

The harmonic F1-score between the resultant clusters and ground-truth was found to be 0.75. Figure 4.2 shows the harmonic F1-scores for the 1000 bootstrapped samples, very closely following a normal distribution.

![Figure 4.2: Harmonic F1-scores to the ground-truth dataset from 1000 bootstrapped samples of 1000 nodes from the Stack Overflow user network. Also plot (blue) is a theoretical normal distribution of the same mean and standard deviation.](image)

The standard error of the community detection evaluation measure $F1h$ was calculated using $M = 1000$ evaluation measures $F1h^*$ from bootstrapped network samples of 1000 nodes each:
\[ SE(F1h) = \sqrt{\frac{\sum_{m=1}^{M} (F1h^{s(m)} - F1h^{s})^2}{M - 1}} = \sqrt{\frac{1000 \sum_{m=1}^{1000} (F1h^{s(m)} - 0.68)^2}{999}} = 0.064 \quad (4.2) \]

\section*{4.4 One-Sample T-test}

A reminder of the hypothesis to be tested:

**H0:** Upon attempting to cluster a weighted network of Stack Overflow users (linked if the users have authored accepted answers on questions tagged with the same primary tag, and weighted by the cosine similarity of such users) using the Louvain Method, the harmonic F1-score between the clusters and the assumed ground-truth roles identified in Gurcan and Sevik (2019) is not statistically significantly equal to 1.

**HA:** A weighted network of Stack Overflow users (linked if the users have authored accepted answers on questions tagged with the same primary tag, and weighted by the cosine similarity of such users) using the Louvain Method, the harmonic F1-score between the clusters and the assumed ground-truth of the groups identified in Gurcan and Sevik (2019) is statistically significantly equal to 1. Hence, these clusters can form the basis for role names in modern software development.

To test the alternate hypothesis that the harmonic F1-score of the communities equals 1 and the Stack Overflow communities and ground-truth roles are equivalent, a one-sample Student’s t-test was conducted. This test has some basic assumptions that must be met in order to conduct it. The observations should be independent, approximately normally distributed and not contain outliers.
To check normality, a visual observation of Figure 4.2 confirms a bell-shaped curve, indicative of a normal distribution. To confirm this, a quantile-quantile plot of the distribution against a normal distribution was constructed (Figure 4.3). A straight line without significant deviation confirms that the distribution is approximately normal without significant outliers.

Figure 4.3: Quantile-quantile plot of bootstrap harmonic F1-score against theoretical normal distribution.

The t-statistic was then calculated as the assumption of normality was met.

\[
t_{\hat{\beta}} = \frac{\hat{\beta} - \beta_0}{SE(\hat{\beta})}
\]

\[
\therefore t_{F1h} = \frac{F1h - 1}{SE(F1h)} = \frac{0.748 - 1}{0.064} = -3.93
\]

The t-statistic was not greater than -1.66. Therefore, the null hypothesis was not rejected.

Recalling that the hypothesis to be tested was that there is no difference in the harmonic F1-score between the clusters and the assumed ground-truth, this test checked whether or not the groups were identical. The actual result of the experiment was that
the harmonic F1-score was 0.75, indicating strong similarity to, albeit not identical to, the ground-truth. Specifically, by definition, the harmonic mean of a) the weighted F1-scores of the best matching ground-truth communities to the formed clusters and b) the best matching formed clusters to the ground-truth communities was 0.75.

Two considerations for this result:

- The ground-truth study empirically formed clusters from job postings, which may not represent the actual truth (see assumptions in section 1.5, page 5). Therefore, a result from this project that is identical to this may be improbable.

- One could argue that any strongly agreeing result is indicative of a mutual causality being present. The harmonic F1-score is large enough to suggest this. If $F1_h$ were closer to zero this would not be considered.

### 4.5 Visualising Software Development Communities

Gephi (subsection 2.1.3, page 12) was used to draw a visual representation of the Stack Overflow user network (Figure 4.4). A random sub-graph of 5000 nodes was used to represent the network due to RAM constraints. An algorithm called ForceAtlas2 (Jacomy, Venturini, Heymann, & Bastian, 2014) was used to arrange the nodes at certain distances from each other. ForceAtlas2 adds a repulsive force similar to Coulomb’s Law for electrically charged particles:

$$F_r = \frac{(\text{deg}(n_1) + 1)(\text{deg}(n_2) + 1)}{d} \quad (4.4)$$

And an attractive force similar to Hooke’s Law for springs:

$$F_a = -wd \quad (4.5)$$

Where $\text{deg}(n)$ is the degree of node $n$ (or number of nodes connecting to that node), $d$ is the distance between two nodes, and $w$ is the edge weight.
This projection groups nodes with a higher density of connections together, and will naturally form visible communities. The nodes were coloured according to their membership in one of 8 communities identified in Table 4.1 (page 39), and the communities were labelled according to their best-matching ground-truth role (Table 4.2, page 41).

Figure 4.4: Stack Overflow user network graph (2018 data) with labelled communities (ForceAtlas2 in Gephi). Edges are not shown.
The visualisation provides intuitive insight to a much greater effect than tables of results. There are multiple insights available from this network graph:

- Similar roles are closer together on the network graph.
- Frontend developers are the largest community on Stack Overflow.
- Each community has many smaller satellite communities visible. These communities tend to position themselves along the shortest paths connecting the larger communities. Perhaps developers in these communities lack one or more core skills present in the main community, which would explain the discretised transitions between the main communities. Counting the number of satellite communities between main communities could represent the number of new skills required to transition from one role to another.
- It appears that Backend consists of two large communities. As PHP is also a backend technology, it was labelled as such. Hence, Backend developers could be divided into three sub-communities. It is difficult to reverse-engineer this observation, however, for a number of reasons. The number of communities identified by the Louvain Method cannot be pre-determined and so the addition of an extra community is not possible through algorithmic means, and it is difficult to ring-fence the potential community manually. Regardless, the Louvain Method returned the current partition as to optimise modularity and the introduction of another Backend community could potentially lower network modularity and be a sub-optimal partition.

The ground-truth dataset and the Stack Overflow communities were then conceptualised as network graphs themselves in order to gain an understanding of their structure. Nodes in this graph are the tags and edges are the communities. Two tags are connected if they are both in the same community. Projections of these graphs are shown in Figure 4.5 and Figure 4.6.
CHAPTER 4. RESULTS, EVALUATION AND DISCUSSION

Figure 4.5: Network projection of the identified Stack Overflow communities.

Figure 4.6: Network projection of the ground-truth roles.
For both graphs, the tag with the highest betweenness centrality (section 2.1, page 7) is Javascript; 58% of shortest paths between tags go through Javascript. This is the tag that acts as a “gateway” between tags, and has the widest application of any tag. It could also be regarded as the most versatile tag in terms of bridging knowledge gaps; if a software developer wishes to pursue a different area of software development, Javascript is the most likely tag to be a transferable skill.

4.6 **Comparison to 2021**

Up to this point, Stack Overflow data from April to August 2018 were used as this was the period when data were collected from job postings to create the ground-truth dataset. These periods must match to retain validity of any result. However, the resulting communities identified may not reflect the current environment in software development. Assuming the analysis performed up to this point to be correct and applicable to other time frames, one could change the underlying data to a different period in software development history to observe the changes in role structure through time.

The same method was used to collect, preprocess and model Stack Overflow data from January to April 2021 as before. As there is no ground-truth dataset to assist in refining the communities, there were no restrictions or limits on tags or clusters as before. 5 communities were found using the Louvain Method. The top 5 most common tags per community were extracted (Table 4.4).

Comparing to Table 4.1 (page 39), it is clear that the software development landscape has changed from 2018 to 2021 as the detected communities are different. There were 5 communities detected from 2021 data while there were 8 detected from 2018 data. It is possible to identify some similar communities, but entirely new communities also appear to exist. To assist in this exploration, the network graph for 2021 was created (Figure 4.7). Through a combination of topologically comparing this network graph to the 2018 network graph and comparing the tags in each community, it was possible to appropriately label the communities.
Table 4.4: Results from community detection between 1/1/2021 and 30/4/2021.

<table>
<thead>
<tr>
<th>Community</th>
<th>Most Answered Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sql, c#, php, azure, .net</td>
</tr>
<tr>
<td>2</td>
<td>java, android, flutter, ios, swift</td>
</tr>
<tr>
<td>3</td>
<td>javascript, react, html, angular, css</td>
</tr>
<tr>
<td>4</td>
<td>python, javascript, django, amazon, pandas</td>
</tr>
<tr>
<td>5</td>
<td>r, c++, c, rust, bash</td>
</tr>
</tbody>
</table>

Figure 4.7: Stack Overflow user network graph (2021 data) with labelled communities (ForceAtlas2 in Gephi).
• Frontend and backend developers are still identifiable, although PHP developers that formed a distinct community in 2018 have now rejoined the rest of the backend developers in a single group. This could be due to the decline in popularity of PHP in recent years. In the 2018 Stack Overflow Developer’s Survey\(^1\), 30.7% of respondents used PHP, while in 2020\(^2\) this reduced to 26.2% (-14.7%). This is likely to reduce again, awaiting the 2021 survey.

• iOS developers are no longer a distinct group; the mobile developer community has emerged as a single role with core technologies for both Android and iOS development, and has coalesced with the Java community.

• The area of the graph that was once full-stack/software developers has now become Python-dominant, with Django and Pandas now in the top 5 technologies. Python has seen strong popularity growth year-on-year, with 38.8% in 2018 vs. 44.1% in 2020 (+13.7%), and has developed into its own ecosystem. Python is also a versatile, “multi-paradigm” programming language, and is the equivalent to the full-stack software developer from 2018.

• Ruby on rails developers no longer have their own community, but a new community consisting of experts in R, C and Rust have emerged. The appropriate label for this community is unknown. Perhaps this could be the early signs of a new role in software development.

A direct comparison between the results obtained from 2018 and 2021 data is shown in Table 4.5.

\(^1\)https://insights.stackoverflow.com/survey/2018#technology
\(^2\)https://insights.stackoverflow.com/survey/2020#technology
Table 4.5: A comparison between the communities identified in 2018 vs. 2021, and their best-matching roles.

<table>
<thead>
<tr>
<th>Most Answered Tags</th>
<th>Best-Matching Role</th>
<th>Most Answered Tags</th>
<th>Best-Matching Role</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2018</td>
<td>2021</td>
<td></td>
</tr>
<tr>
<td>c#, sql, net, javascript, php</td>
<td>backend</td>
<td>sql, c#, php, azure, net</td>
<td>backend</td>
</tr>
<tr>
<td>c++, c, linux, python, java</td>
<td>mobile</td>
<td>java, android, flutter, ios, swift</td>
<td>mobile</td>
</tr>
<tr>
<td>ios, swift, javascript, android, php</td>
<td>iOS</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>java, android, javascript, sql, c#</td>
<td>Java</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>javascript, angular, html, jquery, css</td>
<td>frontend</td>
<td>javascript, react, html, angular, css</td>
<td>frontend</td>
</tr>
<tr>
<td>python, django, javascript, c++, java</td>
<td>full-stack/software</td>
<td>python, javascript, django, amazon, pandas</td>
<td>Python</td>
</tr>
<tr>
<td>php, javascript, sql, html, jquery</td>
<td>PHP</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ruby-on-rails, ruby, sql, javascript</td>
<td>ruby on rails</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>r, c++, c, rust, bash</td>
<td>R/C/Rust</td>
</tr>
</tbody>
</table>
Chapter 5

Conclusion

5.1 Research Overview

This project began with the assumption that the sets of questions software developers answer on Stack Overflow are characteristic of their wheelhouse of knowledge. It was shown that these wheelhouses could be grouped into non-random clusters. It was possible to label these clusters as roles within software development through comparison to a previous study. Although it could not be proven that the clusters were identical to those from the previous study, they strongly agreed with each other. Hence, it could be argued that, through analysis of user behaviour, Stack Overflow could be used to form the standard definitions of roles within software development.

5.2 Problem Definition

The software development industry experiences rapid evolution in tools and technology, so much so that the very definitions of roles within software development can become blurred. Companies may advertise roles but may be unsure of the particular set of tools and technologies that would best attract a modern professional seeking a similar role. Conversely, a company may be unsure what to name the role they wish to hire for, or whether or not the role in mind still exists as a distinct entity.
The same issue arises when designing curricula for computer science courses; the educator should be proactive in teaching the very latest technologies to their students, but must have knowledge of where the technologies fit into the landscape of software development as a whole in order to ensure an even coverage of all areas of software development within the curriculum. They must have an understanding of the number of different roles within software development at any time.

5.3 Design/Experimentation, Evaluation & Results

The solution to this lack of up-to-date understanding of the definitions of software development roles could be to let the professionals define the roles themselves through inference from their online collaboration. Similar professionals will group together more strongly than ones with more different skill-sets.

Over a five month period from April to August 2018, the skill-sets of users of Stack Overflow were inferred from the topics (from the tags) of the questions they answered. The cosine similarity between each pair of users was calculated using their skill-set. This value was used as a weighting between each node in a user-user network. Community detection was run on this network using the Louvain method, and eight communities were identified. The modularity of these clusters was found to be 0.46. To validate the results of this unsupervised clustering against an external dataset, a paper (Gurcan & Sevik, 2019) that used data gathered from the same time in 2018 was used. This paper identified 20 main roles for software development, listing 5 in-demand skills for each role. To compare this to the detected communities from this project, the top 5 most commonly answered questions were extracted for each community. The harmonic F1-score was calculated between the two sets of clusters and was found to be 0.75. A one-sample Student’s t-test was conducted to assess if this result was statistically significantly equal to 1, which would indicate perfect agreement with the external data set. The result of the t-test at a 95% confidence level was that the harmonic F1-score was not equal to 1; therefore could not be said that the identified communities were identical to the external dataset. However, it
was highlighted that a harmonic F1-score of 0.75 indicates a high level of similarity between the results. As the methods used in this project differed from the methods used in Gurcan and Sevik (2019), an identical result would be surprising and a result of closely matching non-identical results was more probable. A comparison to results using data from 2021 was made; it was found that significant evolution of roles has occurred from 2018 to 2021.

### 5.4 Contributions and Impact

The results of this project show that:

- It is possible for emergent communities of Stack Overflow users to form the basis of software development roles.

- A new community of R, C and Rust developers exist in the data from 2021. The appropriate label for this community is yet to be determined.

- Stack Overflow data are useable, useful and relatively under-utilised in a research context.

- Network science approaches provide not only equivalent results to classical data science techniques, but also provide a visually striking and intuitive representation of data.

### 5.5 Future Work & Recommendations

Stack Overflow is one of 177 forums on the Stack Exchange network of sites. Data from each Stack Exchange forum can be queried and downloaded in the same way as was done in this project for Stack Overflow (subsection 3.3.1, page 24). Therefore, the methods used in this project can be extended to other forums and therefore other industries, such as mathematics, engineering or even non-technological industries such as journalism\(^1\). It is possible that other research has been done to identify roles in

\(^1\)https://stackexchange.com/sites
these industries that could be used to validate results using data from these forums and therefore further support the hypothesis proposed in this project. Furthermore, a range of other techniques (section 3.2, page 21) could be used to repeat this set of experiments to support the hypothesis.
References


REFERENCES


REFERENCES


REFERENCES


REFERENCES


Listing A.1: The SQL query used to construct the raw dataset.

```sql
SELECT
    Answers.Id,
    Answers.ParentId,
    Answers.OwnerUserId,
    Questions.Tags

FROM Posts AS Answers

INNER JOIN PostTypes
ON PostTypes.Id = Answers.PostTypeId

INNER JOIN Posts as Questions
ON Questions.Id = Answers.ParentId
AND Questions.AcceptedAnswerId = Answers.Id

WHERE PostTypes.Name = 'Answer'
AND Questions.CreationDate BETWEEN '2018-04-01'
AND '2018-08-31'
```