

A Case for Software Specific Natural Language Techniques

Dave Binkley, Dawn Lawrie
Loyola University Maryland
Baltimore, MD, USA
{binkley,lawrie}@cs.loyola.edu

Abstract—For over two decades, software engineering (SE) researchers have been importing tools and techniques from information retrieval (IR). Initial results have been quite positive. For example, when applied to problems such as feature location or re-establishing traceability links, IR techniques work well on their own, and often even better in combination with more traditional source code analysis techniques such as static and dynamic analysis.

However, recently there has been growing awareness among SE researchers that IR tools and techniques are designed to work under a different set of assumptions than those that hold for a software system. Thus it may be beneficial to consider IR inspired tools and techniques that are specifically designed to work with software. One aim of this work is to provide quantitative empirical evidence in support of this observation. To do so a new technique is introduced that captures the level of difficulty found in an *information need*, the true, often latent, information that a searcher desires to know. The new technique is used to compare two test collections: the natural language TREC 8 collection and the software engineering JabRef collection. Analysis of the data leads to three significant findings. First, the variation in difficulty of the SE information needs is much larger than that of the natural language information needs; second, the most challenging of the SE information needs is far easier than the least challenging of the natural language information needs; and finally, variations of the queries used to uncover a latent information need have far less impact in the natural language collection than in the software engineering collection.

I. INTRODUCTION

The last ten to twenty years have seen a dramatic increase in the application of information retrieval (IR) and natural language processing techniques to source code. One result of this interest has been an increased diversity of tools and techniques caused by the addition of those that consider the natural language present in software [1], [2], [3], [4], [5], [6], [7], [8], [9]. For the most part, these tools and techniques have imported the underlying technology from IR as a black box. Although occasionally they are customized for use in software engineering by, for example, augmenting them with traditional source code analysis such as static and dynamic analysis.¹

An emerging direction for future software engineering (SE) research is to explore the impact of the underlying assumptions used during the initial creation of IR tools and

techniques. For example, IR often equates word frequency with word importance, but the number of occurrences of the variable `tmp` is not necessarily indicative of its importance. The negative impacts of such differences manifest themselves when off-the-shelf IR tools are applied in the software domain.

One way to witness these differences is through the consideration of test collections from the two domains. Generating a high-quality test collection is a non-trivial task. The IR community invests considerable effort curating test collections such as the annual TREC collections. In contrast, SE collections (e.g., the SEMERU collection) are often automatically scraped from repositories such as those maintained by git. While automatic scraping provides an efficient mechanism to construct large collections, their quality is hard to ensure. This challenge has been noted informally during many conference discussions. To date, neither the IR community nor the SE community has developed a way to empirically assess or compare the complexity of the search tasks that make up a test collection.

This paper introduces such a technique, which assesses the challenge and quality found in both the individual search tasks as well as that of an entire test collection. The technique began as a search for the ideal query. However, that search quickly becomes intractable. Fortunately, using the same basic approach, it is possible to describe the *distribution* of query qualities and complexities for a given information need. Thus, the distribution of observed performances over a set of randomly generated queries informs the challenge level of an information need. By aggregating over the information needs, it is possible to assess the quality of an entire test collection.

To this end, this paper introduces *Information Need Analysis* (INA) and the first techniques implementing it, INAT. INAT relies on the assumption that the relevant documents for a search task accurately capture the searcher’s information need. Consider, for example, feature location where the information need is to find the pieces of the code that implement a given feature. Source code methods (or functions) involved capture the information need. In a sense they form a much more concise representation of what a searcher is looking for than, for example, a bug report. This problem is magnified when the bug report is written by someone unfamiliar with the code, as is typically the case. This observation also holds in the natural language context where assessors are forced to interpret a description in order to determine document relevance. This interpretation aims to capture the “latent” information need, while the documents marked as relevant directly capture it.

¹This is the author’s version of the work. It is posted here for your personal use. Not for redistribution. The definitive version was published by and is copyright by IEEE.

Working from the relevant documents associated with an information need, the possible vocabulary used to search for these documents is made up of the words found in the relevant documents themselves. Given this vocabulary, INAT begins by sampling the vocabulary to generate a set of queries. Each of these queries is then submitted to a search engine, which produces a score (e.g., the average precision) capturing how well the query uncovers the relevant documents. INAT also facilitates an exploration of how different query characteristics impact query performance. Thus INAT provides an overall assessment of how challenging an information need is, which is used to explore four research questions. The first research question establishes the viability of the technique, while the subsequent questions investigate the use of the technique to gain insights into particular information needs as well as sets of information needs. In particular, INAT is used to investigate the following four research questions:

- Does INAT yield sufficient variation within and between test collections for it to be a viable approach in the study of test collection challenge?
- Do high quality queries contain high quality sub-queries?
- Is it possible to identify words as particularly helpful or unhelpful?
- Do test collections from different domains differ?

The remainder of this paper first provides background in Section II. Section III then explains the Information Need Analysis Technique. This is followed in Section IV with the experimental setup and then in Section V with the analysis relative to the research questions. Finally, Section VI reviews related work and the paper concludes in Section VII.

II. BACKGROUND

This section provides background on the IR concept of an *information need* and the way in which it can be captured using a test collections. It then introduces two representative test collections. Information Retrieval separates the concept of an *information need*, the often latent topic about which a searcher desires to know more, from that of a *query*, the searcher's (often lacking) attempt to express her particular information need [10]. Unfortunately, while some information needs have obvious queries (e.g., which football team won the last world cup), others are more challenging. As a result, a query can be a rather poor expression of an information need. Obviously, the ability of a search engine to retrieve documents that address a given information need is to some extent dependent on how well that need is expressed by the query.

In general, an information need is a nebulous unknown. However, when evaluating and comparing information retrieval techniques, researchers often utilize test collections. From such a collection, information needs can be easily represented. Each test collection includes a set of documents, a set of *topics*, and, for each topic, a set of relevant documents, which is a subset of the overall document set. The topics are natural language descriptions of the searcher's needs. In

this context, the information need associated with a topic can be captured by the set of relevant documents, because, by definition, these documents satisfy the information need by virtue of being marked relevant. Thus, the set of relevant documents forms an embodiment of the information need.

To form a test collection based on natural language documents, IR researchers typically ask human annotators to judge the ability of some set of documents from a corpus to address each of the collection's topics. For example, the National Institute of Standards and Technology (NIST) develops a new test collection each year as part of the Text REtrieval Conference (TREC) annual challenge [11]. The topics of this collection each consist of a title, a description, and a narrative. The title is a few words that describe the topic, while the description is a longer explanation. The narrative is written to assist the annotators and frequently describes what is not relevant to the topic. As an example, TREC Topic 405 is as follows:

<title> Cosmic Events

<desc> Description: What unexpected or unexplained cosmic events or celestial phenomena, such as radiation and supernova outbursts or new comets, have been detected?

<narr> Narrative: New theories or new interpretations concerning known celestial objects made as a result of new technology are not relevant.

Annotators judge a subset of the documents as either relevant or non-relevant. For each topic, this subset is identified by manually retrieving documents and by pooling the output of the search engines participating in the annual TREC challenge [12].

Typical SE collections have a similar make up. For example, a collection for use in the evaluation of the feature location might consist of the source code, whose files, methods, or functions, form the documents of the collection, modification requests make up the search tasks, and oracle judgements that identify which files, methods, or functions were modified in the repository when the modification request was committed. In the remainder of this paper both information retrieval and software engineering information needs will be referred to as topics.

III. AN INFORMATION NEED ANALYSIS TECHNIQUE

The big picture goal of this research is to provide quantitative data in support of the need for software engineering specific IR tools and techniques. This paper describes two steps toward this goal by first introducing a new technique for assessing how challenging a set of information needs are and then by using this technique to compare an IR and an SE collection.

In short, INAT uses a random sample of queries to quantify the challenge of a given topic from a test collection. Aggregated over all the topics of the collection, INAT can quantify the challenge of the entire collection. In greater detail, to evaluate the challenge posed by a given information need, *IN*, all the words of *IN*'s relevant documents are combined to form a vocabulary *V* from which query words are randomly

drawn. The words of V are exactly those that should be used to form queries. This is because these are the only words that will increase the score of any relevant document in the corpus. A set of queries whose words are randomly drawn from V is used to study the challenge posed by IN by feeding each query from this set to a search engine and recording the resulting performance (e.g., average precision). The distribution of the performance values provides a measure of IN 's challenge. For example, if most randomly-generated queries perform well, then the information need is easy to satisfy (easy to find a high-performing query). Averaging over a set of information needs provides a difficulty measure for a test collection.

In greater detail, INAT takes three inputs: a corpus of documents C , an information need IN , and a maximum query length K . The third parameter ensures that the search space from which queries are drawn is finite. This is a reasonable limitation to impose provided that queries longer than K bring no additional information. In general as query length approaches infinity, a randomly drawn query of that length approach a uniform distribution of the words from V and thus becomes uninteresting in discrimination, which is the challenge inherent in an information need. Empirically, values as small as $K = 8$ are sufficient (i.e., going from $K = 8$ to $K = 16$ did not change the resulting distribution). Using the three inputs, the following steps are used to produce a set of *query, score* pairs for information need IN :

```

let  $V \leftarrow \bigcup_{d \in C_{IN}}, \text{words}(d)$ 
 $QS \leftarrow \emptyset$ 
for  $i \leftarrow 1$  to sample_size
  let  $k \leftarrow \text{random number in } [1..K]$ 
   $q \leftarrow \emptyset$ 
  for  $j \leftarrow 1$  to  $k$ 
     $q \leftarrow q \cup \{v\}$ , a word randomly chosen from  $V$ 
  let  $s \leftarrow \text{score}(q, IN, C)$ 
   $QS \leftarrow QS \cup \{(q, s)\}$ 

```

INAT's output is a set of query, score pairs. The distribution of the scores is used to provide information related to the IN . For example, a distribution that is skewed to the left indicates an IN that has fewer high-quality queries. In contrast, a distribution that is skewed to the right indicates an IN where most queries have good performance. In the analysis, the distribution's mean and standard deviation are used to summarize IN 's overall level of challenge.

IV. DESIGN

The design describes how INAT was implemented in order to investigate the research questions. In the experiments two corpora were chosen, one from software engineering (a feature location collection) and one from information retrieval (a TREC collection). The software engineering corpus was built from the software project JabRef and is part of the SEMERU collection [13]. The information retrieval corpus consists of Text Research Collections 4 and 5² minus the *Congressional*

Record documents [12]. This corpus includes documents from four sources: the *Financial Times*, the *LA Times*, the *Federal Register*, and the Foreign Broadcast Information Service.

The information needs used with each collection are from the relevant documents associated with each topic in the corpus. JabRef's topics include 31 modification requests each with an associated *fix set*, which are used as its relevant documents. These fix sets are the source code methods that were modified in the check-in associated with the modification request. TREC 8's topics include fifty topics (topics 401 to 450) each with an associated human identified relevant document set.

To determine K , the maximum length of the randomly drawn queries, the TREC 8 topic descriptions were utilized. For TREC 8 topics, 401-450, the maximum number of words that occur in any one topic's title and description is 33 words. This value is cushioned, by 20% and rounded to 40, giving the value used for K in the experiments.

To determine the sample size, different sample sizes were experimented with. There was a need to balance the time to generate the sample with sufficient data points to capture the true distribution. One way to verify a sufficient size is to generate multiple independent samples and look for consistency across them. The size of 20,000 was imperially arrived at by considering five samples for each of the 81 topics and visually comparing the differences, as is shown in Figure 1. In all cases the variation ranged from minor to non-existent.

Finally, to "execute" a query requires *configuring* a search engine and selecting an evaluation *measure*. A search engine configuration includes any preprocessing of the documents, the application of stopping and stemming during indexing, and the selection of a retrieval model which assigns a score to each document in order to place them in a ranked list. Then, an evaluation measure is applied to this list and the set of relevant documents. The result is a number that summarizes the performance attained on the query. For the experiments, the search engine configuration includes standard preprocessing of the documents at index time: standard natural language stopwords were used as the stoplist and the Krovetz stemmer was applied [14]. Documents were ranked using the query likelihood model with Dirichlet smoothing. In addition for the source code, identifiers were split using standard camel case and underscore rules as described by Enslen et al. [5] and additional programming language stop words such as the keywords "if" and "for" are used. This search-engine configuration has worked well in the past [15].

The second part of determining the score applies a measure to the ranked list. For this study *Average Precision* (AP) is used to score the ranked lists. AP is a standard measure in information retrieval where precision is calculated at the point where each relevant document occurs in the ranked list. These precision values are then averaged. Precision is computed as the total number of relevant documents in a set divided by the number of documents. AP was chosen because the score is dependent on all of the relevant documents, as compared to measures such as reciprocal rank (RR) where the score is entirely dependent on the first relevant document encountered.

²Distributed by the Linguistic Data Consortium

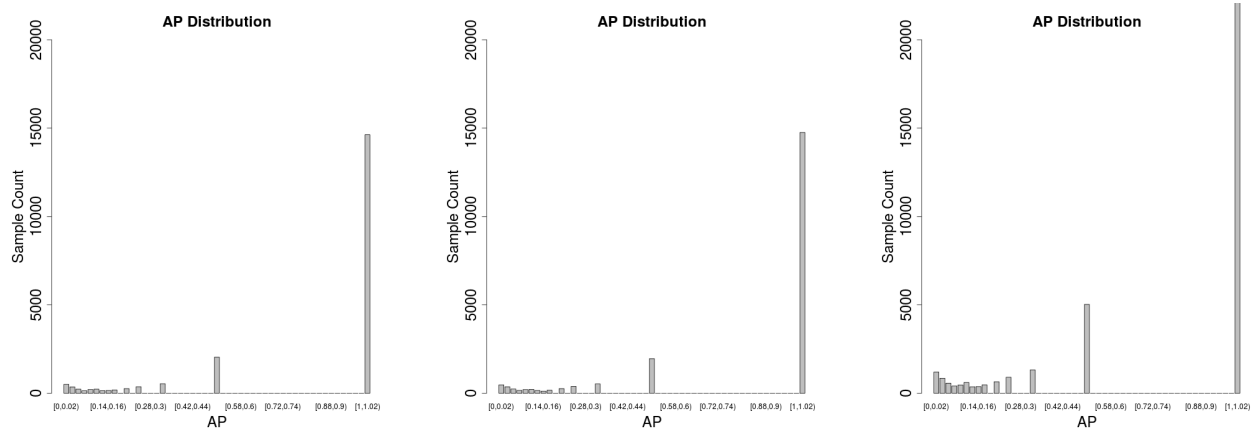


Figure 1. Visual illustration of the sufficiency of using a sample size of 20,000. The three samples are for **JabRef** topic 2905968. That 20,000 is sufficient is visually evident when histogram look virtually identical as is the case with the first two. The third histogram was generated using 50,000 queries rather than 20,000. Evidence here comes from the histogram having the same shape except that bars are 5/2 times taller than in the other two.

Averaging over a set of queries produces the *Mean Average Precision* (MAP).

Putting the pieces together, INAT, uses the distribution of AP values produced for a set of randomly drawn queries to provide an understanding of the challenge associated with an information need. For example, an *easy* query has a distribution skewed to the right with values close to one, indicating that the probability that a query produces a high AP value is larger than for a *hard* query where the distribution is skewed to the left with values close to zero. As a second example, when the standard deviation of the distribution is small then query modification is expected to have little impact on the resulting AP score. In contrast, when large, changes to a query (say through query reformulation [16]) have a much larger potential impact.

V. RESEARCH QUESTIONS

INAT is used to address four research questions, which first consider the challenge of different information needs before turning to the words that are present in different queries in the samples. Finally, the two test collections are compared head to head, which motivates the need for software specific approaches to natural language-based techniques.

A. Information Need Challenge

The first research question asks, “Does INAT yield sufficient variation within and between test collections for it to be a viable approach for studying test collection challenge?” A challenging, or *hard*, information need is one where the likelihood of generating a random query that has good retrieval performance is low, whereas an *easy* information need is one where the likelihood of generating a random query that has good retrieval performance is high. Evidence for information need challenge comes directly from the samples generated using INAT as described in Section III. To evaluate the data for the first research question, both qualitative and quantitative techniques are used. First, histograms of the AP score distribution visually shows where the probability mass lies. Second, these distributions are also described using their mean and standard deviation. For example, the means for a

set of *INs* induce an order of the information needs in terms of difficulty.

The challenge of an information need can be visualized by examining a histogram of the AP scores produced by its set of random queries. Figure 2 shows six examples. In each histogram the *x*-axis partitions the AP values into 50 groups of width 0.02, while the *y*-axis is a count of the number of the 20,000 samples that fall into each *x*-axis range. The top three are taken from the TREC collection and the bottom three from the JabRef collection. From left to right each line includes the hardest information need, the median, and finally the easiest. These graphs show considerable variation both within each collection and between the two collections. In particular, there is a great deal of variability in the JabRef information needs.

Numerically, for all topics in the TREC collection, the sample median is in the lowest range, 0 to 0.02. For the two least-challenging TREC topics, the mean makes it out of the lowest range, where it falls in the second lowest range. In contrast, for JabRef, roughly a quarter of the information needs have a median value of one! This indicates that for over half of the queries in these topics, all of the relevant documents are ranked ahead of all non-relevant documents. Finally, the mean and median of the most challenging JabRef topic is higher than the least challenging TREC topic. This can be seen visually by comparing the far right histogram in the top row with the far left histogram in the second row.

In summary for the first research question, the histograms clearly reveal a significant variation in the challenges of the information needs both within and between the two collections. For some topics, it is clearly quite challenging to randomly generate a query with good retrieval results, while for others it is quite easy. Thus INAT yields sufficient variation to be used in the assessment of test collection challenge.

B. Quality Terms

The second research question asks, “Do high quality queries contain high quality sub-queries?” This question explores the relation between the highest performing queries in the sample and the highest performing queries of a particular length. Asked another way, do words in the best performing

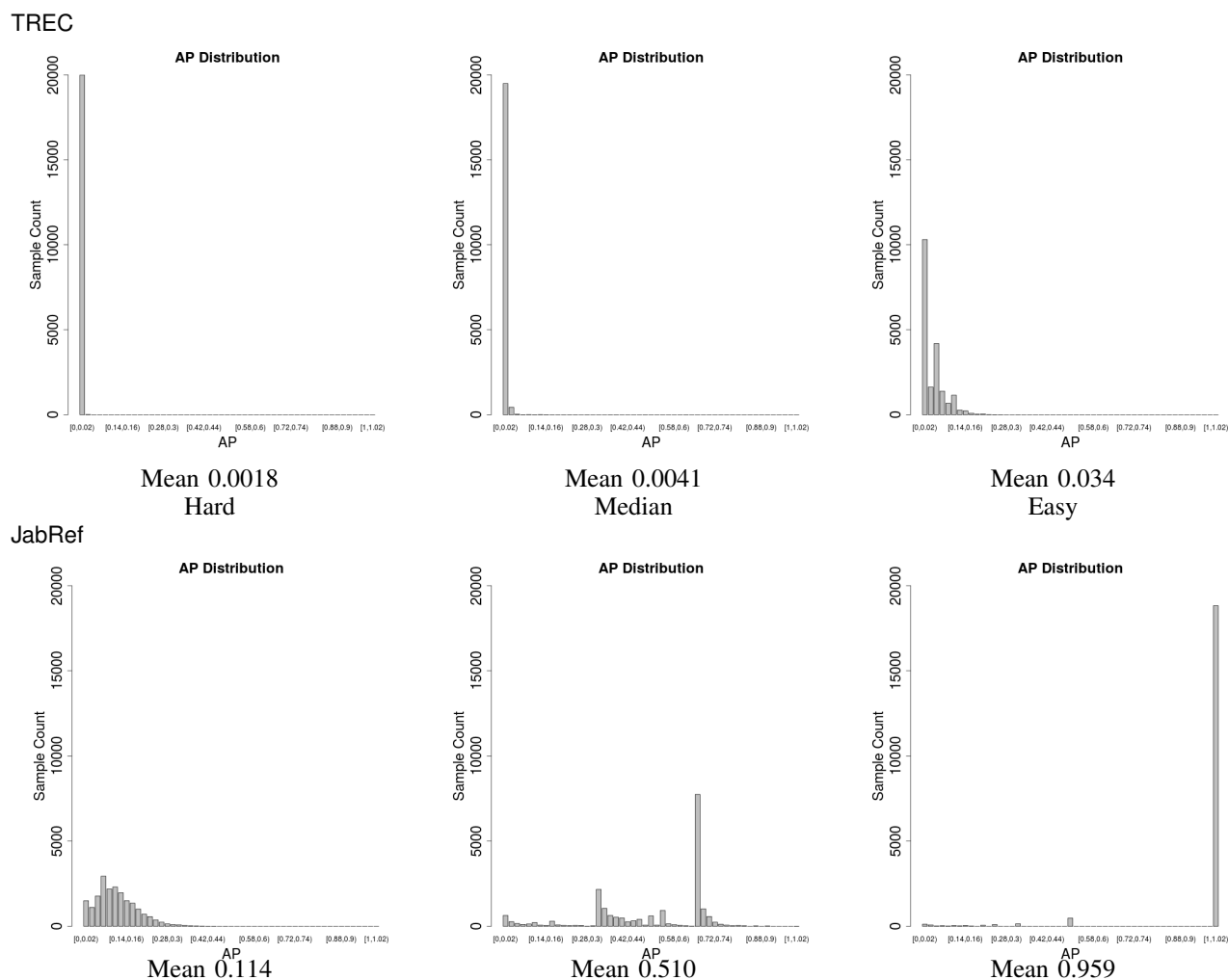


Figure 2. Histograms of the AP scores for the hardest, the median, and the easiest of the information needs found in the two collections.

short queries end up in the best overall queries? An affirmative answer would indicate that there are particular words that need to be present to support good retrieval.

The reason that this is an interesting research question is that the interaction between the words that occur when scoring a document relative to the document's rank is not straight forward. First, retrieval models tend to focus on positive information, rather than penalizing documents for negative information. Therefore, given a particular document, the score can only stay the same or improve as words are added. This means that two words that are present in the document will result in a higher score than either of the words independently. Thus, the document score is monotonically increasing as more and more words are added to the query.

The challenge comes from the fact that the evaluation uses document ranks, not their scores. Consequently, adding a word to the query can improve the score of a relevant document a little while improving the score of a non-relevant document by a lot. This may lead to non-relevant documents receiving higher scores, which has a negative impact on the overall AP value. Given this, examples exist where a six word query has a high AP score despite the fact that none of individual words performed particularly well independently.

This research question looks for trends. Although it is possible for high performing queries to consist of mediocre words when considered independently, it is not clear whether this is the case in practice.

Investigation of this research question examines the presence of the top-ten individual words, the top-ten pairs of words, and the top-ten triples of words in the one hundred highest performing queries from the sample. The queries are broken out in accumulating sets in two dimensions. The results of the data are visualized in 3D column charts appearing in Figure 3. Along the x -axis, the sub-queries are incrementally added to the set. Along the z -axis, the overall best queries in the sample are accumulated in groups of ten. Considering the first column, of interest is the proportion of best queries that contain the sub-query within the query. The data is reported as the proportion without the query, so a column at 40% means that 60% of the highest performing queries contain the sub-query. Considering the second column, the top two sub-queries are considered. In this column, a high performing query has overlap if it has either of the two sub-queries. Therefore, a column at 40% would mean that 60% of the queries contain one of the two sub-queries or both. Each column adds one more sub-query, so it is expected that the columns will become shorter as the distance increases away from the x -axis. In the

3D column chart figures, charts with all tall columns show the absence of sub-queries, whereas short or no columns represent the presence of sub-queries in many of the highest performing sampled queries.

This analysis is challenged by ties in the sub-query score. For identifying the top queries in the sample, it is not a particular concern because the sample is random. Thus the trends are important, and the queries should represent the patterns present in the underlying population. Ties are more problematic when determining which of the sub-queries to consider. This is particularly true when there are a large number of ties. To reflect the presence of high performing sub-queries in the sample, a greedy approach to selecting the top sub-queries is used. Thus, the sub-query that is most prevalent in the sampled set will be chosen ahead of sub-queries that are less prevalent in the sampled set.

Figure 3 shows the lack of overlap in representative topics. For the single-word queries, notice that the presence of sub-queries of length one is a lot higher in JabRef than TREC. In particular, notice that there is a lot more in common in the JabRef topics than the TREC topics. While the TREC topics demonstrate almost nothing in common as is seen in Topic 417, JabRef Topic 1553552 is an example of an information need that is dependent on only a few distinct vocabulary words. Virtually all queries contain at least one high-performing sub-query.

Now consider queries with two words. In order for a query consisting of a pair of words to overlap with a sample query, both words in the pair must be included in the sample query. This requirement is much more stringent; however, many JabRef queries contain at least one high performing pair query. This is seen in Figure 4 in JabRef Topic 1489454. Figure 4 contains the overall highest overlap for the two collections, the median topic based on average overlaps, and the overall least overlap. Although JabRef has a substantial amount of overlap, over half the TREC topics exhibit no overlap. TREC Topic 441 has the most overlap between the query pairs and the sampled queries, but the median and least exhibit no overlap as is seen in the bottom of two TREC topics in the figure.

A similar analysis was done using sub-queries of three words. The trends are similar to those seen using pairs of words. In this case, no TREC topic has a high performing query where all words in the query can be found in the sampled query, but JabRef Topics continue to contain high performing sub-queries. This analysis shows that identifying good performing short queries can be helpful with software engineering information needs, but such efforts appear to be ineffective for a natural language information needs.

C. Helpful or Not

The third research question asks, “Is it possible to identify words as particularly helpful or unhelpful?” In particular it is important to understand if particular query words have a significant negative impact on the performance of a query. For this research question, the top 100 and the bottom 100 queries were examined. The words appearing in the queries was taken as a set of vocabulary words. Given the set of words in the top queries and the set of words in the bottom queries, the

Jaccard similarity coefficient is computed [17]. By comparing Jaccard similarity coefficients, the similarity in the vocabulary among the two sets of queries can be determined.

First the sizes in the vocabularies are vastly different between the information needs coming from JabRef and the information needs coming from TREC. The mean vocabulary size for a JabRef information need is 173.1 words, while the mean for a TREC information need is 6755.0 words. Despite these differences in vocabulary size, the Jaccard similarity coefficients are not predictive of the source of the information need. That means that given a Jaccard similarity coefficient one cannot determine whether the information need came from JabRef or TREC as can be seen in Figure 5. Although the range in similarity for TREC topics is wider than for JabRef, the values do overlap for over half the range.

Looking at the variability in the words that appear in the best and worst queries is also very interesting. Figure 6 shows that all or almost all the vocabulary exist in the best queries for JabRef, indicating that the presence of a particular word does have too much of a negative impact on the performance. However, the opposite is true for TREC where a much smaller portion of the vocabulary is present in the best queries. The data in Figure 6 is ordered by Jaccard Similarity. Given the general downward trend, where the proportion of vocabulary is smaller there is less similarity between the best and worst sets of words.

Finally, Figure 7 looks at the proportion of overall vocabulary found in the queries that perform the worst. The data is again ordered by Jaccard similarity. Here, a greater portion of the vocabulary is represented for TREC. This means that there is a greater proportion of words in the TREC documents that have a negative impact of retrieval. The proportion of vocabulary also does not follow the similarity score as closely. This is represented by the jagged line. Turning to JabRef, it is not surprising that this line trends with the similarity since nearly all the vocabulary is found in the best queries. Thus, there are some words that are particularly helpful for software needs and lacking these words the query does not perform well.

D. Domain Comparison

The final research question asks, “Do test collections from different domains differ?” The final research question attempts to motivate the need for software specific IR-based tools and techniques. Using a case study of two collections, it provides quantitative data showing differences in the complexity of the information needs in a classic IR collection, TREC 8, and the JabRef software engineering collection.

To begin with, the histograms from Figure 2 visually hint at significant differences in the challenge presented by the case-study’s two collections. The entire range of TREC information needs are clearly skewed to the left indicating, among other things, that it is significantly more difficult to produce good queries for these information needs. In contrast, JabRef’s information needs are much more varied. The leftmost JabRef histogram bears some similarity to the right most TREC histogram, but there the similarity ends (perhaps other than noting that JabRef’s easiest information

Single Word

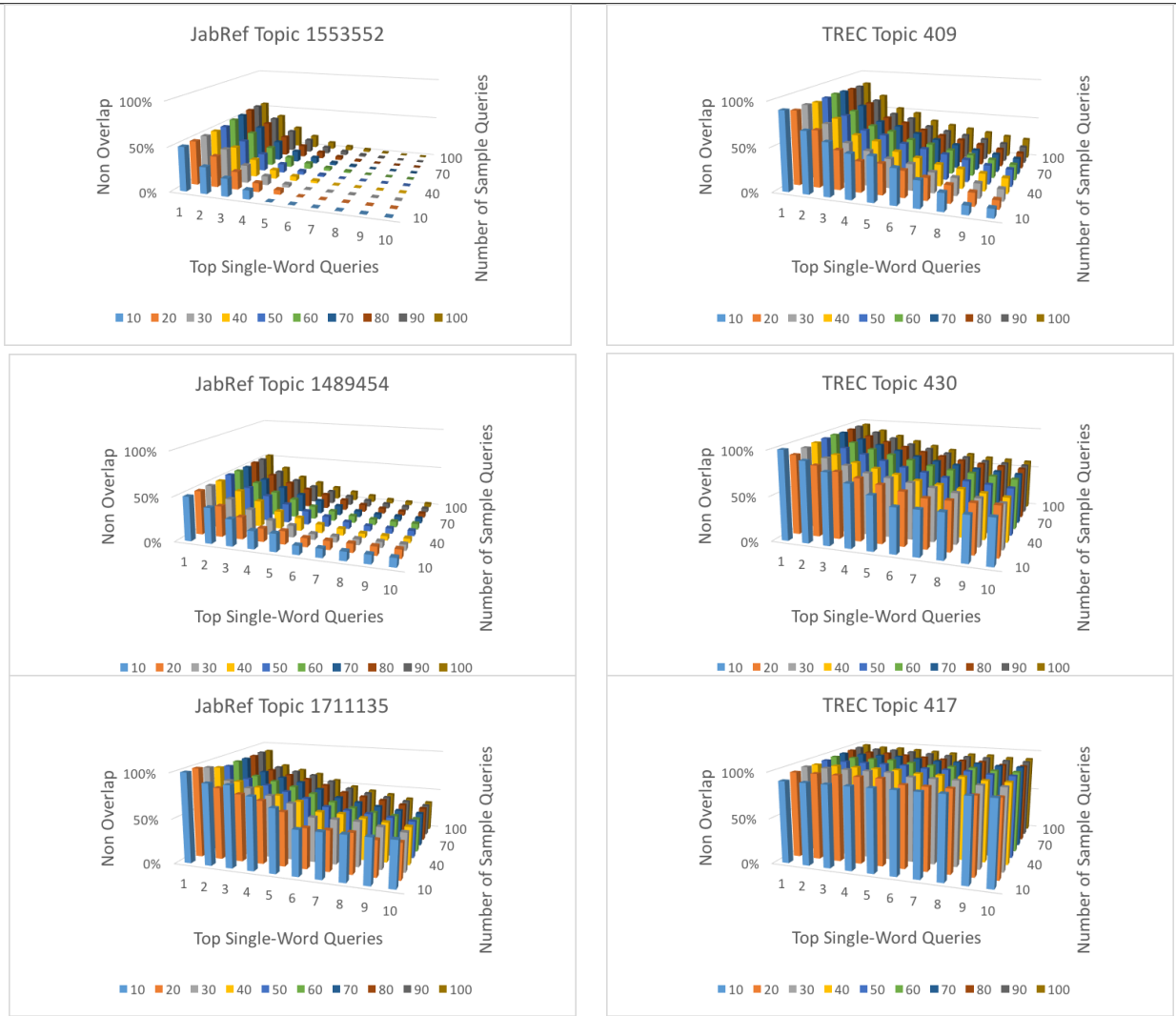


Figure 3. Percent of non-overlap between best single word queries and best queries in the sample.

need, shown at the far right, is a *mirror image* of the TREC's hardest).

To quantify these visual differences, the mean and standard deviation for each information need was computed. The overall mean for JabRef is almost 100 times greater than the mean for TREC. The individual means with whiskers representing the standard deviation, shown in Figure 8, clearly show the difference in the overall means. Furthermore, the standard deviations are much larger for the JabRef collection. On average the JabRef standard deviation is almost 25 times larger than that of the TREC collection.

Together these two values indicate a dramatic difference. Overall the topics of the TREC collection are much harder to do well on (they have much lower means) while at the same time their smaller standard deviation indicates that it is easier to find a query that produces average results. In contrast, the considerably higher means of the topics from the JabRef collection indicate that on an absolute scale the JabRef information needs are easier to retrieve relevant document for.

At the same time, the very high standard deviation means that performance is *very* dependent on the query.

One obvious implication of this comparison is the query reformulation techniques are of little value in IR when used on a collection such as the TREC collection but have great potential when used with an SE collections such as the JabRef collection. In other words, SE researchers should be investing effort in software-specific query reformulation techniques (much more so than IR researchers should be working on query reformulation).

Finally, to put statistical teeth behind the differences seen in Figures 2 and 8 all pair-wise t-tests were run (using the Bonferroni correction for multiple comparisons). One way to summarize the resulting 6480 t-tests, is to separately count, for each of the 81 information needs, the number of other information needs that are harder (their distribution has a statistically significantly lower mean). The easiest of the JabRef queries, shown in the lower right of Figure 2, is not harder than any of the other information needs. (There was one other for which the null hypothesis could not be rejected). That

Word Pairs

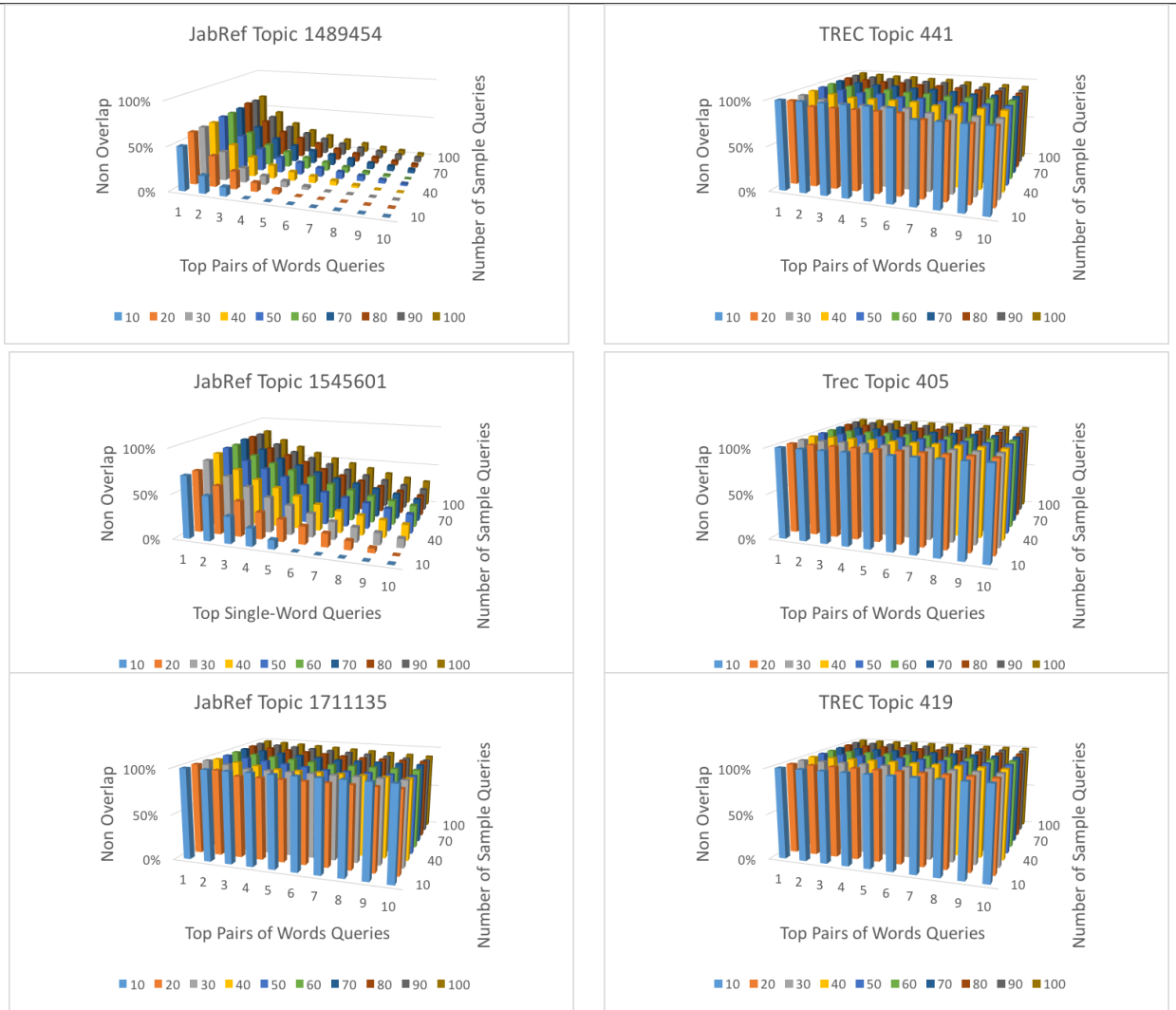


Figure 4. Percent of non-overlap between best pairs of words queries and best queries in the sample.

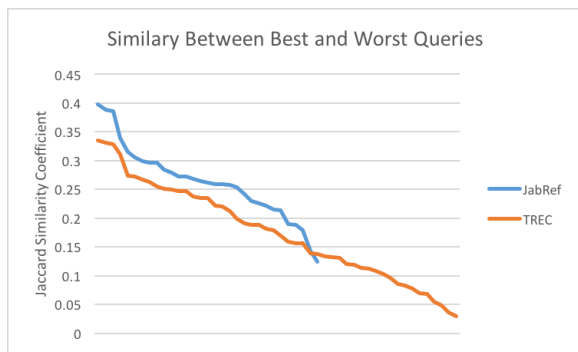


Figure 5. Jaccard Similarity Coefficient between the vocabulary in the best and the worst queries.

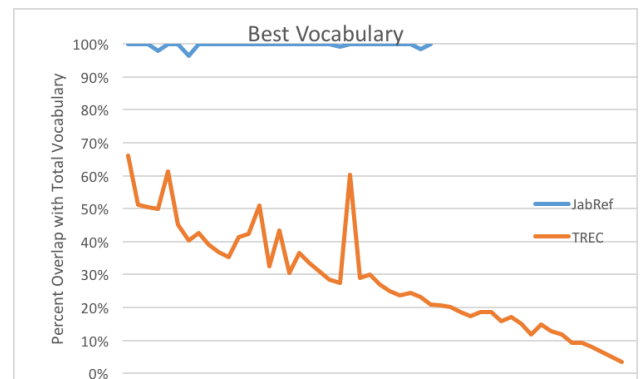


Figure 6. Proportion of the vocabulary found in the best queries.

leaves 79 that were harder. At the other end of the spectrum, the information need shown in the upper left of Figures 2 is harder than 41 of the other information needs while for the other 39 the null hypothesis cannot be rejected.

The most striking observation that can be extracted from the pairwise comparisons is that *all* of the TREC queries are significantly harder than *all* of the JabRef queries. While internally, JabRef shows much greater difficulty variation with 456 of the 465 comparisons showing a statistically

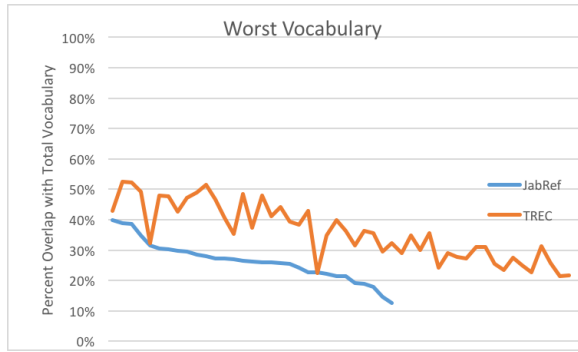


Figure 7. Proportion of the vocabulary found in the worst queries.

significant difference. In contrast, the performance on the TREC information needs is much more similar with only 315 of 1225 showing a difference.

The consideration of the fourth research question ends by considering actual queries based on the topic descriptions that accompany the two collections. That is, for both collections the topic descriptions are used as the queries after applying the standard preprocessing described in Section III. No advanced processing such as relevance feedback [16] or query reformulation [18] is performed. As before for both collections, the AP score is calculated for each query. The query for topic t is then compared to the histogram for topic t . Good queries will have an AP score in a high percentile, while poor queries will have an AP score in a low percentile.

The resulting percentile of each topic of the two collections are graphed in Figure 9. The query analysis again illustrates a significant difference between the two collections. While both collections include good and poor queries, the number of good queries is much higher with TREC. This is in part because of the TREC’s overall lower standard deviation. It also reflects TREC’s inclusion of human generated descriptions aimed at searching versus JabRef’s repurposed maintenance requests. In general there are a wealth of potential causes for these differences. Future work will consider how these differences inform the development of SE specific IR-based tools and techniques.

In answer to the fourth research question, there are clear and significant quantifiable differences between the two test collections used in the case study. While a larger study is warranted, the magnitude of the difference supports the existence of a significant difference in the collections used by the two domains and thus the great potential of SE specific IR research.

E. Threats to Validity

This is an initial study of a new evaluation approach and as such has several standard threats to its validity. For example, using a case study of two collections is sufficient to establish that a larger study is warranted, but the results still suffer from threats to their external validity. Internal validity is also a potential issue because INAT is a new technique and lacks the experience that help dissuade concerns about what is actually being measured by the approach. Statistically, well known statistical tests as implemented in R were used. The only open

question here is if 20,000 samples is sufficient. In addition to looking at the distribution for multiple samples, for a few of the queries a larger sample of 50,000 random queries was used. Again no differences were observed. Thus even with the larger IR corpus, the sampling appears sufficient.

VI. RELATED WORK

Although the authors are unaware of any work on analyzing information needs, there is some work on comparing test collections. This is most notable from the Text REtrieval Conference. In particular, when a new collection is created it is compared to prior collections as was done at TREC 8 [12]. TREC reports on such statistics as the words found in the topic descriptions and the source of relevant documents, whether it be from manual investigation or automatically identified by those that participated in the task. They may also report on the performance of a particular search engine over various years as was also done at TREC 8. By reporting MAP, it becomes clear that some sets of queries are easier than other sets of queries. Although it stands to reason that some information needs are easier than others, this has not been a focus of TREC evaluation of its collections.

Other researchers have investigated the application of other natural language techniques when applied to software. For instance, Hindle et al. [19] looked at whether statistical language models could be applied to software and found the the modeling technique worked well.

VII. CONCLUSION

This work introduces INAT, a novel approach to studying information need. INAT exploits queries constructed from random words drawn from the vocabulary. This supports the comparison of information needs as well as the sets of information needs that make up an information retrieval evaluation set.

INAT was used to explore similarities and differences in two evaluation sets, one from the SE domain and one from the IR domain. Results from the study of the research questions lead to the conclusion that the SE evaluation set as represented by the feature location task on the source code for JabRef is far easier than the IR evaluation set as represented by the TREC 8 ad-hoc retrieval task. Far more interesting is the comparison of how retrieval using the topic descriptions provided as part of the evaluation set is relatively good for the TREC 8 topics, while fairly poor for the JabRef topics.

These results support the need for research on source code specific natural language based tools. For example, unique approaches to query reformulation may help improve the performance on JabRef topics because the descriptions based on the maintenance requests frequently do not contain sufficient overlapping vocabulary with the relevant source code. This overlap is necessary to satisfy the information need. For example, techniques from IR such as relevance feedback cannot be used off-the-shelf when the documents being retrieved have so little in common with the typical query.

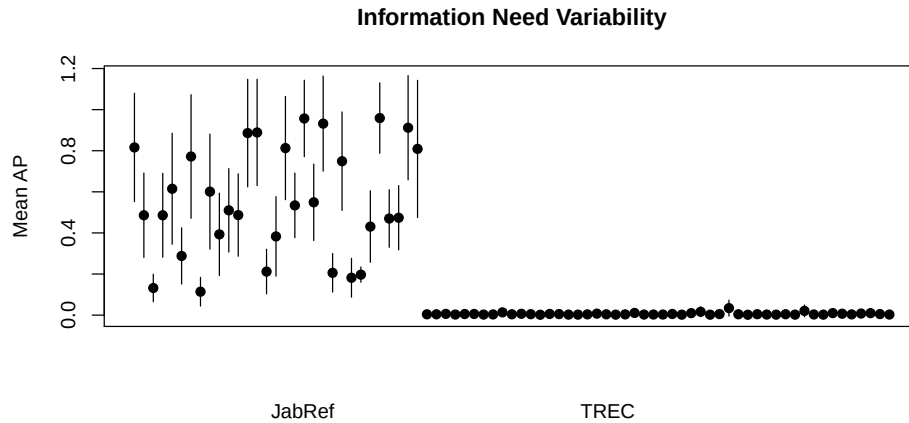


Figure 8. Comparison of non-overlap between best single word queries and best queries in the sample.

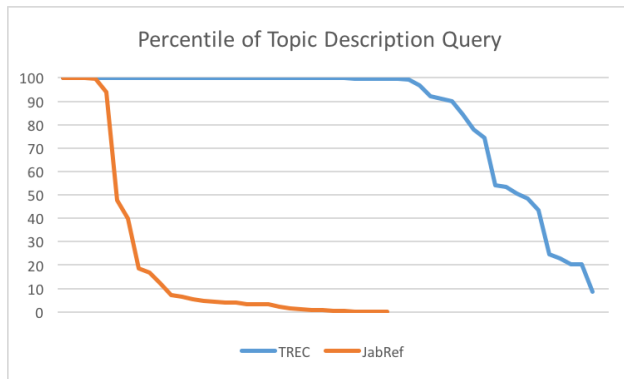


Figure 9. Percentiles of queries sorted by percentile.

REFERENCES

- [1] S. L. Abebe, S. Haiduc, P. Tonella, and A. Marcus, "Lexicon bad smells in software," in *2009 16th Working Conference on Reverse Engineering*. IEEE, 2009, pp. 95–99.
- [2] V. Arnaudova, L. M. Eshkevari, M. Di Penta, R. Oliveto, G. Antoniol, and Y.-G. Gueheneuc, "Repent: Analyzing the nature of identifier renamings," *IEEE Transactions on Software Engineering*, vol. 40, no. 5, pp. 502–532, 2014.
- [3] G. Bavota, A. De Lucia, and R. Oliveto, "Identifying extract class refactoring opportunities using structural and semantic cohesion measures," *Journal of Systems and Software*, vol. 84, no. 3, pp. 397–414, 2011.
- [4] A. De Lucia, M. Di Penta, and R. Oliveto, "Improving source code lexicon via traceability and information retrieval," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 205–227, 2011.
- [5] E. Enslin, E. Hill, L. Pollock, and K. Vijay-Shanker, "Mining source code to automatically split identifiers for software analysis," in *Proceedings of the 2009 Mining Software Repositories (MSR)*. IEEE, May 2009.
- [6] L. Guerrouj, "Automatic derivation of concepts based on the analysis of source code identifiers," *2013 20th Working Conference on Reverse Engineering (WCRE)*, vol. 0, pp. 301–304, 2010.
- [7] M. Pradel and T. R. Gross, "Detecting anomalies in the order of equally-typed method arguments," in *Proceedings of the 2011 International Symposium on Software Testing and Analysis*. ACM, 2011, pp. 232–242.
- [8] T. Savage, M. Revelle, and D. Poshyvanyk, "Flat³: Feature location and textual tracing tool," in *Proceedings of 32nd ACM/IEEE International Conference on Software Engineering (ICSE'10), Formal Research Tool Demonstration*. ACM, May 2010.
- [9] K. Tian, M. Revelle, and D. Poshyvanyk, "Using latent dirichlet allocation for automatic categorization of software," in *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*. IEEE, 2009, pp. 163–166.
- [10] C. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [11] E. M. Voorhees, "Overview of trec 2007," Tech. Rep., 2008.
- [12] E. M. Voorhees and D. K. Hardman, "Overview of the eighth text retrieval conference (trec-8)," in *Trec*, vol. 99, 1999, pp. 1–25.
- [13] B. Dit, M. Revelle, M. Gethers, and D. Poshyvanyk, "Feature location in source code: A taxonomy and survey," *Journal of Software Maintenance and Evolution*, vol. 23, no. 7, 2011.
- [14] R. Krovetz, "Viewing morphology as an inference process," in *Special Interest Group on Information Retrieval*, R. K. et al., Ed., June 1993.
- [15] D. Binkley, D. J. Lawrie, C. Uehlinger, and D. Heinz, "Enabling improved ir-based feature location," *Journal of Systems and Software*, vol. 101, no. 0, pp. 30–42, 2015.
- [16] R. Baeza-Yates, B. Ribeiro-Neto et al., *Modern information retrieval*. ACM press New York, 1999, vol. 463.
- [17] M. Anderberg, *Cluster Analysis for Applications*. Academic Press, 1973.
- [18] S. Haiduc, G. Bavota, A. Marcus, R. Oliveto, A. De Lucia, and T. Menzies, "Automatic query reformulations for text retrieval in software engineering," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.
- [19] A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, "On the naturalness of software," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 837–847.