AUTOMATICALLY IDENTIFYING DEVELOPER GOALS AND SYMPTOMS IN Q&A FORUMS TO HELP FORUM SEARCH AND MINING

by

Zachary R. Senzer

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Honors Bachelor of Science in Computer Science with Distinction

Spring 2017

© 2017 Zachary R. Senzer
All Rights Reserved

AUTOMATICALLY IDENTIFYING DEVELOPER GOALS AND SYMPTOMS IN Q&A FORUMS TO HELP FORUM SEARCH AND MINING

by

Zachary R. Senzer

Signed: _____

Lori Pollock, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: _____

K. Vijay-Shanker, Ph.D.
Committee member from the Department of Computer & Information Sciences

Approved: _____

Sebastian Cioaba, Ph.D.
Committee member from the Board of Senior Thesis Readers

Approved: _____

Michael Arnold, Ph.D.
Director, University Honors Program

ACKNOWLEDGMENTS

I would like to thank Dr. Lori Pollock and Dr. Vijay Shanker for taking a chance on an inexperienced freshman and guiding him along the road of success. Your mentorship throughout my college career has been invaluable and it has been an honor working with both of you.

I would also like to thank Dr. Sebastian Cioaba for his guidance with this research and Samir Gupta, for his time and natural language processing expertise to assist with parsing and pattern matching.

Lastly, I would like to thank my family. Thank you to my mother, Deborah Senzer, for devoting your life to me with your love. Thank you to my father, Richard Senzer, for being a role model of success. Thank you to my brother, Benjamin Senzer, for driving me to work harder and achieve more.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

When software developers need help with their development or maintenance task, they seek information from other developers using question and answer forums. Developers often search the forum by submitting queries, which results in a ranked listing of posts displayed as question title and excerpt from the question description. This thesis presents a technique to automatically extract the questioner's software-related goals and symptoms from the question description. We show how explicit identification of goal and symptom information can help developers determine relevance of search results. Classification of natural language text in forums could also be useful to improve forum mining techniques for other purposes.

Chapter 1

INTRODUCTION

A substantial portion of modern computer science research focuses on the connection between computation and linguistics. One of the more prevalent fields of study concerning this connection is natural language processing (NLP). NLP has the goal of automatically analyzing human (natural) language text. A text source that can reveal powerful knowledge using NLP is question and answer forums. This thesis explores the application of NLP on question and answer forums within software engineering.

1.1 Question and Answer Forums in Software Engineering

Question and answer (Q&A) forums are websites where a community of users can ask questions on specific domains and receive answers, crowd-sourced from the community. Q&A forums are a great way to efficiently obtain information that is tailored to one's specific inquiry. Initially conceived only as a means of providing answers to questions, Q&A forums now focus on knowledge creation that results in enduring value to a large audience of people [2]. A domain in which Q&A forums are incredibly popular is software engineering. As software development teams are more globally distributed and the open source community has grown, developers rely increasingly on Q&A forums for help that they might have previously obtained through in-person conversations. These forums contain vast amounts of valuable collective knowledge from other programmers' experiences.

It is the popular opinion of many software developers that Stack Overflow [9] is the best Q&A forum for the software engineering domain. Stack Overflow boasts a community of 6.5 million users and over 13 million questions. The most popular questions on Stack Overflow are about specific programming languages (e.g., Javascript,

Java, C#) and operating systems (e.g., Android, iOS). Stack Overflow has two main use cases: (1) finding posts with similar issues that you are encountering and (2) responding to questions to which you know the answer. To achieve these two goals, users need to be able to quickly identify what the asker of a question is trying to achieve and what is preventing him/her from achieving it.

Identifying and classifying the different types of structured information (e.g., code, natural language) within developer communications, such as Q&A forums, allows analysis tools to treat them differently. This research goes beyond prior work on identifying and classifying content in Q&A forums by further characterizing the natural language in the forum posts to aid analyses. Specifically, natural language content is analyzed to identify a developer's goals and symptoms. A *goal* is defined as text that indicates what the developer is trying to do or achieve. A *symptom* is defined as text that indicates what is going wrong when the developer is trying to achieve a goal or what is preventing the developer from achieving his/her goal.

Explicitly identifying the goals and symptoms of a given forum post can quickly make it clear to a developer whether a question is relevant or not, which assists in achieving the two use cases of Stack Overflow. If a developer is having a problem with a project, he/she will frequently visit Stack Overflow and compose a search. The two main types of queries one would search for are the task to be accomplished and the difficulties being encountered. Automatically generated goal/symptom classifications can help improve the accuracy of the mining and analyses of Q&A forums for supporting various software engineering tasks. These analyses can use the goal and symptom information to focus their mining on specific content within posts, as appropriate to their analysis.

1.2 Thesis Contributions

This thesis makes the following contributions:

- A technique for the automatic identification and classification of developer goal and symptom from Q&A forums
- An evaluation for goal and symptom identification on Stack Overflow entries

Chapter 2

BACKGROUND

This chapter provides background on software developer Q&A forums and the challenges posed by the targeted research problem of identifying goal and symptom descriptions in Q&A forums.

2.1 Stack Overflow

The standard Stack Overflow post consists of an original question and answers to that question. Answers that are deemed "good" by the community are voted up and rise to the top of the page. The original asker of the question can mark one answer as accepted, indicating that it worked. To categorize posts, Stack Overflow implements a tagging system, such that all questions are tagged with their subject areas (e.g., Javascript, C#). If users want to search for a question, they can specify tags, words that appear in the title, body, or answer, and other miscellaneous features about the status of the post. To ensure that the best answers to the question can be provided, Stack Overflow enables users to add comments to ask for more information or to clarify a question or answer.

A typical question on Stack Overflow can consist of natural language text, interspersed with code segments, code patches, stack traces, and even small code statements or single identifiers within natural language sentences. Thus, there exist many different types of information that users need to take into account when either asking or answering a question. The main type of question content that users of Stack Overflow need to digest is natural language text.

2.2 Successful Question Composition

Stack Overflow provides recommendations on the types of questions users should ask. It suggests to "focus on questions about an actual problem you have faced" and "include details about what you have tried and exactly what you are trying to do" [10]. Further, it notes that "not all questions work well in our format" and to "avoid questions that are primarily opinion-based, or that are likely to generate discussion rather than answers" [10]. To ensure questions meet certain standards, moderators have the ability to close questions.

In an additional attempt to keep post quality high, Stack Overflow provides links to articles on successful question composition. The main advice is to imagine you are trying to answer the question being asked to ensure it is clear what is being asked and easy to read and understand [5]. It is also important to provide context on the language, platform, and operating system, along with any steps already attempted and the results of any research [5].

It is important to emphasize what you are trying to get out of the question. It is crucial to be explicit with what is trying to be accomplished and what is happening during attempts [5]. One article notes that "too many 'questions' are actually just statements: when I do X, something goes wrong" [5]. Further, many posters fail to elucidate what the large aim of the question is. Other problems include the addition of distractions such as greetings and sign-offs (e.g., "Hi everyone", "thanks"), incorrect formatting, and grammatical errors.

Even with Stack Overflow's resources on question composition, users ask many questions that make the comprehension process difficult. In certain cases, this can be attributed to a lack of effort; however, it is important to keep in mind that it is challenging to convey a personal problem on a complex subject to others. This motivates research on how to make Stack Overflow easier to use.

2.3 Software Developer Goals

A software developer's goal is defined as text that indicates what the developer is trying to do or achieve. Some example sentences that contain goals are: "What I'm trying to do is increase the height of each row of my listview", "I want to retrieve the added year", and "My objective is to get the elements of an xml file". For sentences containing goals, part of the sentence will typically contain the goal itself, and a separate portion of the sentence will indicate that the sentence contains a goal. In the example sentences, the explicit goals are: "increase the height of each row of my listview", "retrieve the added year", and "get elements of an xml file". However, it is important to note that the main indicators of goals are words such as "trying", "want", and "objective".

2.4 Software Developer Symptoms

A software developer's symptom is defined as text that indicates what is going wrong when the developer is trying to achieve a goal or preventing the developer from achieving his/her goal. Example sentences that reveal symptoms include "It presses the button, but the file is not downloaded", "Unfortunately, the program gives me a run time error", and "The console.log is saying that a variable is undefined." The explicit symptom text in the example sentences are: "file is not downloaded", "run time error", and "variable is undefined".

2.5 Motivating Example

To further understand software developer goals and symptoms at a high level, consider the sample post in Figure 2.1.

For simplicity, this post is smaller and much less complex than many posts on Stack Overflow. The text in the first sentence, "I hope someone can help me", does not tell the reader anything about the problem at hand and should be considered noise. The next two sentences state that the form is being coded in C# and that there are some problems with a text box. Each of these sentences contains key details

~~I hope someone can help me.~~ **I am doing a Windows Form in C# and I have some problems with a TextBox.** ~~I am new to programming.~~

```
double n, x, i = 15.0, act = 25.0, f, z;
string ac;
int n2, pos, f2;
private void Form1_Load(object sender, EventArgs e)
{
     //This allow to have decimal with the initial value of 25.0
     textBox2.Text = string.Format("{0:0.0}", act);
}
```

I want to enter a decimal value in textBox2. But when I press the return key for delete 25.0 it return this error: An unhandled exception of type 'System.FormatException'.

Figure 2.1: Sample Stack Overflow post with key contents bolded

pertaining to the comprehension of the post. The first sentence explains that the user is working with a Windows Form. Since that is what the user is trying to do, that is part of the goal. In the next sentence, since the textbox problems are preventing the user from achieving the aforementioned goal, that is a symptom. "I am new to programming" is also noise and does not assist with the comprehension of the post. Due to the complexity of deriving meaning by looking at code, only natural language text is considered in this research. Thus, any non-text portions (e.g., code) will be ignored. The next sentence explicitly states what the user wants to do and should appropriately be marked as a goal. This is followed with what is preventing the user from achieving the goal (unhandled exception) and is a symptom.

In summary, "I am doing a Windows Form in C#" is what the user is trying to do and contains a goal. "I have some problems with a TextBox" explains a symptom being encountered. "I want to enter a decimal value" contains another explicit goal. Finally, the returning of an error/exception is a symptom.

2.6 Challenges Imposed by Q&A Forums

There are several major challenges associated with automatically identifying natural language text describing goals and symptoms in Q&A forums. Firstly, unclear intentions, along with formatting, spelling, and grammatical errors, make it incredibly difficult for humans — let alone an automated system — to identify the goals and symptoms present in the post. Secondly, the goal and symptom information is disconnected. There are no designated locations in a post where all of the goal text or symptom text are written. The goal and symptom information is also highly interspersed with other content types (e.g., code), textual context, and textual noise. In an attempt to explain as much information as possible to other users, Stack Overflow questions are dense with sample code, stack traces, and miscellaneous system logs.

Thirdly, the goal and symptom information is not necessarily a whole sentence. Sometimes, the goal/symptom is described in only part of a sentence along with phrases that are not goal/symptom information. Furthermore, part of a sentence might describe the goal while the remainder of the sentence describes the symptoms. Thus, a sentence-level granularity is not appropriate for identifying goal and symptom information. Each of these factors make question understanding increasingly onerous, and automatic identification of goal and symptom information challenging.

2.7 Summary

Stack Overflow provides a helpful platform for developers to ask questions of a knowledgeable community. The most important pieces of information contained in a question are the goals (what the user is trying to do/achieve) and the symptoms (what is going wrong/preventing the user from achieving the goal). Posts on Stack Overflow contain many obstacles involving diverse content types and natural language composition that make question comprehension, and thus goal and symptom identification, a difficult task.

Chapter 3

A STUDY OF STACK OVERFLOW USERS

Stack Overflow contains a search feature to help users discover relevant posts. The results page lists the title, post excerpt, tags, and voting information for each question result. The result excerpt consists of a few lines of text based on keyword matching between the question and the search query. Unfortunately, many search result excerpts contain fragments, code, and non-pertinent information. The structure of the excerpt succeeds in showing that the results match terms contained in the query; however, it often fails to provide the user with enough information to determine whether a question is relevant to a user's needs and contains noise.

We hypothesize that an excerpt consisting of the asker's goals and symptoms would make it quicker and more efficient to understand a post and determine whether it is relevant to one's specific needs. To examine this hypothesis further, we considered typical Stack Overflow search queries. For each query, we presented the search results in two different formats:

1. original excerpts (Figure 3.1)
2. excerpts replaced with goal and symptom information for each search result (Figure 3.2).

To determine which presentation format of Stack Overflow search results is more useful, we presented both formats to human judges. Ten human annotators examined search result pages in both formats for three queries (30 responses). For each query, the human annotators had to decide:

1. Which search result format was *more helpful* in deciding if a search result for the query is relevant?

2. Which search result format was *faster* in helping them decide if a search result for the query is relevant?

For each of these decisions, judges were asked to assign a score from 1 to 5, where a score of 5 indicated that the format with goal and symptom information was significantly more helpful/faster, a score of 1 indicated that the format with original excerpts was significantly more helpful/faster, and a score of 3 indicated the two formats were equally helpful/fast. The average score over all judges for all three queries for the first question, which is concerned with the helpfulness of the format in determining relevance, was 4.30. The average score for the question asking which search result format was faster for determining relevance was 4.43. For both questions, over 93% of responses indicated that the search result presentation format with the goal and symptom information was the superior option (i.e., a score of 4 or 5). These results suggest that our hypothesis is correct, and motivate the automatic identification of goal and symptom information from Q&A forums to enable such a result display to help search results analysis.

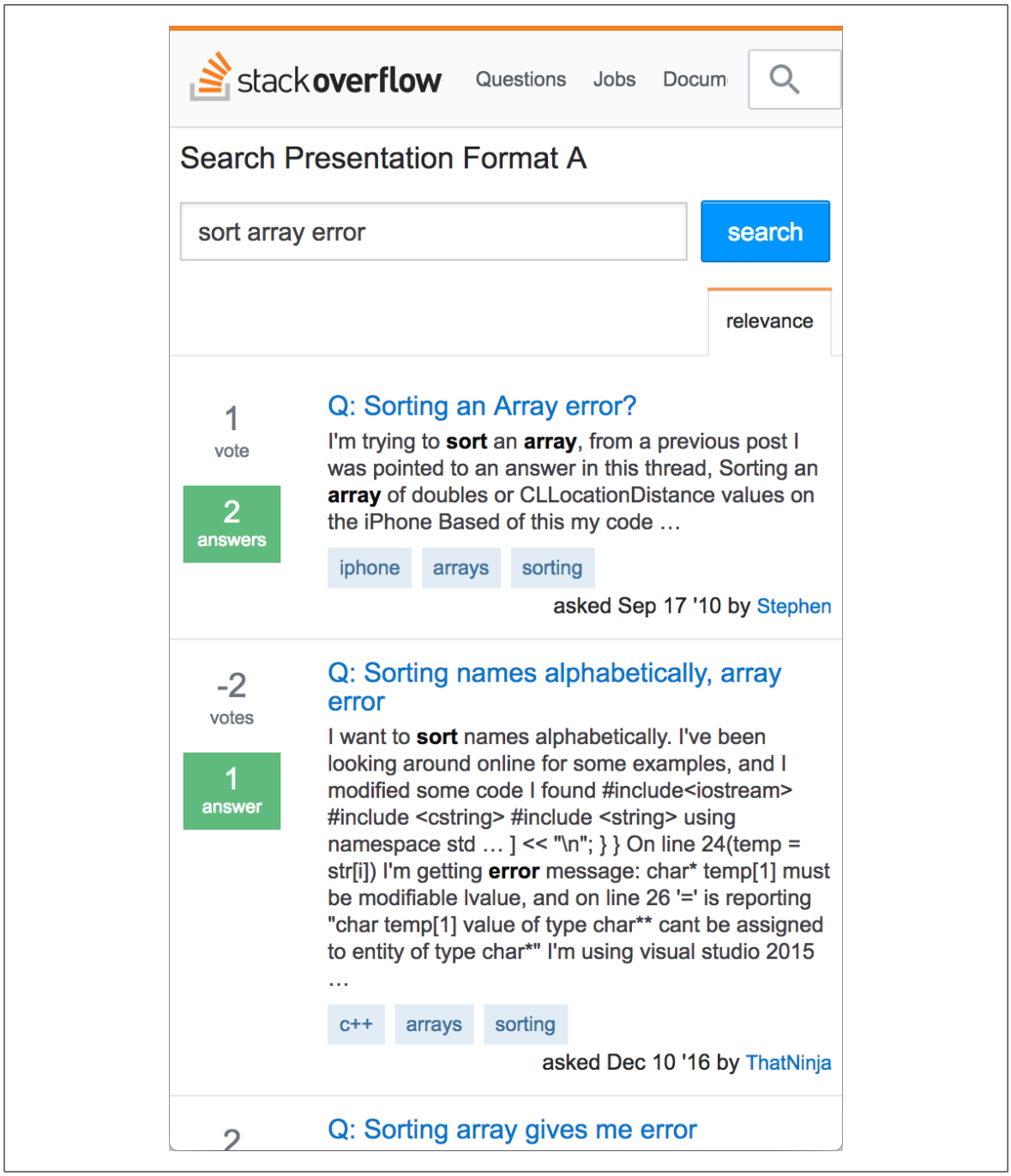

Figure 3.1: Original search result excerpts

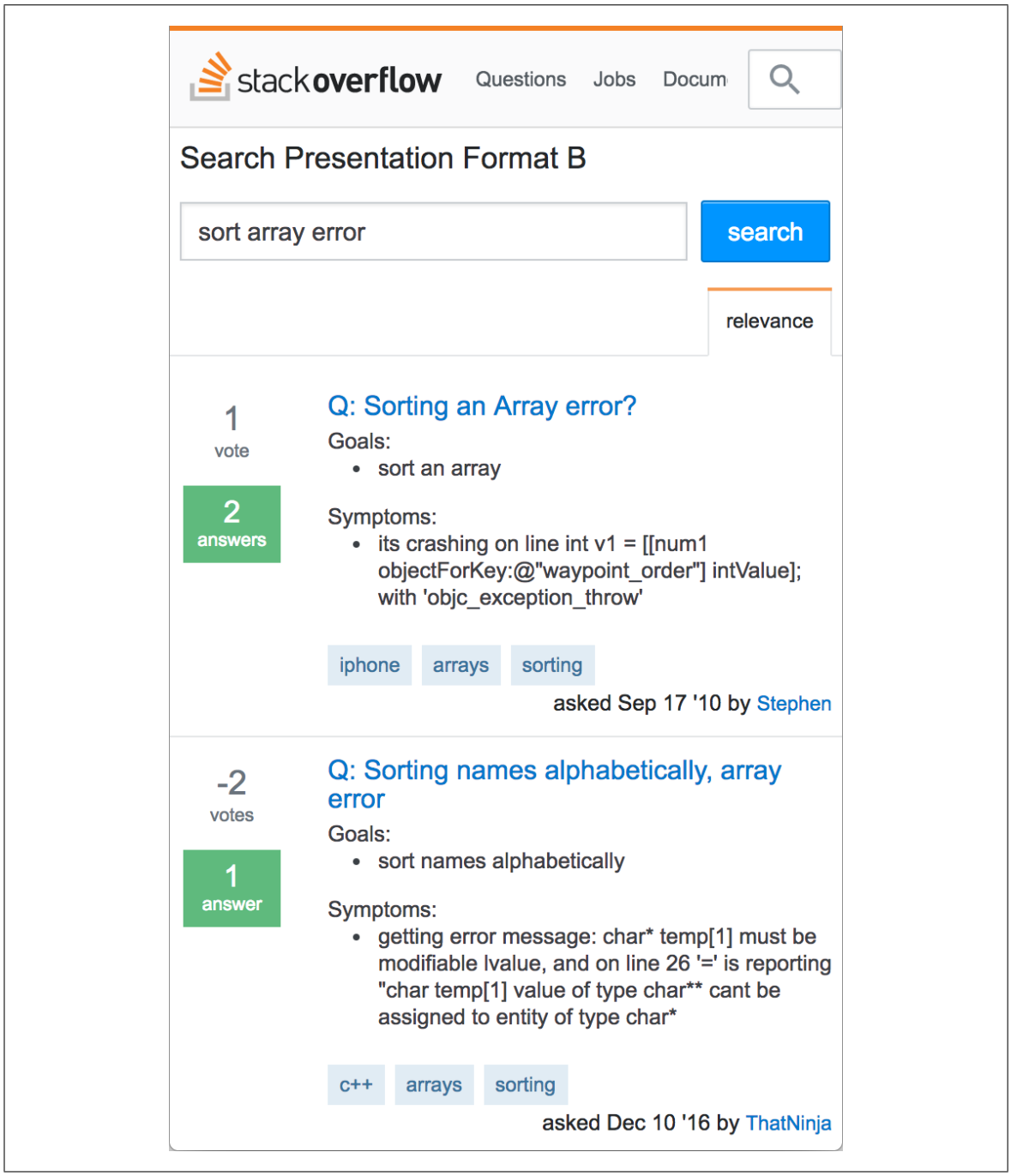

Figure 3.2: Search results with goal and symptom information

Chapter 4

A LEXICOSYNTACTIC APPROACH TO AUTOMATIC IDENTIFICATION OF GOAL AND SYMPTOM INFORMATION

The process of automatically identifying goals and symptoms in Q&A forum posts consists of two components:

1. the construction of patterns/rules to determine the classification of a given text snippet
2. the application of these patterns in a system that automatically identifies goals and symptoms in natural language text.

4.1 Key Insights

Posts on Stack Overflow can be viewed as an interaction taking place between a developer and a system. Goals explain what the developer is trying to get the system to do or achieve. Symptoms explain what is going wrong with the system that is preventing the developer from achieving his/her goal. This insight led to increased focus on the subjects of the natural language text and which actions/verbs are attached to the subject when we analyzed natural language text of forum posts for patterns of expressing developer goals and symptoms. We refer to our approach as the *lexicosyntactic approach* for the remainder of the thesis, since it places emphasis on both the lexical and syntactical structure of the text. There exist certain words that developers use to explain goals and symptoms. However, the presence of these words is insufficient. A system simply built around the presence of words is too general to ensure that goals and symptoms are being correctly identified. It is important to examine the syntactical structure of the text that contains key diction. Specifically, these words need to be

associated with specific subjects. This is especially important for goals, as a goal is what the *developer* is trying to do/achieve. Thus, the subject of the text needs to be the developer. The syntactic role that specific words play in text is meaningful in understanding natural language.

4.2 Pattern Identification

To determine the patterns that exist for describing developer goals and symptoms, we evaluated 200 Stack Overflow posts, manually annotated the goals and symptoms text, and generalized the annotations to consider the text's subject, verbs, and other key diction. This analysis gave rise to three patterns (one for goals and two for symptoms).

Since a goal is what the developer is trying to do/achieve, we expect two components in the text that expresses the goal:

1. a verb that expresses the developer's desire, and
2. that it is the developer who desires it.

The first component can be captured by a list of words that we call *desire verbs*. These components note that the text contains a goal, but the goal itself will appear after the desire verb as its argument. The second component can be captured by the subject of the desire verb being "I".

Thus, the goal pattern consists of a first person pronoun subject and a desire verb. The first person pronoun (e.g., I, we) subject places the focus on the asker of the Stack Overflow question. The desire verb is connected to the subject and expresses what the asker is trying to do. Our initial list of desire verbs included "want", "wish", "like", "need", and "try". We also consulted a thesaurus to expand this list, and all inflections of these verbs were taken into account. An example of this patten is:

"I want to retrieve the added year"

where “I” + “want” matches the pattern, and the goal itself is “retrieve the added year”.

Symptom patterns are more complex to identify since there are many ways that people describe what is going wrong on Q&A forums. For symptoms, it can not be expected that the subject will always be first person. Symptoms can reveal themselves on either side of the developer/system interaction.

We call the first symptom pattern *developer-identified symptoms* because the text mentions the developer noticing/identifying a symptom. Therefore, we expect the subject to be a first person pronoun and the text to have a verb corresponding to observation (we call these *system reference verbs*). A system reference verb illustrates an action taken by the system and directed to the developer. Our initial list of system reference verbs includes words such as “get” and “have” (e.g., “I get an error”).

With symptoms, the developer must be noticing something wrong with the system’s behavior. This can be stated using a range of diction, hence we look for words that indicate undesired behavior. We call this list a *negative sentiment list*, intended to capture both general and programming-related negative sentiment, including “issue” as well as “error”, “exception”, “null”, and “undefined”. We also consulted sentiment analysis word lists to expand our set. We extract the entire clause as the developer-identified symptom. An example of the the developer-identified symptom pattern is:

“I have an issue with the structure”

Not all symptoms are expressed as the developer noticing an issue. In many cases, the symptom can simply be stated as the system’s undesired response. In these cases, we expect the subject to be indicative of the “system”, therefore, we include words such as “console.log” and “code” as a proxy for “system”. As before, we expect to see negative sentiment. Thus, our second symptom pattern, which we call *system-identified symptoms*, consists of a system subject and negative sentiment. An example of this pattern is:

Table 4.1: Example goals and symptoms (patterns bolded and goals/symptoms italicized)

Goal	**I'm trying** to *"cacheify" my angular service factory.*
	I need to *have a multidimensional array in a shared memory between two processes.*
	I'm trying to *use the str.index method for a given DNA strand, but to no avail.*
	I want to *have a view that exposes employee data across years.*
	I am having trouble passing an array to my sorting classes because **i need** to *sort the same array with two different algorithms.*
Developer-Identified Symptom	However it seems that **I** am ***making mistake*** *in getting input to program in atoi() function.*
	I *got* *this* ***error****, pointing at the line of the connectionstring.*
	I'm *having* *some* ***issues*** *getting the stl to be properly referenced.*
	I *get* *an* ***error*** *on the line of code trying to assign tag 102 to *tmp.*
	However, **I *don't see*** *an easy way to link to such a file.*
System-Identified Symptom	My ***console*** *keeps giving me a message that says "Uncaught* ***TypeError****:* ***Cannot*** *read property 'replace' of* ***undefined*** *".*
	It just start *eating memory (will goes up to 1GB on simulator) and cause a* ***crash****.*
	However, when I put that date format into a Java program and use a prepared statement **it** *throws a very strange and meaningless* ***error*** *of "****Missing*** *IN OUT parameter at index:: 6" which is* ***strange*** *because there are only 5 parameters.*
	I don't know why any entered ***value*** *returns* ***false*** and the ***programs stops***.
	I am trying to place two image buttons on my image background in a certain position, but my ***buttons*** *are* ***not*** *appearing.*

Table 4.2: Frequency counts for desire verbs, goal verbs, and negative sentiment (for 50,045 Stack Overflow posts)

Rank	Desire Verb	Freq.	Rank	Goal Verb	Freq.	Rank	Neg. Sent.	Freq.
1.	want	11499	1.	use	2828	1.	not	26350
2.	trying	8129	2.	get	1837	2.	error	7444
3.	need	5713	3.	create	1760	3.	just	5131
4.	like	4052	4.	make	1452	4.	change	5097
5.	tried	3775	5.	add	1355	5.	only	3920
6.	try	2345	6.	know	1248	6.	no	3218
7.	wanted	533	7.	have	928	7.	out	2693
8.	attempting	296	8.	run	839	8.	problem	2663
9.	wish	165	9.	change	701	9.	issue	1864
10.	prefer	138	10.	write	700	10.	every	1754

"The console.log is saying that a variable is undefined."

where the presence of "console.log" and "undefined" allows us to fit the pattern.

Table 4.1 shows additional examples of goals and symptoms for each pattern. Based on our manual analysis, we compiled lists of words for desire verbs, system reference verbs, system subjects, and negative sentiment. As shown in Figure 4.1, these lists and patterns are incorporated into the approach as input to pattern matching. Table 4.2 presents some frequencies of the words that people use to express goals and symptoms on Stack Overflow.

4.3 Pattern Matching and System Output

Our automated system accepts a Stack Overflow post as input and generates the Stack Overflow post with a goal and symptom header. The different steps of the automated system, as depicted in Figure 4.1, include:

1. Preprocess the Stack Overflow post for natural language text analysis
2. Tokenize and split the natural language text into sentences
3. Identify syntactic dependencies for each sentence using a dependency parser
4. Apply the patterns to extract goals and symptoms by using the syntactic dependency information

Given the original content for a Stack Overflow post, the preprocessing step removes all code blocks and replaces any in-line code with a noun-phrase placeholder, "IN-LINE CODE". This ensures that only the natural language text is analyzed, and any embedded code snippets do not misguide the parse during the goal/symptom identification.

Next, we tokenize the natural language text into a sequence of tokens and split the text into sentences using the Stanford CoreNLP toolkit [7]. We then apply the Stanford Constituent Parser [6]. The parse tree is converted into syntactic dependencies using the syntactic dependencies converter [3].

Stanford dependencies provide a representation of grammatical relations between words in a sentence. They are often represented as a directed graph where the words in the phrase or sentence are nodes, and the grammatical relations are edge labels. The dependencies are expressed as triplets: name of the relation, governor (or head), and dependent. The dependencies use the Penn Treebank part-of-speech tags and phrasal labels.

As shown in Figure 4.2, one such dependency triplet is *nsubj* (I, retrieve), where the relation is *nsubj* (nominal subject), the governor of the relation is "retrieve", and the dependent is "I", respectively. The edge labeled *nsubj* leads from "retrieve" to "I". The use of syntactic dependencies allows us to examine sentences at a level that abstracts away from many textual variations. In addition, we use the "CCprocessed" dependency representation [3], which allows for an appropriate treatment of sentences that involve conjunctions.

To implement the patterns to extract goals and symptoms, we translated the patterns into constraints on the dependency representation of the sentences. The goal pattern requires the presence of a desire verb. We identify verbs by using the part of speech information and see if any of the words is a "desire verb". Next, the goal pattern requires that its subject is a first person pronoun. This gets translated into a constraint that there be an *nsubj* dependency in which the governor is the desire verb ("want" in Figure 4.2) and in which the dependent is a first person pronoun ("I"). All

desire verbs take an infinitival clause (subordinate clause whose verb is in the infinitive form such as "to retrieve") as their arguments. As this corresponds to an open clausal complement, *xcomp*, dependency, we need to look for an *xcomp* relation with governor as the same desire verb.

We will retrieve the entire verb phrase that is dependent on the *xcomp* relation. To obtain the full verb phrase (which will be the actual goal), we start with the dependent verb ("retrieve") and need to consider all relations starting from this verb. This verb phrase will be retrieved as the goal ("to retrieve the added year").

To recognize developer-identified symptoms, we first look for system reference verbs ("have" in Figure 4.3). Our pattern for developer-identified symptoms requires that the subject of the system reference verb be a first person pronoun ("I"). Thus, similar to the goal pattern, we look for an *nsubj* syntactic dependency edge from the system reference verb ("have") and the first person pronoun ("I"). Then, we extract the complete phrases, which are dependents of the system reference verb ("have an issue with the structure") and look for negative sentiment words within these phrases. In the example illustrated in Figure 4.3, the presence of negative sentiment ("issue"), allows us to conclude that this sentence describes a developer-identified symptom.

For system-identified symptoms, we need to consider the subjects of the verb and consider whether they refer to the system. When we find the presence of such a system subject, similar to the developer-identified symptoms pattern, we look for negative sentiment in the clause headed by this verb. Figure 4.4 depicts the syntactic dependency of such a case for system-identified symptoms, where the system subject is "console.log", which is the subject of the verb "saying", indicated by the *nsubj* edge between them. The clause headed by "saying", "saying that a variable is undefined", contains the negative sentiment "undefined" and is extracted as a symptom.

The patterns for goal and symptom identification have been translated as patterns on the dependency representation of sentences. We use Semgrex, which is a part of the Stanford NLP Toolkit, to specify the translated patterns as regular expressions based on lemmas, part-of-speech tags, and dependency labels, which will automatically

match with the sentence dependency parse structure.

4.4 Summary

Our approach leverages the words and syntax present in a post to derive patterns for goal and symptom identification. A Stack Overflow post is first preprocessed to remove instances of code. The post with only natural language text then undergoes sentence splitting, tokenization, and dependency parsing. Finally, the goal and symptom patterns allow us to perform pattern matching to obtain the desired output of a Stack Overflow post with a goal and symptom header.

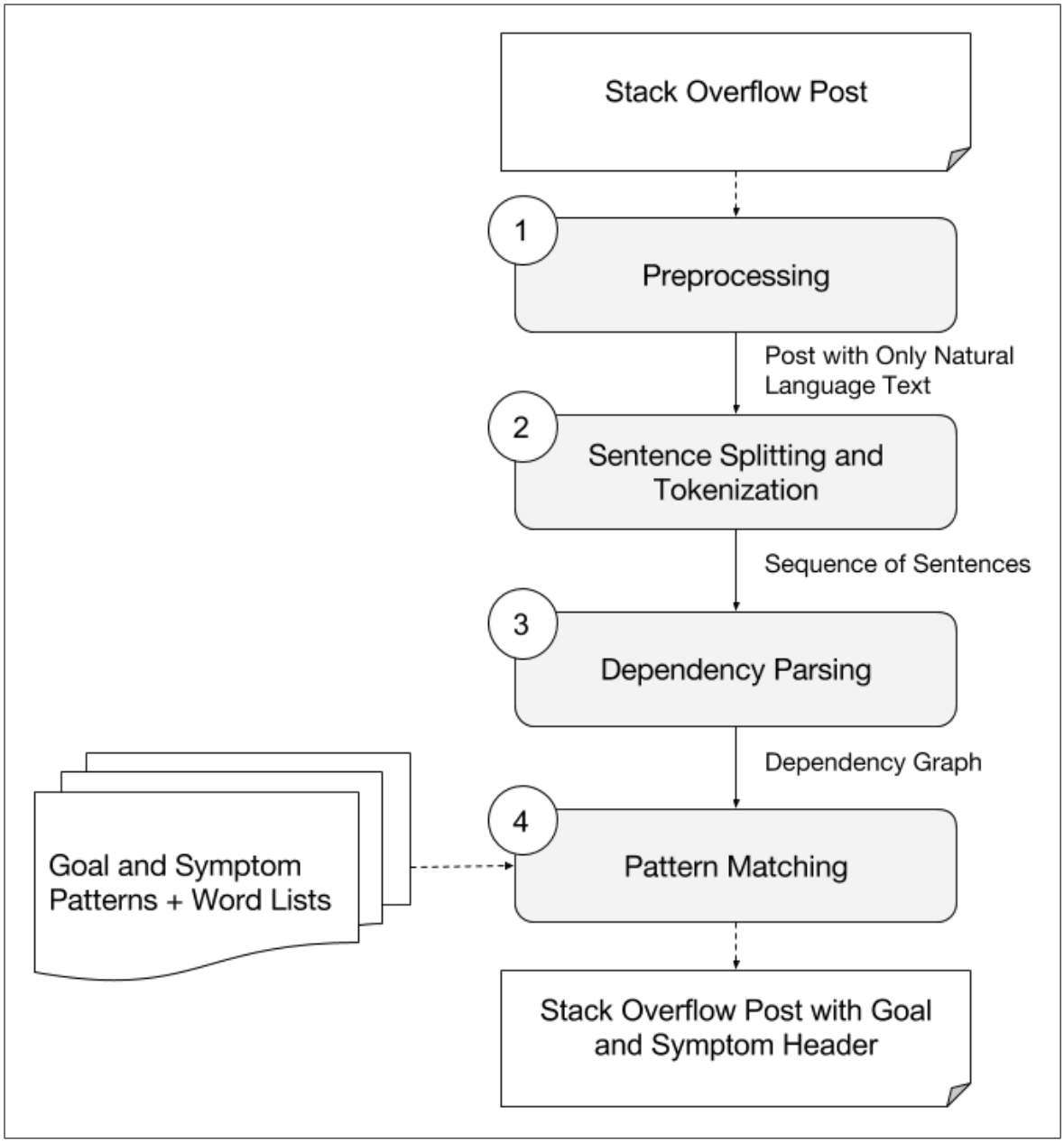

Figure 4.1: Lexicosyntactic system overview

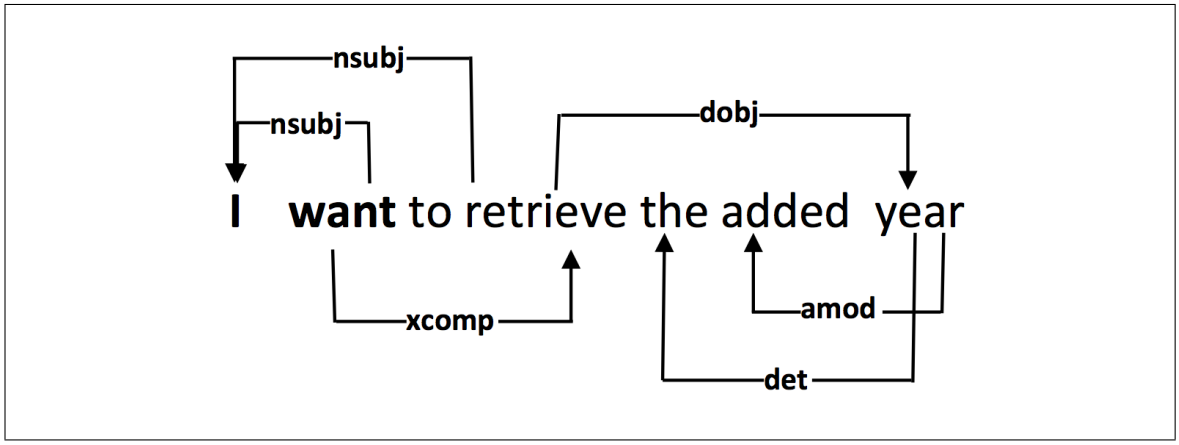

Figure 4.2: Example syntactic dependency of a developer goal

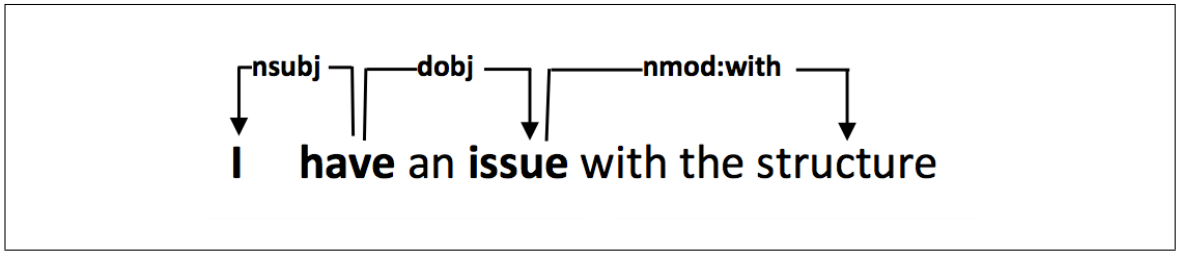

Figure 4.3: Example developer-identified symptom syntactic dependency

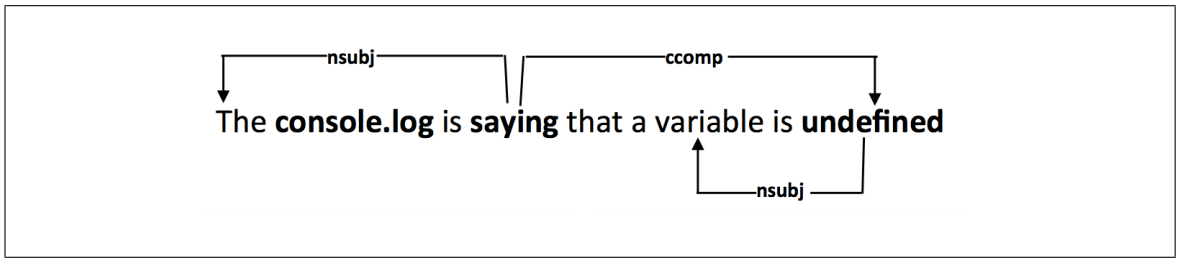

Figure 4.4: Example system-identified symptom syntactic dependency

Chapter 5

EVALUATION

We designed our evaluation study to answer one primary question:

How well does our goal/symptom identification technique classify text snippets in Q&A forums?

5.1 Subjects, Variables, and Measures

The subjects in our goal/symptom identification are the natural language question text representing 75 randomly selected Stack Overflow questions from April 2014. Each subject question contains at least one code segment, as the majority of symptoms on Stack Overflow are stated in regard to code that a developer is referencing. Additionally, questions were filtered to ensure that the main goal and symptom information did not contain anaphoric expressions such as "this", "that", "they", "these", and "those".

The independent variable is the technique for goal/symptom identification of natural language text. The dependent variable is the effectiveness of the approach in terms of the precision, recall, and F-measure in identifying the goal and symptom content in question text from Q&A forum posts.

Table 5.1: Results for lexicosyntactic identification of goal and symptom information

Class	**Precision**	**Recall**	**F-measure**
Goal	76.85	72.81	74.77
Overall Symptom	64.83	69.12	66.90
Developer-Identified Symptom	57.89	N/A	N/A
System-Identified Symptom	78.00	N/A	N/A

```
Introduction:

Typically, a Stack Overflow post includes information
about a developer's goal(s) and his/her encountered
symptom(s).

In the context of a Q&A forum, we define a goal as:
"text that indicates what the developer is trying to do
or achieve."

We define a symptom/problem as: "text that indicates
what is going wrong when the developer is trying to
achieve their goal or preventing the developer from
achieving his/her goal."

-------------------------------------------------------

Study Instructions:

This study seeks to identify goals and symptoms in
Stack Overflow posts.

Attached, you will find a ZIP file containing 75 PDF
files. Each PDF file is a Stack Overflow post with the
file name, [StackOverflowID].pdf.

For each PDF file, highlight the goal(s) in green and
highlight the symptom(s) in pink. Only consider the
English text in the question portion of the post (ignore
titles, answers, and code). If you have questions
about whether a section should be considered, please
ask me.

Note: You should even highlight restated/repeated
goals and symptoms. Cover all question text and
CONSIDER WHETHER ALL TEXT CONTAINS
EITHER A GOAL OR A SYMPTOM.

Note: Some posts might not contain a goal or a
symptom. Some posts might contain multiple goals
and multiple symptoms. Highlight however many you
feel apply.

Please save each PDF with the colored highlights and
compress/ZIP a folder containing the 75 highlighted
PDF files.

Thank you for your participation!
```

Figure 5.1: Instructions provided to human judges

5.2 Procedure

Our lexicosyntactic identification system was run on the 75 Stack Overflow posts. As a gold set, the system's classifications of goal and symptom were compared with that of our human annotators. Three human annotators were given the 75 Stack

Overflow posts and asked to highlight which natural language text were goals and which text were symptoms. They were given the opportunity to look at the complete content of each Stack Overflow post that they were analyzing. Each of the 75 Stack Overflow posts was evaluated by the three (non-author) annotators. The instructions provided to our human judges are shown in Figure 5.1. If 2/3+ of the human judges had a text segment highlighted as a goal or symptom, it was added to the gold set. The overlap between human judge highlights was determined by checking whether the head goal verb (for goals) and head symptom verb and negative sentiment (for symptoms) were shared. The same overlap methodology was used to determine overlap between system and gold set highlights.

5.3 Results and Qualitative Analysis

Table 5.1 presents the precision, recall, and F-measure for goal and symptom identification for the lexicosyntactic approach.

The lexicosyntactic approach correctly identifies at least one goal in over 81% of posts and at least one symptom in over 80% of posts. All measures for our data indicate that our lexicosyntactic system is a bit more effective at identifying goals than it is at identifying symptoms. This is essentially due to a lower precision for the developer-identified symptoms than the system-identified symptoms. The system-identified symptom pattern had precision of 78%, while the developer-identified symptom pattern had precision of 57.9%.

To obtain more insight into the challenges of our lexicosyntactic system, we analyzed the differences between our system and the gold set developed with human judges. Table 5.2 shows specific examples of false positives and false negatives for goals and symptoms. This qualitative analysis leads to several future improvements to the technique.

For goal precision, the largest issue involved the tense of desire verbs. Since our approach uses lemmas, all tenses of the desire verb were considered. Many times, a question would contain “I tried...”. This past tense led human judges to consider

Table 5.2: Examples of cases where the lexicosyntactic approach falsely identifies and misses goals and symptoms

Class	Text	Explanation
Goals: False Pos.	Everyone that use my application will be able to publish events to my facebook page, so **I don't want to give admin or contributor to everyone**.	Desire verb negation
	I tried to do some countif operations in Excel, but it was prohibitively slow.	Past tense desire verb
	You may notice that **I needed to escape the quotation marks to keep them in the string**.	Past tense desire verb
Goals: False Neg.	I'm **getting an image from a base64 string, stored in a datatable**.	No desire verb
	But the goal is that the **javascript is triggered when the user clicks the button and an item is printed to the screen**.	No first person pronoun
	Any ideas how to **get it to stick the right content into the right containers**?	No first person pronoun
Symp.: False Pos.	When the controller handles the POST, **I can access the posted form field with no problem**.	Negative sentiment negation
	I don't work in linux much so please try to be explicit with any answers.	Negative sentiment incorrectly applied
	Now as **I had no errors or exceptions thrown** I believe I have succeed in writing an output file.	Negative sentiment negation
Symp.: False Neg.	The lattice graphs display just fine, the **image.png doesn't display at all**.	System missing subject
	Why is **my current use giving a gray box and in no way copying or drawing the pixels in the image**?	System missing subject
	EDIT: Nneoneo's answer works if "exisiting_folder" exists, but **does not behave properly if "existing_folder" and "existing_folder_1" exist**.	System missing subject

the statement as more of an already attempted troubleshooting step, as opposed to a currently open goal. Having the system filter according to tense, specifically removing past-tense desire verbs, would increase goal precision. Additionally, our system was unable to detect the negation of desire verbs. Thus, "I don't want..." was marked by our system as a goal, but not by our human judges. A future system should notice that the negation of certain keywords should invalidate the goal identification. The main source of recall challenges for goals was the lack of a pattern to address "how to" goals (e.g., "how to add the numbers in my array"). These goals appeared relatively frequently, but did not contain a first person pronoun or a desire verb, and thus, were not marked as goals by our system. The inclusion of a pattern to incorporate "how to" goals would lead to a drastic increase in goal recall.

The main source of precision challenges for symptoms was due to the developer-identified symptom pattern. The identification of developer-identified symptoms mainly struggled due to the incorrect detection of negative sentiment. There were too many times when a negative sentiment word was not truly describing a symptom, but since the text contained the word, it was marked as a symptom by our system. Future work should further analyze the occurrences of negative sentiment in our symptom pattern matches and examine the specific contexts in which it is not linked to a symptom. Thus, these instances could be generalized and filtered by our system. Additionally, negative sentiment could be weighted to increase the confidence that there is enough negative sentiment present in the text to identify it as a symptom. The overarching issue with symptom recall was the lack of certain nouns from our system subject list. Since some system subjects were missing, the negative sentiment attached to the subject was not detected and we were unable to extract the symptom. To increase symptom recall, a list of the frequencies of subjects across a set of Stack Overflow posts should be analyzed. From there, if we notice that a subject appears frequently in the Stack Overflow domain and is indicative of the system, it can be added to our system subject list.

5.4 Threats to Validity

Our techniques pulled from Stack Overflow questions for the development sets. The results may not transfer to other Q&A forums; we chose Stack Overflow as it is the most used and we believe that it is a good representation of most software developer forums. A study that uses larger development sets might yield different results. As is the case with any study that uses human judges to obtain the ground truth, there might exist cases where the humans may not have accurately answered their portion of the study. To limit this threat, we ensured that our judges had considerable programming experience and familiarity with Q&A forums. We also ensured each forum post was judged by at least three judges, leading us to take the majority opinion. It is also possible that scaling to more forum posts in our evaluation study would yield different results, but we needed to make the human judgement work reasonable to recruit judges. We plan to expand the evaluation studies in the near future with more participants.

Chapter 6

RELATED WORK

To our knowledge, our work is the first that strives to automatically distinguish goal and symptom information in Q&A forums. However, there has been related research on information categorization, recommendation, and mining within forums and requirement documents. The goal and symptom information that we are identifying, along with the granularity with which we analyze the question content on Stack Overflow differentiates our work from the others in the field.

Allamanis and Sutton [1] categorized Stack Overflow questions based on programming concepts and type of information being sought. Concepts included categories such as "applets" and "games". It was also discovered that certain topics, such as memory management and compatibility issues, do not typically involve the use of code snippets. Question types represented "the kind of information requested in a way that is orthogonal to any particular technology." A major discovery was that question type distributions did not vary among programming languages. The insights from their study were used to perform analyses such as, "what types of questions are most commonly asked about the Date object in Java?" The words and phrases present in the question were used to categorize a given Stack Overflow question.

de Souza et al. [4] created a recommendation strategy that leverages the information in Stack Overflow to "suggest question/answer pairs that may be useful to the programming task that a developer needs to solve." The recommendations were based upon both textual similarity and post score. They classified questions from Stack Overflow into five categories: how-to-do-it, conceptual, seeking-something,

debug-corrective, and miscellaneous. However, only questions classified in the how-to-do-it category were used for the recommendation strategy. Recommendations considered both relevance and reproducibility criteria.

Also exploring Stack Overflow recommendation, Ponzanelli et al. [11] developed Prompter, a "plug in for the Eclipse IDE, which automatically searches and identifies relevant Stack Overflow discussions, evaluates their relevance given the code context in the IDE, and notifies the developer if and only if a user-defined threshold is surpassed." Since developers have to exit their development environment to find information online, the information retrieval process can be disruptive. Prompter was able to retrieve and recommend relevant Stack Overflow discussions within the development environment. Relevance was determined by evaluating code and textual similarity.

Wong et al. [12] built AutoComment, which automatically generates code comments by analyzing Q&A sites. They used Stack Overflow to develop a set of code-description matches by taking a code snippet and mapping it to the title and paragraph preceding the code. This text was identified as a candidate description. Natural language processing techniques were used to filter the descriptions and code cloning was used to match new code to the existing mapping database.

Nguyen et al. [8] developed a "rule-based approach to automatically extract goal and use case models from natural language requirements documents." Their approach could "automatically categorize goals and ensure that they are properly specified." Their objective was to ensure that requirements were being properly specified. The format and diction of requirements specification documents differs greatly from that of Q&A forums. Requirements documents concern multiple levels including business, product, and service. Conversely, forums are much more targeted to a specific issue that a developer is trying to solve.

Lastly, Zhang and Hou [13] explored the automatic extraction of "problematic API design features from forum threads." They leveraged the insight that API problems "tend to be described in negative sentences using negative sentiment words and

phrases." They utilized sentiment analysis and natural language parse trees to identify negative sentences. From there, these negative sentences and their neighboring sentences were used to identify desired natural language patterns. This work focuses on more general developer forums and not question and answer forums, which differ in structure.

Chapter 7

CONCLUSIONS AND FUTURE WORK

This thesis investigates whether the natural language text of software developer Q&A forum posts can be further characterized towards improving mining analyses and search. We created a lexicosyntactic system for automatic identification of developer goals and symptoms from Q&A forum posts. Our goal/symptom identification technique shows promise by being able to classify natural language text with goal precision of 76.85% and symptom precision of 64.83%. This system can serve as a base system for researchers and tool developers who want to utilize a standard XML-based representation of forum posts, which could be expanded to other kinds of developer communications. In presenting formats of Stack Overflow search results in original excerpt form and in a format with goal and symptom headers, 93% of the responses indicated that the search results presentation format with goal and symptom headers was the superior option for being helpful in deciding if a search result is relevant. This provides strong motivation for this work.

In the future, we would like to extend our goal and symptom identification system in three ways: (1) adding new patterns to handle cases such as "how to" questions, (2) having the lexicosyntactic system consider tense and negation, and (3) analyzing the use of negative sentiment and system subjects across Stack Overflow to assist with pattern generalization. These extra measures would help our system discern for the vast majority of cases whether natural language text should be classified in a certain way.

BIBLIOGRAPHY

[1] Miltiadis Allamanis and Charles Sutton. Why, when, and what: Analyzing stack overflow questions by topic, type, and code. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 53–56, May 2013.

[2] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Discovering value from community activity on focused question answering sites: A case study of stack overflow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 850–858, New York, NY, USA, 2012. ACM.

[3] Marie-Catherine de Marneffe and Christopher D. Manning. Stanford typed dependencies manual. Sep 2008.

[4] Lucas B. L. de Souza, Eduardo C. Campos, and Marcelo de A. Maia. Ranking crowd knowledge to assist software development. In *Proceedings of the 22Nd International Conference on Program Comprehension*, ICPC 2014, pages 72–82, New York, NY, USA, 2014. ACM.

[5] Jonskeet. Writing the perfect question. https://codeblog.jonskeet.uk/2010/08/29/writing-the-perfect-question/, 2016.

[6] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[7] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Jun 2014.

[8] Tuong Huan Nguyen, John Grundy, and Mohamed Almorsy. Rule-based extraction of goal-use case models from text. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 591–601, New York, NY, USA, 2015. ACM.

[9] Stack Overflow. Stack Overflow. http://stackoverflow.com/.

[10] Stack Overflow. Welcome to Stack Overflow–Tour. https://stackoverflow.com/tour.

[11] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. Prompter: A self-confident recommender system. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 577–580, Sept 2014.

[12] Edmund Wong, Jinqiu Yang, and Lin Tan. Autocomment: Mining question and answer sites for automatic comment generation. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 562–567, Nov 2013.

[13] Yingying Zhang and Daqing Hou. Extracting problematic api features from forum discussions. In *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*, pages 142–151, May 2013.

Appendix A

LEXICOSYNTACTIC APPROACH WORD LISTS

Desire Verbs

- aim
- anticipate
- aspire
- assume
- attempt
- believe
- can
- demand
- desire
- expect
- expectation
- goal
- hope
- imagine
- intend
- intent
- intention
- interested
- like
- must
- need
- objective
- ought
- picture
- plan
- prefer
- presume
- purpose
- require
- seek
- should
- strive
- suppose
- think
- try
- want
- wish

Negative Sentiment

- abnormal
- abnormality
- accident
- always
- anomaly
- aren't
- arent
- atypical
- bad
- break
- bug
- can't

- cant
- change
- complication
- conflict
- couldn't
- couldnt
- crash
- defect
- defective
- deficient
- defunct
- delay
- deviate
- diappear
- didn't
- didnt
- difficulty
- dilemma
- disable
- discontinued
- doesn't
- doesnt
- don't
- dont
- duplicate
- empty
- error
- every
- exception
- exhaust
- expose
- fade
- fail
- fault
- faulty
- flaw
- forget
- freeze
- glitch
- hadn't
- hadnt
- hasn't
- hasnt
- haven't
- havent
- illegal
- imperfect
- inaccurate
- inadequate
- inadvertently
- incapable
- incompatible
- incomplete
- incorrect
- incorrectly
- ineffective
- inefficient
- infinite
- insecure
- insists
- instead
- interrupt
- invalid
- irregular
- irregularity
- isn't
- isnt
- issue
- just
- lag
- leak
- long
- lose
- lot
- malfunction
- mightn't
- mightnt
- miscalculate
- miscalculation
- misprint

- mistake
- mustn't
- mustnt
- negative
- never
- nil
- no
- none
- not
- null
- obsolete
- obstacle
- oddity
- only
- opposite
- out
- oversight
- poor
- predicament
- problem
- reject
- setback
- shouldn't
- shouldnt
- skip
- slow
- stall
- stop
- strange
- stuck
- symptom
- too
- trouble
- unable
- unacceptable
- uncommon
- undefined
- unintended
- unintentionally
- unprotected
- unreachable
- unresponsive
- unsatisfactory
- unsecure
- unstable
- untypical
- unusable
- unusual
- vanish
- void
- warning
- wasn't
- wasnt
- weren't
- werent
- won't
- wont
- worry
- wouldn't
- wouldnt
- wrong
- wrongly
- zero

System Subjects

- algorithm
- application
- array
- boolean
- button
- char
- class
- client
- code

- column
- compiler
- console
- console.log
- data
- database
- environment
- float
- function
- instance
- int
- interface
- it
- item
- library
- loop
- method
- object
- output
- package
- page
- platform
- process
- program
- result
- row
- script
- server
- set
- string
- structure
- system
- table
- thread
- type
- value
- variable
- view