Interpreting Cloud Computer Vision Pain-Points: A Mining Study of Stack Overflow

Alex Cummaudo
ca@deakin.edu.au
Applied Artificial Intelligence Inst.
Deakin University
Geelong, Victoria, Australia

Rajesh Vasa
rajesh.vasa@deakin.edu.au
Applied Artificial Intelligence Inst.
Deakin University
Geelong, Victoria, Australia

Scott Barnett
scott.barnett@deakin.edu.au
Applied Artificial Intelligence Inst.
Deakin University
Geelong, Victoria, Australia

John Grundy
john.grundy@monash.edu
Faculty of Information Technology
Monash University
Clayton, Victoria, Australia

Mohamed Abdelrazek
mohamed.abdelrazek@deakin.edu.au
School of Information Technology
Deakin University
Geelong, Victoria, Australia

ABSTRACT

Intelligent services are becoming increasingly more pervasive; application developers want to leverage the latest advances in areas such as computer vision to provide new services and products to users, and large technology firms enable this via RESTful APIs. While such APIs promise an easy-to-integrate on-demand machine intelligence, their current design, documentation and developer interface hides much of the underlying machine learning techniques that power them. Such APIs look and feel like conventional APIs but abstract away data-driven probabilistic behaviour—the implications of a developer treating these APIs in the same way as other, traditional cloud services, such as cloud storage, is of concern. The objective of this study is to determine the various pain-points developers face when implementing systems that rely on the most mature of these intelligent services, specifically those that provide computer vision. We use Stack Overflow to mine indications of the frustrations that developers appear to face when using computer vision services, classifying their questions against two recent classification taxonomies (documentation-related and general questions). We find that, unlike mature fields like mobile development, there is a contrast in the types of questions asked by developers. These indicate a shallow understanding of the underlying technology that empower such systems. We discuss several implications of these findings via the lens of learning taxonomies to suggest how the software engineering community can improve these services and comment on the nature by which developers use them.

CCS CONCEPTS

• **Information systems → Web services**; Data mining; • **Software and its engineering → Software creation and management**; • **General and reference** → *Empirical studies*; • **Computing methodologies** → *Artificial intelligence*.

KEYWORDS

intelligent services, computer vision, documentation, pain points, stack overflow, empirical study

ACM Reference Format:
Alex Cummaudo, Rajesh Vasa, Scott Barnett, John Grundy, and Mohamed Abdelrazek. 2020. Interpreting Cloud Computer Vision Pain-Points: A Mining Study of Stack Overflow. In *42nd International Conference on Software Engineering (ICSE '20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3377811.3380404

1 INTRODUCTION

The availability of recent advances in artificial intelligence (AI) over simple RESTful end-points offers application developers new opportunities. These new intelligent services (AI components) abstract complex machine learning (ML) and AI techniques behind simpler API calls. In particular, they hide (either explicitly or implicitly) any data-driven and non-deterministic properties inherent to the process of their construction. The promise is that software engineers can incorporate complex machine learnt capabilities, such as computer vision, by simply calling an API end-point.

The expectation is that application developers can use these AI-powered services like they use other conventional software components and cloud services (e.g., object storage like AWS S3). Furthermore, the documentation of these AI components is still anchored to the traditional approach of briefly explaining the end-points with some information about the expected inputs and responses. The presupposition is that developers can reason and work with this high level information. These services are also marketed to suggest that application developers do not need to fully understand how these components were created (i.e., assumptions in training data and training algorithms), the ways in which the components can fail, and when such components should and should not be used.

The nuances of ML and AI powering intelligent services have to be appreciated, as there are real-world consequences to software quality for applications that depend on them if they are ignored [16]. This is especially true when ML and AI are abstracted and masked behind a conventional-looking API call, yet the mechanisms behind the API are data-dependent, probabilistic and potentially non-deterministic [29]. We are yet to discover what long-term impacts exist during development and production due to poor documentation that do not capture these traits, nor do we know the depth of understanding application developers have for these components. Given the way AI-powered services are currently presented, developers are also likely to reason about these new services much like a string library or a cloud data storage service. That is, they may not fully consider the implications of the underlying statistical nature of these new abstractions or the consequent impacts on productivity and quality.

Typically, when developers are unable to correctly align to the mindset of the API designer, they attempt to resolve issues by (re-)reading the API documentation. If they are still unable to resolve these issues on their own after some internet searching, they consider online discussion platforms (e.g., Stack Overflow, GitHub Issues, Mailing Lists) where they seek technological advice from their peers [1]. Capturing what developers discuss on these platforms offers an insight into the frustrations developers face when using different software components as shown by recent works [10, 22, 32, 35, 41]. However, to our knowledge, no studies have yet analysed what developers struggle with when using the new generation of *intelligent* services. Given the re-emergent interest in AI and the anticipated value from this technology [27], a better understanding of issues faced by developers will help us improve the quality of services. Our hypothesis is that application developers do not fully appreciate the probabilistic nature of these services, nor do they have sufficient appreciation of necessary background knowledge—however, we do not know the specific areas of concern. The motivation for our study is to inform API designers on which aspects to focus in their documentation, education, and potentially refine the design of the end-points.

This study involves an investigation of 1,825 Stack Overflow (SO) posts regarding one of the most mature types of intelligent services—computer vision services—dating from November 2012 to June 2019. We adapt existing methodologies of prior SO analyses [10, 37] to extract posts related to computer vision services. We then apply two existing SO question classification schemes presented at ICPC and ICSE in 2018 and 2019 [1, 9]. These previous studies focused on mobile apps and web applications. Although not a direct motivation, our work also serves as a validation of the applicability of these two issue classification taxonomies [1, 9] in the context of intelligent services (hence potential for generalisation). Additionally our work is the first—to our knowledge—to *test* the applicability of these taxonomies in a new study.

The taxonomies in previous works focus on the specific aspects from the domain (e.g. API usage, specificity within the documentation etc.) and as such do not deeply consider the learning gap of an application developer. To explore the API learning implications raised by our SO analysis, we applied an additional lens of two taxonomies from the field of pedagogy. This was motivated by the need to offer an insight into the work needed to help developers learn how to use these relatively new services.

The key findings of our study are:

- The primary areas that developers raise as issues reflect a relatively primitive understanding of the underlying concepts of data-driven ML approaches used. We note this via the issues raised due to conceptual misunderstanding and confusion in interpreting errors,
- Developers predominantly encounter a different distribution of issue types than were reported in previous studies, indicating the complexity of the technical domain has a non-trivial influence on intelligent API usage; and
- Most of these issues can be resolved with better documentation, based on our analysis.

The paper also offers a data-set as an additional contribution to the research community and to permit replication [42]. The paper structure is as follows: section 2 provides motivational examples to highlight the core focus of our study; section 3 provides a background on prior studies that have mined SO to gather insight into the SE community; section 4 describes our study design in detail; section 5 presents the findings from the SO extraction; section 6 offers an interpretation of the results in addition to potential implications that arise from our work; section 7 outlines the limitations of our study; concluding remarks are given in section 8.

2 MOTIVATION

"Intelligent" services are often available as a cloud end-point and provide developers a friendly approach to access recent AI/ML advances without being experts in the underlying processes. Figure 1 highlights how these services abstract away much of the technical know-how needed to create and operationalise these intelligent services [30]. In particular, they hide information about the training algorithm and data-sets used in training, the evaluation procedures, the optimisations undertaken, and—surprisingly—they often do not offer a properly versioned end-point [16, 29]. That is, the cloud vendors may change the behaviour of the services without sufficient transparency.

The trade-off towards ease of use for application developers, coupled with the current state of documentation (and assumed developer background) has a cost as reflected in the increasing discussions on developer communities such as SO (see fig. 2). To illustrate the key concerns, we list below a few up-voted questions:

- **unsure of ML specific vocabulary:** "*Though it's now not so clear to me what 'score' actually means.*" [43]; "*I'm trying out the [intelligent service], and there's a score field that returns that I'm not sure how to interpret [it].*" [44]
- **frustrated about non-deterministic results:** "*Often the API has troubles in recognizing single digits... At other times Vision confuses digits with letters.*" [45]; "*Is there a way to help the program recognize numbers better, for example limit the results to a specific format, or to numbers only?*" [46]
- **unaware of the limitations behind the services:** "*Is there any API available where we can recognize human other body parts (Chest, hand, legs and other parts of the body), because as per the Google vision API it's only able to detect face of the human not other parts.*" [47]
- **seeking further documentation:** "*Does anybody know if Google has published their full list of labels (*`['produce', 'meal', ...]`*) and where I could find that? Are those labels*

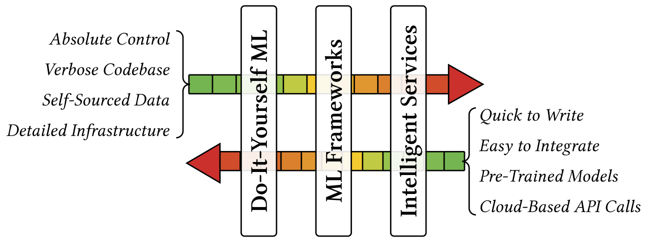


Figure 1: Some traits of Intelligent Services vs. 'Do-It-Yourself' ML. Green-to-red arrows indicate the presence of these traits. *Adapted from Ortiz [30].*

> *structured in any way? - e.g. is it known that* `'food'` *is a superset of* `'produce'`*, for example.*" [48]

The objective of our study is to better understand the nature of the questions that developers raise when using intelligent services, in order to inform the service designers and documenters. In particular, the knowledge we identify can be used to improve the documentation, educational material and (potentially) the information contained in the services' response objects—these are the main avenues developers have to learn and reason about when using these services. There is previous work that has investigated issues raised by developers [1, 9, 37]. We build on top of this work by adapting the study methodology and apply the taxonomies offered to identify the nature of the issues and this results in the following research questions in this paper:

RQ1. How do developers mis-comprehend intelligent services as presented within Stack Overflow pain-points? While the AI community is well aware in the the nuances that empower intelligent services, such services are being released for application developers who may not be aware of their limitations or how they work. This is especially the case when machine intelligence is accessed via web-based APIs where such details are not fully exposed.

RQ2. Are the distribution of issues similar to prior studies? We compare how the distributions of previous studies' of posts about conventional, deterministic API services differ from those of intelligent services. By assessing the distribution of intelligent services' issues against similar studies that focus on mobile and web development, we identify whether a new taxonomy is needed specific to AI-based services, and if gaps specific to AI knowledge exist that need to be captured in these taxonomies.

3 BACKGROUND

The primary goal of analysing issues is to better understand the root causes. Hence, a good issue classification taxonomy should ideally capture the underlying causal aspects (instead of pure functional groupings) [13]. Although this idea (of cause related classification) is not new (Chillarege advocated for it in this TSE paper in 1992), this is not a universally followed approach when studying online discussions and some recent works have largely classified issues into the *"what is"* and not *"how to fix it"* [7, 10, 39]. They typically (manually) classify discussion into either *functional areas* (e.g., Website Design/CSS, Mobile App Development, .NET Framework, Java [7]) or *descriptive areas* (e.g., Coding Style/Practice, Problem/Solution, Design, QA [7, 39]). As a result, many of these studies do not give us a prioritised means of targeted attack on how to *resolve* these issues with, for example, improved documentation. Interestingly, recent taxonomies that studied SO data (Aghajani et al. [1] and Beyer et al. [9]) were causal in nature and developed to understand discussions related to mobile and web applications. However, issues that arise when developers use intelligent services have not been studied, nor do we know if existing issue classification taxonomies are sufficient in this domain.

Researchers studying APIs have also attempted to understand developer's opinions towards APIs [39], categorise the questions they ask about these APIs [7–9, 32], and understand API related documentation and usage issues [1–3, 7, 19, 37]. These studies often employ automation to assist in the data analysis stages of their research. Latent Dirichlet Allocation [3, 7, 32, 39] is applied for topic modelling and other ML techniques such as Random Forests [9], Conditional Random Fields [2] or Support Vector Machines [9, 19] are also used.

However, automatic techniques are tuned to classify into *descriptive* categories, that is, they help paint a landscape of *what is*, but generally do not address the causal factors to address the issues in great detail. For example, functional areas such as 'Website Design' [7], 'User Interface' [10] or 'Design' [40] result from such analyses. These automatic approaches are generally non-causal, making it hard to address reasons for *why* developers are asking such questions. However, not all studies in the space use automatic techniques; other studies employ manual thematic analysis [1, 8, 37] (e.g., card sorting) or a combination of both [9, 10, 32, 38]. Our work uses a manual approach for classification, and we use taxonomies that are more causally aligned allowing our findings to be directly useful in terms of addressing the issues.

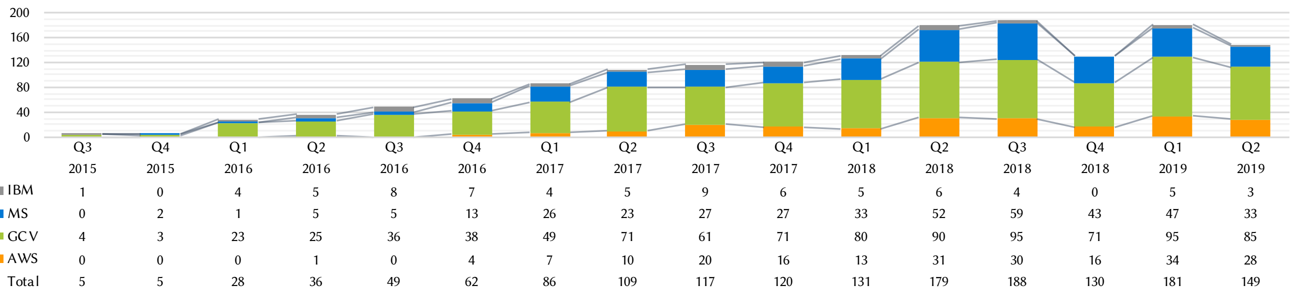
Evidence-based SE [23] has helped shape the last 15 years worth of research, but the reliability of such evidence has been questioned [20, 21, 33]. Replication studies, especially in empirical works, can give us the confidence that existing results are adaptable to new domains; in this context, we extend (to intelligent services) and work with study methods developed in previous works.

4 METHOD

4.1 Data Extraction

This study initially attempted to capture SO posts on a broad range of many intelligent services by identifying issues related to four popular intelligent service cloud providers: Google Cloud [49], AWS [50], Azure [51] and IBM Cloud [52]. We based our selection criteria on the prominence of the providers in industry (Google, Amazon, Microsoft, IBM) and their ubiquity in cloud platform services. Additionally, in 2018, these services were considered the most adopted cloud vendors for enterprise applications [31].

However, during the filtering stage (see section 4.2), we decided to focus on a subset of these services, computer vision, as these are one of the more mature and stable ML/AI-based services with widespread and increasing adoption in the developer community (see fig. 2). We acknowledge other services beyond the four analysed provide similar capabilities [53–58] and only English-speaking services have been selected, excluding popular services from Asia



	Q3 2015	Q4 2015	Q1 2016	Q2 2016	Q3 2016	Q4 2016	Q1 2017	Q2 2017	Q3 2017	Q4 2017	Q1 2018	Q2 2018	Q3 2018	Q4 2018	Q1 2019	Q2 2019
■ IBM	1	0	4	5	8	7	4	5	9	6	5	6	4	0	5	3
■ MS	0	2	1	5	5	13	26	23	27	27	33	52	59	43	47	33
■ GCV	4	3	23	25	36	38	49	71	61	71	80	90	95	71	95	85
■ AWS	0	0	0	1	0	4	7	10	20	16	13	31	30	16	34	28
Total	5	5	28	36	49	62	86	109	117	120	131	179	188	130	181	149

Figure 2: Trend of posts, where IBM = IBM Watson Visual Recognition, MS = Azure Computer Vision, AWS = AWS Rekognition and GCV = Google Cloud Vision. Three MS posts from Q4 2012, Q3 2013 and Q4 2013 have been removed for graph clarity.

(e.g., [59–63])—see section 7. For comprehensiveness, we explain below our initial attempts to extract *all* intelligent services.

4.1.1 *Defining a list of intelligent services.* As there exists no global 'list' of intelligent services to search on, we needed to derive a *corpus of initial terms* to allow us to know *what* to search for on the Stack Exchange Data Explorer[1] (SEDE). We began by looking at different brand names of cloud services and their permutations (e.g., Google Cloud Services and GCS) as well as various ML-related products (e.g., Google Cloud ML). To do this, we performed extensive Google searches[2] in addition to manually reviewing six 'overview' pages of the relevant cloud platforms. We identified 91 initial intelligent services to incorporate into our search terms[3].

4.1.2 *Manual search for relevant, related terms.* We then ran a manual search[2] on each term to determine if these terms were relevant. We did this by querying each term within SO's search feature, reviewing the titles and body post previews of the first three pages of results (we did not review the answers, only the questions). We also noted down the user-defined *Tags* of each post (up to five per question); by clicking into each tag, we could review similar tags (e.g., 'project-oxford' for 'azure-cognitive-services') and check if the tag had synonyms (e.g., 'aws-lex' and 'amazon-lex'). We then compiled a *corpus of tags* consisting of 31 terms.

4.1.3 *Developing a search query.* We recognise that searching SEDE via *Tags* exclusively can be ineffective (see [7, 37]). To mitigate this, we produced a *corpus of title and body terms*. Such terms are those that exist within the title and body of the posts to reflect the ways in which individual developers commonly use to refer to different intelligent services. To derive at such a list, we performed a search[2,3] of the 31 tags above in SEDE, filtering out posts that were not answers (i.e., questions only) as we wanted to see how developers *phrase* their questions. For each search, we extracted a random sample of 100 questions (400 total for each service) and reviewed each question. We noted many patterns in the permutations of how developers refer to these services, such as: common misspellings ('bind' vs. 'bing'); brand misunderstanding ('Microsoft computer vision' vs. 'Azure computer vision'); hyphenation ('Auto-ML' vs. 'Auto ML'); UK and US English ('Watson Analyser' vs. 'Watson Analyzer'); and, the use of apostrophes, plurals, and abbreviations ('Microsoft**'s** Computer Vision API', 'Microsoft Computer Vision Services', 'GCV' vs. 'Google Cloud Vision'). We arrived at a final list of 229 terms compromising all of the intelligent services provided by Google, Amazon, Microsoft and IBM as of January 2019[3].

4.1.4 *Executing our search query.* Our next step was to perform a case-insensitive search of all 229 terms within the body or title of posts. We used Google BigQuery's public data-set of SO posts[4] to overcome SEDE's 50,000 row limit and to conduct a case-insensitive search. This search was conducted on 10 May 2019, where we extracted 21,226 results. We then performed several filtering steps to cleanse our extracted data, as explained below.

4.2 Data Filtering

4.2.1 *Refining our inclusion/exclusion criteria.* We performed an initial manual filtering of the 50 most recent posts (sorted by descending *CreationDate* values) of the 21,226 posts above, assessing the suitability of the results and to help further refine our inclusion and exclusion criteria. We did note that some abbreviations used in the search terms (e.g., 'GCV', 'WCS'[5]), resulting in irrelevant questions in our result set. We therefore removed abbreviations from our search query and consolidated all overlapping terms (e.g., 'Google Vision **API**' was collapsed into 'Google Vision').

We also recognised that 21,226 results would be non-trivial to analyse without automated techniques. As we wanted to do manual qualitative analysis, we reduced our search space to 27 search terms of just the *computer vision services* within the original corpus of 229 terms. These were Google Cloud Vision [49], AWS Rekognition [50], Azure Computer Vision [51], and IBM Watson Visual Recognition [52]. This resulted in 1,425 results that were extracted on 21 June 2019. The query used and raw results are available online in our supplementary materials [42].

4.2.2 *Duplicates.* Within 1,425 results, no duplicate questions were noted, as determined by unique post ID, title or timestamp.

4.2.3 *Automated and manual filtering.* To assess the suitability and nature of the 1,425 questions extracted, the first author began with a manual check on a randomised sample of 50 questions. As the questions were exported in a raw CSV format (with HTML tags included in the post's body), we parsed the questions through an ERB

[1]http://data.stackexchange.com/stackoverflow
[2]This search was conducted on 17 January 2019
[3]For reproducibility, this is available at http://bit.ly/2ZcwNJO.
[4]http://bit.ly/2LrN7OA
[5]Watson Cognitive Services

templating engine script[6] in which the ID, title, body, tags, created date, and view, answer and comment counts were rendered for each post in an easily-readable format. Additionally, SQL matches in the extraction process were also highlighted in yellow (i.e., in the body of the post) and listed at the top of each post. These visual cues helped to identify 3 false positive matches where library imports or stack traces included terms within our corpus of 26 computer vision service terms. For example, `aws-java-sdk-rekognition:jar` is falsely matched as a dependency within an unrelated question. As such exact matches would be hard to remove without the use of regular expressions, and due to the low likelihood (6%) of their appearance, we did not perform any followup automatic filtering.

4.2.4 Classification. Our 1,425 posts were then split into 4 additional random samples (in addition to the random sample of 50 above). 475 posts were classified by the first author and three other research assistants, software engineers with at least 2 years industry experience, assisted to classify the remaining 900. This left a total of 1,375 classifications made by four people plus an additional 450 classifications made from reliability analysis, in which the remaining 50 posts were classified nine times (as detailed in section 4.3.1). Thus, a total of 1,825 classifications were made from the original 1,425 posts extracted.

Whilst we could have chosen to employ topic modelling, these are too descriptive in nature (as discussed in section 3). Moreover, we wanted to see if prior taxonomies can be applied to intelligent services (as opposed to creating a new one) and compare if their distributions are similar. Therefore, we applied the two existing taxonomies described in section 3 to each post; (i) a documentation-specific taxonomy that addresses issues directly resulting from documentation, and (ii) a generalised taxonomy that covers a broad range of SO issues in a well-defined SE area (specifically mobile app development). Aghajani et al.'s documentation-specific taxonomy (Taxonomy A) is multi-layered consisting of four dimensions and 16 sub-categories [1]. Similarly, Beyer's SO generalised post classification taxonomy (Taxonomy B) consists of seven dimensions [9]. We code each dimension with a number, X, and each sub-category with a letter y: (Xy). We describe both taxonomies in detail within table 1. Where a post was included in our results but not applicable to intelligent services (see section 4.2.3) or not applicable to a taxonomy dimension/category, then the post was flagged for removal in further analysis. Table 1 presents *our understanding* of the respective taxonomies; our intent is not to methodologically replicate Aghajani et al. or Beyer et al.'s studies in the intelligent service domain, rather to acknowledge related work in the area of SO classification and reduce the need to synthesise a new taxonomy. We baseline all coding against *our interpretation only.* Our classifications are therefore independent of the previous authors' findings.

4.3 Data Analysis

4.3.1 Reliability of Classification. To measure consistency of the categories assigned by each rater to each post, we utilised both intra- and inter-rater reliability [28]. As verbatim descriptions from dimensions and sub-categories were considered quite lengthy from their original sources, all raters met to agree on a shared interpretation of the descriptions, which were then paraphrased as discussed in the previous subsection and tabulated in table 1. To perform statistical calculations of reliability, each category was assigned a nominal value and a random sample of 50 posts were extracted. Two-phase reliability analysis followed.

Firstly, intra-rater agreement by the first author was conducted twice on 28 June 2019 and 9 August 2019. Secondly, inter-rater agreement was conducted with the remaining four co-authors in addition to three research assistants within our research group in mid-August 2019. Thus, the 50 posts were classified an additional nine times, resulting in 450 classifications for reliability analysis. We include these classifications in our overall analysis.

At first, we followed methods of reliability analysis similar to previous SO studies (e.g., [37]) using the percentage agreement metric that divides the number of agreed categories assigned per post by the total number of raters [28]. However, percentage agreement is generally rejected as an inadequate measure of reliability analysis [14, 18, 24] in statistical communities. As we used more than 2 coders and our reliability analysis was conducted under the same random sample of 50 posts, we applied *Light's Kappa* [26] to our ratings, which indicates an overall index of agreement. This was done using the `irr` computational R package [17] as suggested in [18].

4.3.2 Distribution Analysis. In order to compare the distribution of categories from our study with previous studies we carried out a χ^2 test. We selected a χ^2 test as the following assumptions [34] are satisfied: (i) the data is categorical, (ii) all counts are greater than 5, and (iii) we can assume simple random sampling. The null hypothesis describes the case where each population has the same proportion of observations and the alternative hypothesis is where at least one of the null hypothesis statements is false. We chose a significance value, α, of 0.05 following a standard rule of thumb. As to the best of our knowledge this is the first statistical comparison using Taxonomy A and B on SO posts. To report the effect size we selected Cramer's Phi, ϕ_c which is well suited for use on nominal data [34].

5 FINDINGS

We present our findings from classifying a total of 1,825 SO posts aimed at answering RQs 1 and 2. 450 posts were classified using Taxonomies A and B for reliability analysis as described in section 4.3.1 and the remaining 1,375 posts were classified as per section 4.2.4. A summary of our classification using Taxonomies A and B is shown in fig. 3.

5.1 Post classification and reliability analysis

When undertaking the classification, we found that 238 issues (13.04%) did not relate to intelligent services directly. For example, library dependencies were still included in a number of results (see section 4.2.3), and we found there to be many posts discussing Android's Mobile Vision API as Google (Cloud) Vision. These issues were flagged and ignored for further analysis (see section 4.2.4).

For our reliability analysis, we classified a total of 450 posts of which 70 posts were flagged as irrelevant. Landis and Koch [25] provide guidelines to interpret kappa reliability statistics, where

[6]We make this available for future use at: http://bit.ly/2NqBB70

Table 1: Descriptions of dimensions (■) and sub-categories (↪) from both taxonomies used.

A	Documentation-specific classification (Aghajani et al. [1])	
A-1	**■ Information Content (What)**	Issues related to what is written in the documentation
A-1a	↪ ***Correctness***	What exists in the documentation actually matches what is implemented in code
A-1b	↪ ***Completeness***	The documentation fully covers all aspects of the API's components
A-1c	↪ ***Up-to-dateness***	What is documented is accurate to the current version of the API
A-2	**■ Information Content (How)**	Issues related to how the document is written and organised
A-2a	↪ ***Maintainability***	The upkeep effort to ensure the documentation remains up to date
A-2b	↪ ***Readability***	The extent to which the documentation is interpretable
A-2c	↪ ***Usability***	How useable the organisation, look and feel of the documentation is
A-2d	↪ ***Usefulness***	The usefulness of the documentation, avoiding misinformation.
A-3	**■ Process-Related**	Issues related to the documentation process
A-3a	↪ ***Internationalisation***	Translating the documentation into other languages
A-3b	↪ ***Contribution-Related***	Contribution issues encountered when people contribute to the documentation
A-3c	↪ ***Configuration-Related***	Configuration issues of the documentation tool
A-3d	↪ ***Implementation-Related***	Unwanted development issues caused by (poor) documentation
A-3e	↪ ***Traceability***	Tracing documentation changes (when, when, who and why)
A-4	**■ Tool-Related**	Issues related to documentation tools (e.g., Javadoc)
A-4a	↪ ***Tooling Bugs***	Bugs that exist within the documentation tooling
A-3b	↪ ***Tooling Discrepancy***	Support as expectations not being fulfilled by these documentation tools
A-3c	↪ ***Tooling Help Required***	Help required due to improper usage of the tools
A-3d	↪ ***Tooling Migration***	Issues migrating the tool to a new version or another tool
B	**Generalised classification (Beyer et al. [9])**	
B-1	**■ API usage**	Issue on how to implement something using a specific component provided by the API
B-2	**■ Discrepancy**	The questioner's *expected behaviour* of the API does not reflect the API's *actual behaviour*
B-3	**■ Errors**	Issue regarding some form of error when using the API, and provides an exception and/or stack trace to help understand why it is occurring
B-4	**■ Review**	The questioner is seeking insight from the developer community on what the best practices are using a specific API or decisions they should make given their specific situation
B-5	**■ Conceptual**	The questioner is trying to ascertain limitations of the API and its behaviour and rectify issues in their conceptual understanding on the background of the API's functionality
B-6	**■ API change**	Issue regarding changes in the API from a previous version
B-7	**■ Learning**	The questioner is seeking for learning resources to self-learn further functionality in the API, and unlike discrepancy, there is no specific problem they are seeking a solution for

$0.00 \leq \kappa \leq 0.20$ indicates *slight* agreement and $0.21 \leq \kappa \leq 0.40$ indicates *fair* agreement. Despite all raters meeting to agree on a shared interpretation of the taxonomies (see section 4.3.1) our inter-rater measures aligned *slightly* (0.148) for Taxonomy A and *fairly* (0.295) for Taxonomy B. We report further in section 7.

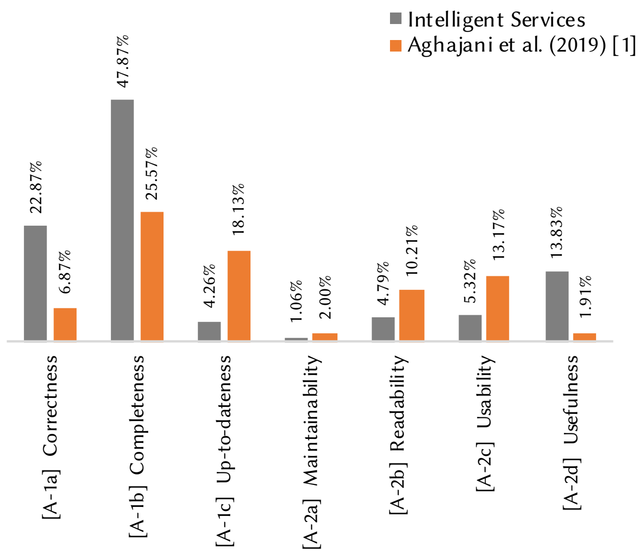
5.2 Developer Frustrations

We found Beyer et al.'s high-level abstraction taxonomy (Taxonomy B) was able to classify 86.52% of posts. 10.30% posts were assigned exclusively under Aghajani et al.'s documentation-specific taxonomy (Taxonomy A). We found that developers do not generally ask questions exclusive to documentation, and typically either pair documentation-related issues to their own code or context. The following two subsections further explain results from both Taxonomy A and B's perspective.

5.2.1 Results from Aghajani et al.'s taxonomy. Results for Aghajani et al.'s low-level documentation taxonomy (Taxonomy A), indicates that most discussion on SO does not directly relate to documentation about an intelligent service. We did not find any process-related (A-3) or tool-related (A-4) questions as, understandably, the developers who write the documentation of the intelligent services would not be posting questions of such nature on SO. One can *infer* documentation-related issues from posts (i.e., parts of the documentation *lacking* that may cause the issue posted). However, there are few questions that *directly* relate to documentation of intelligent services.

Few developers question or ask questions directly about the API documentation, but some (47.87%) posts ask for additional information to understand the API (**completeness (A-1b)**), for example: "*Is there a full list of potential labels that Google's Vision API will return?*" [48]; "*There seems to be very little to no documentation for AWS iOS text recognition inside an image*" [64].

22.87% of posts question the **accuracy (A-1a)** of certain parts of the cloud documentation, especially in relation to incorrect quotas and limitations: "*Are the Cloud Vision API limits in documentation*



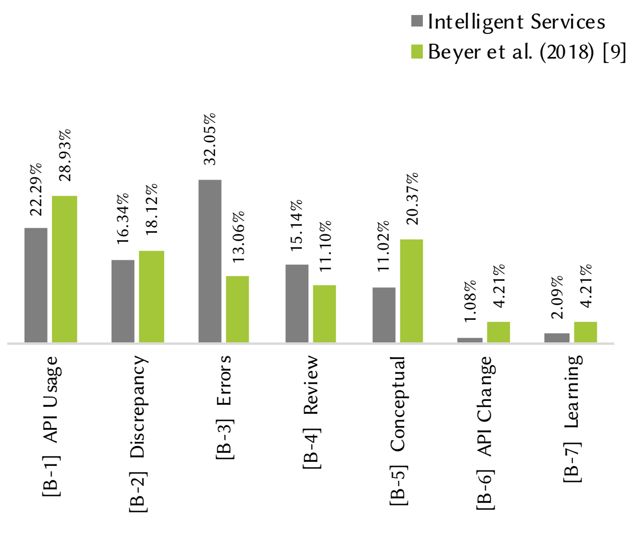


Figure 3: *Left:* Documentation-specific classification taxonomy results highlights a mostly similar distribution to that of Aghajani et al.'s findings [1]. *Right:* Generalised classification taxonomy results highlight differences from more mature fields (i.e., Android APIs in Beyer et al. [9]) to less mature fields (i.e., intelligent services).

correct?" [65], "*According to the Google Vision documentation, the maximum number of image files per request is 16. Elsewhere, however, I'm finding that the maximum number of requests per minute is as high as 1800.*" [66].

There are also many references (23.94%) addressing the confusing nature of some documentation, indicating that the **readability, usability and usefulness of the documentation (A-2b, A-2c and A-2d)** could be improved. For example, "*Am I encoding it correctly? The docs are quite vague.*" [67], "*The aws docs for this are really confusing.*" [68].

5.2.2 Results from Beyer et al.'s taxonomy. We found that a majority (32.05%) of posts are primarily **error-related questions (B-3)**, including a dump of the stack trace or exception message from the service's programming-language SDK (usually Java, Python or C#) that relates to a specific error. For example: "*I can't fix an error that's causing us to fall behind.*" [69]; "*I'm using the Java Google Vision API to run through a batch of images... I'm now getting a* `channel closed` *and* `ClosedChannelException` *error on the request.*" [70].

API usage questions (B-1) were the second highest category at 22.29% of posts. Reading the questions revealed that many developers present an insufficient understanding of the behaviour, functional capability and limitation of these services and the need for further data processing. For example, while Azure provides an image captioning service, this is not universal to all computer vision services: "*In Amazon Rekognition for image processing how do I get the caption for an image?*" [71]. Similarly, OCR-related and label-related questions often indicate interest in cross-language translation, where a separate translation service would be required: "*Can Google Cloud Vision generate labels in Spanish via its API?*" [72]; "*[How can I] specify language for response in Google Cloud Vision API*" [73]; "*When I request a text detection of an image, it gives only English Alphabet characters (characters without accents) which is not enough for me. How can I get the UTF-32 characters?*" [74].

It was commonplace to see questions that demonstrate a lack of depth in understanding and appreciating how these services work, instead posting simple debugging questions. For instance, in the 11.02% of **conceptual-related questions (B-5)** that we categorised, we noticed causal links to a misunderstanding (or lack of awareness) of the vocabulary used within computer vision. For example: "*The problem is that I need to know not only <u>what</u> is on the image but also the <u>position</u> of that object. Some of those APIs have such feature but only for face detection.*" [75]; "*I want to know if the new image has a face <u>similar</u> to the original image.... [the service] can <u>identify</u> faces, but can I use it to get similar faces to the identified face in other images?*" [76]. It is evident that some application developers are not aware of conceptual differences in computer vision such as object/face *detection* versus *localisation* versus *recognition*.

In the 16.34% of **discrepancy-related questions (B-2)**, we see further unawareness from developers in how the underlying systems work. In OCR-related questions, developers do not understand the pre-processing steps required before an OCR is performed. In instances where text is separated into multiple columns, for example, text is read top-down rather than left-to-right and segmentation would be required to achieve the expected results. For example, "*it appears that the API is using some kind of logic that makes it scan top to bottom on the left side and moving to right side and doing a top to bottom scan.*" [77]; "*this method returns scanned text in wrong sequence... please tell me how to get text in proper sequence.*" [78].

A number of **review-related questions (B-4)** (15.14%) seem to provide some further depth in understanding the context to which these systems work, where training data (or training stages) are needed to understand how inferences are made: "*How can we find an exhaustive list (or graph) of all logos which are effectively recognized using Google Vision logo detection feature?*" [79]; "*when object* banana *is detected with accuracy greater than certain value, then next action will be dispatched... how can I confidently define and validate the threshold value for each item?*" [80].

API change (B-6) was shown in 1.08% of posts, with evolution of the services occurring (e.g., due to new training data) but not necessarily documented "*Recently something about the Google Vision API changed... Suddenly, the API started to respond differently to my requests. I sent the same picture to the API today, and I got a different response (from the past).*" [81].

5.3 Statistical Distribution Analysis

We obtained the following results $\chi^2 = 131.86$, $\alpha = 0.05$, $p\ value = 2.2 \times 10^{-16}$ and $\phi_c = 0.362$ from our distribution analysis with Taxonomy A to compare our study with that of Aghajani et al. [1]. Comparing our study to Beyer et al. [9] produced the following results $\chi^2 = 145.58$, $\alpha = 0.05$, $p\ value = 2.2 \times 10^{-16}$ and $\phi_c = 0.252$. These results show that we are able to reject the null hypothesis that the distribution of posts using each taxonomy was the same as the comparison study. While there are limited guidelines for interpreting ϕ_c when there is no prior information for effect size [36], Sun et al. suggests the following: $0.07 \leq \phi_c \leq 0.20$ indicates a *small* effect, $0.21 \leq \phi_c \leq 0.35$ indicates a *medium* effect, and $0.35 > \phi_c$ indicates a *large* effect. Based on this criteria we obtained a *large* effect size for the documentation-specific classification (Taxonomy A) and a *medium* effect size for the generalised classification (Taxonomy B).

6 DISCUSSION

6.1 Answers to Research Questions

RQ1. How do developers mis-comprehend intelligent services as presented within Stack Overflow pain-points? Upon meeting to discuss the discrepancies between our categorisation of intelligent service usage SO posts, we found that our interpretations of the *posts themselves* were largely subjective. For example, many posts presented multi-faceted dimensions for Taxonomy B; Beyer et al. [9] argue that a post can have more than one question category and therefore multi-label classification is appropriate at times. We highlight this further in the threats to validity (section 7).

We have to define the context of intelligent services to address RQ1. We use the concept of a "technical domain" [4] to define this context. A technical domain captures the domain-specific concerns that influence the non-functional requirements of a system [4]. In the context of intelligent services, the technical domain includes exploration, data engineering, distributed infrastructure, training data, and model characteristics as first class citizens [4]. We would then expect to see posts on SO related to these core concerns.

In fig. 3, for the documentation-specific classification, the majority of posts were classified as **Completeness (A1-b)** related (47.87%). An interpretation for this is that the documentation does not adequately cover the technical domain concerns. Comments by developers such as "*I'm searching for a list of all the possible image labels that the Google Cloud Vision API can return?*" [82] indicates the documentation does not adequately describe the training data for the API—developers do not know the required usage assumptions. Another quote from a developer, "*Can Google Cloud Vision generate labels in Spanish via its API? ... [Does the API] allow to select which language to return the labels in?*" [72] points to a lack of details relating to the characteristics of the models used by the API. It would seem that developers are unaware of aspects of the technical domain concerns.

The next most frequent category is **Correctness (A-1a)** with 22.87% of posts. In the context of the technical domain there are many limits that developers need to be aware of: range and increments of a model score [16]; required data pre-processing steps for optimal performance; and features provided by the models (as explained in section 5.2.2). Considering the relation between technical concerns and software quality, developers are right to question providers on correctness; "*Are the Cloud Vision API limits in documentation correct?*" [65].

RQ2. Are the distribution of issues similar to prior studies? Visual inspection of fig. 3 shows that the distributions for the documentation-specific classification and the generalised classification are different (compared to prior studies). As a sanity check we conducted a χ^2 test and calculated the effect size ϕ_c. We were able to reject the null hypothesis for both classification schemes, that the distribution of issues were the same as the previous studies (see section 5). We now discuss the most prominent differences between our study and the previous studies.

In the context of intelligent service SO posts, Taxonomy B suggests that Errors (B-3) are discussed most amongst developers. These results are in contrast to similar studies made in more *mature* API domains, such as Mobile Development [5, 6, 9, 10, 32] and Web Development [38]. Here, API Usage (B-1) is much more frequently discussed, followed by Conceptual (B-5), Discrepancy (B-2) and Errors (B-3). We argue in the following section that an improved developer understanding can be achieved by educating them about the intelligent service lifecycle and the 'whole' system that wraps such services.

In the Android study API usage questions (B-1) were the highest category (28.93% compared to 22.29% in our study). As stated in the analysis of the Error questions this discrepancy could be due to the maturity of the domain. However, another explanation could be the scope of the two individual studies. Beyer et al. [9] used a broad search strategy consisting of posts tagged Android. This search term fetches issues related to the entire Android platform which is significantly larger than searching for computer vision APIs using 229 search terms. As a consequence of more posts and more APIs there would be use cases resulting in additional posts related to API Usage (B-1).

Applying existing SO taxonomies allowed us to better understand the distribution of the issues across different domains. In particular, the issues raised around intelligent services appear to be primarily due to poor documentation, or insufficient explanation around errors and limitations. Hence, many of the concerns could be addressed by adding more details to the end-point descriptions, and by providing additional information around how these services are designed to work.

6.2 The Developer's Learning Approach

In this subsection, we offer an explanation as to why developers are complaining about certain things when trying to use intelligent services on SO (RQ1), as characterised through the use of prior SO classification frameworks (RQ2). This is described through the theoretical lenses of two learning taxonomies: Bloom's context complexity and intellectual ability taxonomy, and the SOLO taxonomy (i.e., the nature by which developer's learn). We argue that the issues with using intelligent services relating to the lower-levels of these learning taxonomies are easily solvable by slight fixes and improvements to the documentation of these services. However, the higher dimensions of these taxonomies demand far more rigorous mitigation strategies than documentation alone (potentially more structured education). Thus, many of the questions posted are from developers who are *learning to understand* the domain of intelligent services and AI, and (hence) both SOLO and Bloom's taxonomies are applicable for this discussion—as described below within the context of our domain—as pedagogical aides.

6.2.1 Bloom's Taxonomy. The cognitive domain under Bloom's taxonomy [12] consists of six objectives. Within the context of intelligent services, developers are likely to ask questions due to causal links that exist in the following layers of Bloom's taxonomy: (i) *knowledge*, where the developer does not remember or know of the basic concepts of computer vision and AI (in essence, they may think that AI is as smart as a human); (ii) *comprehension*, where the developer does not understand how to interpret basic concepts, or they are mis-understanding how they are used in context; (iii) *application*, where the developer is struggling to apply existing concepts within the context of their own situation; (iv) *analysis*, where the developer is unable to analyse the results from intelligent services (i.e., understand response objects); (v) *evaluation*, where the developer is unable to evaluate issues and make use of best-practices when using intelligent services; and (vi) *synthesise*, where the developer is posing creative questions to ask if new concepts are possible with computer vision services.

6.2.2 SOLO Taxonomy. The SOLO taxonomy [11] consists of five levels of understanding. The causal links behind the SO questions we have found relate to the following layers of the SOLO taxonomy: (i) *pre-structural*, where the developer has a question indicating incompetence or has little understanding of computer vision; (ii) *uni-structural*, where the developer is struggling with one key aspect (i.e., a simple question about computer vision); (iii) *multi-structural*, where the developer is questioning multiple concepts (independently) to understand how to build their system (e.g., system integration with the intelligent service); (iv) *relational*, where the developer is comparing and contrasting the best ways to achieve something with intelligent services; and (v) *extended abstract*, where the developer poses a question theorising, formulating or postulating a new concept within intelligent services.

6.2.3 Aligning SO taxonomies to Bloom's and SOLO taxonomies. To understand our findings with the lenses of pedagogical aids, we aligned Taxonomies A and B to Bloom's and the SOLO taxonomies for a random sample of 50 issues described in section 4.3.1. To do this, we reviewed all 50 of these SO posted questions and applied both the Bloom and SOLO taxonomies. The primary author assigned each of the 50 questions a level within the Bloom and SOLO taxonomies, removed out noise (i.e., false positive posts of no relevance to intelligent services) and unassigned dimensions from reliability agreement, and then compared the relevant dimensions of Taxonomy A and B dimensions (not sub-categories). The comparison of alignments of posts to the five SOLO dimensions and

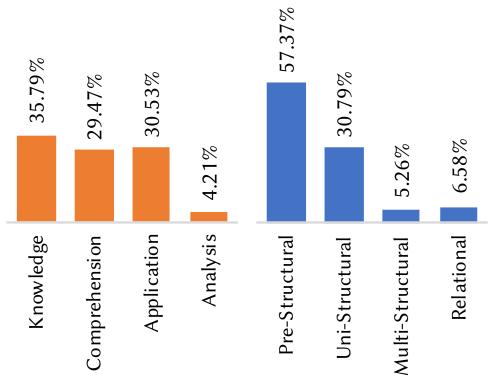


Figure 4: Alignment of Bloom (Orange) and SOLO (Blue) taxonomies against Taxonomy A and B dimensions against all 213 classifications made in the random sample of 50 posts.

six Bloom dimensions are shown in fig. 4. We acknowledge that this is only an approximation of the current state of the developer's understanding of intelligent services. This early model will require further studies to perform a more thorough analysis, but we offer this interpretation for early discussion.

As shown in fig. 4, the bulk of the posts fall in the lower constructs of Bloom's and the SOLO taxonomy. This indicates that modification to certain documentation aspects can address many of these issues. For example, many issues can be ratified with better descriptions of response data and error messages: "*I was exploring google vision and in the specific function 'detectCrops', gives me the crop hints. what does this means exactly?*" [88]; "*I am a making a very simple API call to the Google Vision API, but all the time it's giving me error that 'google.oauth2' module not found.*" [89]

However, and more importantly, the higher-construct questions ranging from the middle of the third dimensions on are not as easily solvable through improved documentation (i.e., apply and multi-structural) which leaves 34.74% (Bloom's) and 11.84% (SOLO) unaccounted for, resolvable only through improved education practices.

6.3 Implications

6.3.1 Researchers. (i) Investigate the evolution of post classification: Analysing how the distribution of the reported issues changes over time would be an important study. This study could answer questions such as '*Does the evolution of intelligent services follow the same pattern as previous software engineering trends such as mobile app or web development?*' As with any new emerging field, it is key to analyse how developers perceive such issues over time. For instance, early issues with web or mobile app development matured as their respective domain matured, and we would expect similar results to occur in the intelligent services space. Future researchers could plan for a longitudinal study, such as a long-term survey with developers to gather their insights in this evolving domain, reviewing case studies of projects that use intelligent web services from now into the future, or re-mining SO at a later date and comparing the results to this study. This will help assess evolving trends and characteristics, and determine how and if the nature of the developer's experience with intelligent services (and AI in general) changes with time. *(ii) Investigate the impact of technical challenges*

Table 2: Example Alignments of Stack Overflow posts to Bloom's and SOLO taxonomy.

Issue Quote	Bloom	SOLO
"*I'm using Microsoft Face API for a small project and I was trying to detect a face inside a .jpg file in the local system (say, stored in a directory* `D:\Image\abc.jpg`*)... but it does not work.*" [83]	Knowledge	Pre-Structural
"*The problem is that the response JSON is rather big and confusing. It says a lot about the picture but doesn't say what the whole picture is of (food or something like that).*" [84]	Comprehension	Uni-Structural
"*The bounding box around individual characters is sometimes accurate and sometimes not, often within the same image. Is this a normal side-effect of a probabilistic nature of the vision algorithm, a bug in the Vision API, or of course an issue with how I'm interpreting the response?*" [85]	Comprehension	Multi-Structural
"*I'm working on image processing. So far Google Cloud Vision and Clarifai are the best API's to detect objects from images and videos, but both API's doesn't support object detection from 360 degree images and videos. Is there any solution for this problem?*" [86]	Application	Uni-Structural
"*Before I train Watson, I can delete pictures that may throw things off. Should I delete pictures of: Multiple dogs, A dog with another animal, A dog with a person, A partially obscured dog, A dog wearing glasses, Also, would dogs on a white background make for better training samples? Watson also takes negative examples. Would cats and other small animals be good negative examples?*" [87]	Analysis	Relational

on API usage: As discussed above, intelligent services have characteristics that may influence API usage patterns and should be investigated as a further avenue of research. Further mining of open source software repositories that make use of intelligent services could be assessed, thereby investigating if API patterns evolve with the rise of AI-based applications.

6.3.2 Educators. (i) Education on high-level aspects of intelligent services: As demonstrated in our analysis of their SO posts, many developers appear to be unaware of the higher-level concepts that exist within the AI and ML realm. This includes the need to pre- and post-process data, the data dependency and instability that exists in these services, and the specific algorithms that empower the underlying intelligence and hence their limitations and characteristics. However, most developers don't seem to complain about these factors due to the lack of documentation (i.e., via Taxonomy A). Rather, they are unaware that such information should be documentation and instead ask generalised and open questions (i.e., via Taxonomy B). Thus, documentation improvements alone may not be enough to solve these issues. This results in uncertainty during the preparation and operation (usage) of such services. Such high-level conceptual information is currently largely missing in developer documentation for intelligent services. Furthermore, many of the background ML and AI algorithm information needed to understand and use intelligent systems in context are built within data science (not SE) communities. A possible road-map to mitigate this issue would be the development of a software engineer's 'crash-course' in ML and AI. The aim of such a course would encourage software engineers to develop an appreciation of the nuances and the inherent risks and implications that comes with using intelligent services. This could be taught at an undergraduate level to prepare the next generation of developers of a 'programming 2.0' era. However, the key aspects and implications that are presented with AI would need to be well-understood before such a course is developed, and determining the best strategy to curate the content to developers would be best left to the SE education domain. Further investigation in applying educational taxonomies in the area (such as our attempts to interpret our findings using Bloom's and the SOLO taxonomies) would need to be thoroughly explored beforehand.

6.3.3 Software Engineers. (i) Better understanding of intelligent API contextual usage: Our results show that developers are still learning to use these APIs. We applied two learning perspectives to interpret our results. In applying the two pedagogical taxonomies to our findings, we see that most issues seem to fall into the pre-structural and knowledge-based categories; little is asked of higher level concepts and a majority of issues do not offer complex analysis from developers. This suggests that developers are struggling as they are unaware of the vocabulary needed to actually use such APIs, further reinforcing the need for API providers to write overview documentation (as noted in prior work [15]) and not just simple endpoint documentation. This said, improved documentation isn't always enough—as suggested by our discussion in section 6.2, software engineers should explore further education to attain a greater appreciation of the nuances of ML when attempting to use these services.

6.3.4 Intelligent Service Providers. (i) Clarify use cases for intelligent services: Inspecting SO posts revealed that there is a level of confusion around the capabilities of different intelligent services. This needs to be clarified in associated API documentation. The complication with this comes with targeting the documentation such that software developers (who are untrained in the nuances of AI and ML as per section 6.3.2) can to digest it and apply it in-context to application development. *(ii) Technical domain matters:* More needs to be provided than a simple endpoint description as conventional APIs offer by describing the whole framework by which the endpoint sits, giving further context. This said, compared to traditional APIs, we find that developers complain less about the documentation and more about shallower issues. All expected pre-processing and post-processing needs to be clearly explained. A possible mitigation to this could be an interactive tutorial that helps developers fully understand the technical domain using a hands-on approach. For example, websites offer interactive Git tutorials[7] to

[7] For example, https://learngitbranching.js.org.

help developers understand and explore the technical domain matters under version control in their own pace. *(iii) Clarify limitations:* API developers need to add clear limitations of the existing APIs. Limitations include list of objects that can be returned from an endpoint. We found that the cognitive anchors of how existing, conventional API documentation is written has become 'ported' to the computer vision realm, however a lot more overview documentation than what is given at present (i.e., better descriptions of errors, improved context of how these systems work in etc.) needs to be given. Such documentation could be provided using interactive tutorials.

7 THREATS TO VALIDITY

Construct validity: Some questions extracted from SO produced false positives, as mentioned in sections 4.2.1, 4.2.3 and 5. However, all non-relevant posts were marked as noise for our study, and thus did not affect our findings. Moreover, SO is known to have issues where developers simply ask basic questions without looking at the actual documentation where the answer exists. Such questions, although down-voted, were still included in our data-set analysis, but as these were so few, it does not have a substantial impact on categorised posts.

Internal validity: As detailed in section 4.3.1, Taxonomies A and B present slight and fair agreement, respectively, when inter-rater reliability was applied. The nature of our disagreements largely fell due to the subjectivity in applying either taxonomies to posts. Despite all coders agreeing to the shared interpretation of both taxonomies, both taxonomies are subjective in their application, which was not reported by either Aghajani et al. or Beyer et al.. In many cases, multi-label classification seemed appropriate, however both taxonomies use single-label mapping which we find results in too much subjectivity. This subjectivity, therefore, ultimately adversely affects IRR analysis. Thus, a future mitigation strategy for similar work should explore multi-label classification to avoid this issue; Beyer et al., for example, plan for multi-label classification as future work. However, these studies would need to consider the statistical challenges in calculating multi-rater, multi-label IRR for thorough reliability analysis in addressing subjectivity. The selection of SO posts used for our labelling, chiefly in the subjectivity of our classifications, is of concern. We mitigate this by an extensive review process assessing the reliability of our results as per section 4.3.1. The classification of our posts into the SOLO and Bloom's taxonomies was performed by the primary author only, and therefore no inter-rater reliability statistics were performed. However, we used these pedagogy related taxonomies as a lens to gain an additional perspective to interpret our results. Future studies should attempt a more rigorous analysis of SO posts using Bloom's and SOLO taxonomies. We only aligned posts to one category for each taxonomy and did not align these using multi-label classification. This brings more complexity to the analysis, and our attempts to repeat prior studies' methodologies (see section 3). Multi-label classification for intelligent services SO posts is an avenue for future research.

External validity: While every effort was made to select posts from SO relevant to computer vision services, there are some cases where we may have missed some posts. This is especially due to the case where some developers mis-reference certain intelligent services under different names (see section 4.2.1). Our SOLO and Bloom's taxonomy analysis has only been investigated through the lenses of intelligent services, and not in terms of conventional APIs (e.g., Andriod APIs). Therefore, we are not fully certain how these results found would compare to other types of APIs. Two *existing* SO classification taxonomies were used rather than developing our own. We wanted to see if previous SO taxonomies could be applied to intelligent services before developing a new, specific taxonomy, and these taxonomies were applied based on our interpretation (see section 4.2.4) and may not necessarily reflect the interpretation of the original authors. Moreover, automated techniques such as topic modelling were not utilised as we found these produce descriptive classifications only (see section 3). Hence, manual analysis was performed by humans to ensure categories could be aligned back to causal factors. Only English-speaking intelligent services were selected; the applicability of our analysis to other, non-English speaking services may affect results. Use of computer vision in this study is an illustrative example to focus on one area of the intelligent services spectrum. While our narrow scope helps us obtain more concrete findings, we suggest that wider issues exist in other intelligent service domains may affect the genralisability of this study, and suggest future work be explored in this space.

8 CONCLUSIONS

Intelligent services, such as computer vision services, offer powerful capabilities that can be added into the developer's toolkit via simple RESTful APIs. However, certain technical nuances of computer vision become abstracted away. We note that this abstraction comes at the expense of a full appreciation of the technical domain, context and proper usage of these systems. We applied two recent existing SO classification taxonomies (from 2018 and 2019) to see if existing taxonomies are able to fully categorise the types of complaints developers have. Intelligent services have a diverging distribution of the types of issues developers ask when compared to more mature domains (i.e., mobile app development and web development). Developers are more likely to complain about shallower, simple debugging issues without a distinct understanding of the AI algorithms that actually empower the APIs they use. Moreover, developers are more likely to complain about the completeness and correctness of existing intelligent service documentation, thereby suggesting that the documentation approach for these services should be reconsidered. Greater attention to education in the use of AI-powered APIs and their limitations is needed, and our discussion offered in section 6.2 motivates future work in resolving these issues in the SE education space.

ACKNOWLEDGEMENTS

We acknowledge Zac Brannelly, Fayaz Beigh, and Vivian Dabre for their assistance in categorising posts.

REFERENCES

[1] Emad Aghajani, Csaba Nagy, Olga Lucero Vega-Márquez, Mario Linares-Vásquez, Laura Moreno, Gabriele Bavota, and Michele Lanza. 2019. Software documentation issues unveiled. In *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, Montreal, QC, Canada, 1199–1210.

[2] Md Ahasanuzzaman, Muhammad Asaduzzaman, Chanchal K Roy, and Kevin A Schneider. 2018. Classifying stack overflow posts on API issues. In *2018 IEEE*

25th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, Campobasso, Italy, 244–254.

[3] Miltiadis Allamanis and Charles Sutton. 2013. Why, when, and what: analyzing stack overflow questions by topic, type, and code. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, San Francisco, CA, USA, 53–56.

[4] Scott Barnett. 2018. *Extracting technical domain knowledge to improve software architecture*. Ph.D. Dissertation. Hawthorn, VIC, Australia.

[5] Scott Barnett, Rajesh Vasa, and John Grundy. 2015. Bootstrapping mobile app development. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2*. IEEE Press, Florence, Italy, 657–660.

[6] Scott Barnett, Rajesh Vasa, and Antony Tang. 2015. A Conceptual Model for Architecting Mobile Applications. In *25th Working IEEE/IFIP Conference on Software Architecture*. IEEE Computer Society, Montreal, QC, Canada, 105–114.

[7] Anton Barua, Stephen W Thomas, and Ahmed E Hassan. 2014. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empirical Software Engineering* 19, 3 (2014), 619–654.

[8] Ohad Barzilay, Christoph Treude, and Alexey Zagalsky. 2013. Facilitating Crowd Sourced Software Engineering via Stack Overflow. *Finding Source Code on the Web for Remix and Reuse* 37, 4 (2013), 289–308.

[9] Stefanie Beyer, Christian Macho, Martin Pinzger, and Massimiliano Di Penta. 2018. Automatically classifying posts into question categories on stack overflow. In *the 26th Conference*. ACM, Gothenburg, Sweden, 211–221.

[10] Stefanie Beyer and Martin Pinzger. 2014. A Manual Categorization of Android App Development Issues on Stack Overflow. In *2014 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE Computer Society, Victoria, BC, Canada, 531–535.

[11] John B. Biggs. 1987. Evaluating the Quality of Learning: The SOLO Taxonomy (Structure of the Observed Learning Outcome). *Management in Education* 1, 4 (1987), 20–20.

[12] Benjamin S Bloom. 1956. *Taxonomy of educational objectives. Vol. 1: Cognitive domain*. David McKay Co Inc., New York, NY, USA.

[13] Ram Chillarege, Inderpal S Bhandari, Jarir K Chaar, Michael J Halliday, Diane S Moebus, Bonnie K Ray, and Man-Yuen Wong. 1992. Orthogonal Defect Classification-A Concept for In-Process Measurements. *IEEE Trans. Softw. Eng.* 18, 11 (1992), 943–956.

[14] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46.

[15] Alex Cummaudo, Rajesh Vasa, and John Grundy. 2019. What should I document? A preliminary systematic mapping study into API documentation knowledge. In *13th International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, Porto de Galinhas, Recife, Brazil, 1–6.

[16] Alex Cummaudo, Rajesh Vasa, John Grundy, Mohamed Abdelrazek, and Andrew Cain. 2019. Losing Confidence in Quality: Unspoken Evolution of Computer Vision Services. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Cleveland, OH, USA, 333–342.

[17] Matthias Gamer, Jim Lemon, and Ian Fellows Puspendra Singh. 2012. irr: Various Coefficients of Interrater Reliability and Agreement. (2012).

[18] Kevin A Hallgren. 2012. Computing Inter-Rater Reliability for Observational Data: An Overview and Tutorial. *Tutorials in Quantitative Methods for Psychology* 8, 1 (Feb. 2012), 23–34.

[19] Daqing Hou and Lingfeng Mo. 2013. Content Categorization of API Discussions. In *2013 IEEE International Conference on Software Maintenance (ICSM)*. IEEE Computer Society, Eindhoven, Netherlands, 60–69.

[20] Magne Jorgensen, Tore Dybå, Knut Liestøl, and Dag I K Sjøberg. 2016. Incorrect results in software engineering experiments: How to improve research practices. *Journal of Systems and Software* 116 (2016), 133–145.

[21] Natalia Juristo and Omar S Gómez. 2012. Replication of Software Engineering Experiments. *Empirical Software Engineering and Verification* 7007, 5 (2012), 60–88.

[22] David Kavaler, Daryl Posnett, Clint Gibler, Hao Chen, Premkumar Devanbu, and Vladimir Filkov. 2013. Using and Asking: APIs Used in the Android Market and Asked about in StackOverflow. In *International Conference on Social Informatics*, Adam Jatowt, Ee-Peng Lim, Ying Ding, Asako Miura, Taro Tezuka, Gaël Dias, Katsumi Tanaka, Andrew Flanagin, and Bing Tian Dai (Eds.). Springer International Publishing, Kyoto, Japan, 405–418.

[23] Barbara A Kitchenham, Tore Dybå, and Magne Jorgensen. 2004. Evidence-Based Software Engineering. In *Proceedings of the th International Conference on Software Engineering*. IEEE Computer Society, Edinburgh, UK, 273–281.

[24] Klaus Krippendorff. 2004. *Content Analysis*. SAGE.

[25] J Richard Landis and Gary G Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (March 1977), 159–17.

[26] Richard J Light. 1971. Measures of response agreement for qualitative data: Some generalizations and alternatives. *Psychological Bulletin* 76, 5 (1971), 365–377.

[27] Diego Lo Giudice, Christopher Mines, Amanda LeClair, Rowan Curran, and Amy Homan. 2016. *How AI Will Change Software Development And Applications*. Technical Report.

[28] Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica* 22, 3 (2012), 276–282.

[29] Tomohiro Ohtake, Alex Cummaudo, Mohamed Abdelrazek, Rajesh Vasa, and John Grundy. 2019. Merging Intelligent API Responses Using a Proportional Representation Approach. In *Empirical Software Engineering and Verification*. Springer International Publishing, Cham, 391–406.

[30] Andres L Martinez Ortiz. 2017. Curating Content with Google Machine Learning Application Programming Interfaces. In *EIAPortugal*.

[31] RightScale Inc. 2018. *RightScale 2018 State of the Cloud Report*. Technical Report.

[32] Christoffer Rosen and Emad Shihab. 2016. What are mobile developers asking about? A large scale study using stack overflow. *Empirical Software Engineering* 21, 3 (2016), 1192–1223.

[33] Martin Shepperd. 2018. Replication studies considered harmful. In *the 40th International Conference*. ACM, Gothenburg, Sweden, 73–76.

[34] David J Sheskin. 2003. *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC, New York, NY, USA.

[35] Ryan Stevens, Jonathan Ganz, Vladimir Filkov, Premkumar Devanbu, and Hao Chen. 2013. Asking for (and about) permissions used by Android apps. *Proceedings of the 10th Working Conference on Mining Software Repositories* (2013), 31–40.

[36] Shuyan Sun, Wei Pan, and Lihshing Leigh Wang. 2010. A comprehensive review of effect size reporting and interpreting practices in academic journals in education and psychology. *Journal of Educational Psychology* 102, 4 (2010), 989.

[37] Amjed Tahir, Aiko Yamashita, Sherlock Licorish, Jens Dietrich, and Steve Counsell. 2018. Can you tell me if it smells?: A study on how developers discuss code smells and anti-patterns in Stack Overflow. In *22nd International Conference on Evaluation and Assessment in Software Engineering (EASE)*. ACM, Christchurch, New Zealand, 68–78.

[38] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. 2011. How do programmers ask and answer questions on the web? (NIER track). In *33rd International Conference on Software Engineering (ICSE)*. ACM, Waikiki, Honolulu, HI, USA, 804–807.

[39] Gias Uddin and Foutse Khomh. 2019. Automatic Mining of Opinions Expressed About APIs in Stack Overflow. *International Software Engineering Research Network* PP, 99 (Feb. 2019), 1–1.

[40] Gias Uddin and Martin P Robillard. 2015. How API Documentation Fails. *IEEE Software* 32, 4 (June 2015), 68–75.

[41] Wei Wang and Michael W Godfrey. 2013. Detecting API usage obstacles: a study of iOS and Android developer questions. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, San Francisco, CA, USA, 61–64.

ONLINE ARTEFACTS

[42] "ICSE 2020 Submission #564 Supplementary Materials," http://bit.ly/2Z8zOKW.

[43] "Vision API topicality and score always the same," http://bit.ly/2TD5As2.

[44] "Meaning of score in Microsoft Cognitive Service's Entity Linking API," http://bit.ly/2TD9vVw.

[45] "Google Vision API does not recognize single digits," http://bit.ly/31Ws1Nj.

[46] "Google Cloud Vision - Numbers and Numerals OCR," http://bit.ly/31P07mi.

[47] "Human body part detection in Android," http://bit.ly/31T5pxd.

[48] "Is there a full list of potential labels that Google's Vision API will return?" http://bit.ly/2KNnJSB.

[49] "Vision API - Image Content Analysis Âă|Âă Cloud Vision API Âă|Âă Google Cloud," http://bit.ly/2TD9mBs.

[50] "Amazon Rekognition," https://amzn.to/2TyT2BL.

[51] "Image Processing with the Computer Vision API | Microsoft Azure," http://bit.ly/2YqhkS6.

[52] "Watson visual recognition," https://ibm.co/2TBNIO4.

[53] "Detecting labels in an image," http://bit.ly/2UlkW9K.

[54] "Clarifai," http://bit.ly/2TB3kSa.

[55] "Image Recognition API & Visual Search Results," http://bit.ly/2UmNPCw.

[56] "Image Recognition API | DeepAI," http://bit.ly/2TBNYgf.

[57] "Imagga - powerful image recognition apis for automated categorization & tagging in the cloud and on-premises," http://bit.ly/2TxsyRe.

[58] "Image recognition - talkwalker," http://bit.ly/2TyT7W5.

[59] "Megvii," http://bit.ly/2WJYFzk.

[60] "Tuputech," http://bit.ly/2uF4IsN.

[61] "Yitu technology," http://bit.ly/2uGvxgf.

[62] "Sensetime," http://bit.ly/2WH6RjF.

[63] "Deepglint," http://bit.ly/2uHHdPS.

[64] "aws Rekognition not initializing on iOS," http://bit.ly/31UeqG9.

[65] "Are the Cloud Vision API limits in documentation correct?" http://bit.ly/31SsNLg.

[66] "Multiple Google Vision OCR requests at once?" http://bit.ly/31P09dU.

[67] "AWS Rekognition PHP SDK gives invalid image encoding error," http://bit.ly/31Sgpec.

[68] "How to set up AWS mobile SDK in iOS project in Xcode," http://bit.ly/31St2pE.

[69] "Google Api Vision, ""before_request"" error," http://bit.ly/31Z27Zt.

[70] "Google Cloud Vision fails at batch annotate images. Getting Netty Shaded ClosedChannelException error," http://bit.ly/31UrBH9.

[71] "Amazon Rekognition Image caption," http://bit.ly/31P08qm.
[72] "Can Google Cloud Vision generate labels in Spanish via its API?" http://bit.ly/31UcBsY.
[73] "Specify language for response in Google Cloud Vision API," http://bit.ly/31SsUGG.
[74] "Google Vision Accent Character Set NodeJs," http://bit.ly/31TsVdp.
[75] "How to get a position of custom object on image using vision recognition api," http://bit.ly/3210Q49.
[76] "similar face recognition using google cloud vision API in android studio," http://bit.ly/31WhMZy.
[77] "Text extraction - line-by-line," http://bit.ly/31Yc21s.
[78] "Getting wrong text sequence when image scanned by offline google mobile vision API," http://bit.ly/31Sgr5O.
[79] "How can we find an exhaustive list (or graph) of all logos which are effectively recognized using Google Vision logo detection feature?" http://bit.ly/31Z27IX.
[80] "How to confidently validate object detection results returned from Google Cloud Vision," http://bit.ly/31UcCNy.
[81] "Google Vision API text detection strange behaviour - Javascript," http://bit.ly/31Ucyxi.
[82] "All GoogleVision label possibilities?" http://bit.ly/31R4FZi.
[83] "Adding a local path to Microsoft Face API by Python," http://bit.ly/2KLeMt3.
[84] "google cloud vision category detecting," http://bit.ly/31Uf32t.
[85] "Google Cloud Vision OCR API returning incorrect values for bounding box/vertices," http://bit.ly/31UeZjf.
[86] "Google cloud Vision and Clarifai doesn't Support tagging for 360 degree images and videos," http://bit.ly/31StuEm.
[87] "Image Selection for Training Visual Recognition," http://bit.ly/31W8lcw.
[88] "Can i give aspect ratio in Google Vision api?" http://bit.ly/2KSJwsp.
[89] "Google Vision API: ModuleNotFoundError: module not found 'google.oauth2'," http://bit.ly/31VlZfU.