Efficiently Annotating Source Code Identifiers Using a Scalable Part of Speech Tagger

Gavin Burris
gb9951@rit.edu

Follow this and additional works at: https://repository.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.
Efficiently Annotating Source Code Identifiers Using a Scalable Part of Speech Tagger

by

Gavin Burris

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in Computing and Information Sciences

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY

May 10, 2023
Committee Approval:

Dr. J Scott Hawker  
SE Graduate Program Director  

Dr. Mohamed Wiem Mkaouer  
Assistant Professor  

Dr. Christian D. Newman  
Assistant Professor  

Date
Abstract

This thesis details the process in which a part-of-speech tagger is developed in order to determine grammar patterns in source code identifiers. These grammar patterns are used to aid in the proper naming of identifiers in order to improve reader comprehension. This tagger is a continuation of an effort of a previous Ensemble Tagger [62], but with a focus on increasing the tagging rate while maintaining the accuracy, in order to make the tagger scalable. The Scalable Tagger will be trained on open source data sets, with a machine learning model and training features that are chosen to best suit the needs for accuracy and tagging rate. The results of the experiment will be contrasted with the results of the Ensemble Tagger to determine the Scalable Tagger's efficacy.
I would like to dedicate this thesis to my family for their encouragement and support throughout my academic career.
Contents

1 Introduction ................................................. 1

2 Research Objective ...................................... 3
  2.1 Motivation and Contribution ......................... 3
  2.2 Research Questions .................................. 4

3 Related Work .............................................. 5
  3.1 Part-of-Speech Taggers ................................. 5
  3.2 Part-of-speech-based Analysis of Identifiers ........ 6

4 Grammar Pattern Definitions ............................ 8
  4.1 Annotating Words in Identifiers .................... 10

5 Methodology ............................................... 11
  5.1 5-fold test set ......................................... 14
  5.2 Quality Model Measurements ............................ 14
  5.3 Choosing a machine learning approach ................. 15
  5.4 Features ............................................... 15
CONTENTS

5.4.1 Existing Features ........................................ 16
5.4.2 New Features .............................................. 17
5.4.3 Removed and Failed Features ............................... 18
5.4.4 Feature Testing ............................................ 19
5.5 Tagger Scalability Timing ..................................... 21

6 Analysis & Discussion 22

6.1 Tagger Accuracy ............................................... 22
  6.1.1 5-fold Test Results ..................................... 24
  6.1.2 Unseen Test Set Results ................................. 26
6.2 Tagger Scalability ........................................... 27

7 Threats to Validity 30

8 Conclusion 32
List of Tables

4.1 Part-of-Speech Categories to be Used by POS Tagger .... 9
5.1 Distribution of Annotations in Training and Test Sets .... 12
5.2 Systems used to create training (unbolded) and unseen test (bolded) sets ................................. 13
5.3 F1 weighted importances for best features ................. 20
5.4 Balanced accuracy importances for best features .......... 20
5.5 Accuracy importances for best features .................. 20
6.1 Benchmark Tagger quality measurement metrics for each annotation category ............................ 23
6.2 Benchmark Tagger’s average quality measurement metrics .... 23
6.3 Scalable Tagger quality measurement metrics for each annotation category ............................. 25
6.4 Scalable Tagger’s average quality measurement metrics .... 25
Chapter 1

Introduction

The ability of a software engineer to comprehend code that others have written is an integral part of the software engineering process [62]. In fact, it is estimated that developers spend more time reading and comprehending code than actually writing the code itself. A developer’s increased comprehension, while reading other’s code, can lead to a decrease in time spent reading code, a reduction in developer stress, and better overall code development [62]. One of the primary ways that code is understood is through identifiers, which compromise approximately 70 percent of characters found in code [62]. Improving the comprehension of these identifiers can therefore have the aforementioned desired positive effects.

One way to begin addressing the problem of source code comprehension is to understand how identifiers are connected with program behavior. This can be accomplished by using a part-of-speech tagger [62]. A part-of-speech tagger is a natural language processing technique in which words in a sentence
are annotated based on their linguistic role and how they interact with surrounding words. Part-of-speech tags can be used to partially understand how an identifier conveys program behavior [64]. The majority of part-of-speech taggers deal with the annotation and tagging of conventional text, however, the tagger addressing the problem of code comprehension needs to tag source code identifiers. This issue has been faced before as although part-of-speech tagging is the most popular method for natural language semantics, it has been widely inaccurate and therefore been deemed untrustworthy. [62]

Prior researchers have attempted to solve this issue by creating an ensemble part-of-speech tagger for tagging identifiers in source code [62]. The tagger was an ensemble of SWUM, POSSE, and Stanford taggers and used the decision tree and random forest machine learning techniques in order to achieve significant results [62]. The goal of this thesis is to create a part-of-speech tagger that maintains or improves on the state-of-the-art accuracy [62] while significantly improving the tagging rate.

The rest of this thesis is organized as follows: Chapter 2 details the motivations behind the creation of the scalable part-of-speech tagger, and lists the research questions that this study sets out to answer. Chapter 3 is the related work that is relevant to key concepts in the thesis. Chapter 4 details grammar pattern definitions and gives context to the part-of-speech annotation. Chapter 5 details the methodology behind the Scalable Tagger’s creation. Chapter 6 presents the evaluation of the Scalable Tagger and gives answers to the research questions. Chapter 7 goes into potential threats to the validity of the thesis. Lastly, Chapter 8 summarizes everything in the conclusion.
Chapter 2

Research Objective

2.1 Motivation and Contribution

The research in this thesis, as stated in Chapter 1, builds upon the research of an ensemble part-of-speech tagger [62]. The Ensemble Tagger achieved an accuracy rate of 75% for tagging identifiers and an 86% accuracy rate for tagging at the word level, which is an increase of 17% from the closest individual tagger which it’s composed of [62]. Despite its apparent success, the Ensemble Tagger has a major ineptitude: it can only identify one identifier per second. This flaw significantly limits its capability to scale to large source code bases as at this rate tagging one million identifiers could take longer than 12 days. This thesis has two main goals: (1) The creation of a scalable part-of-speech tagger that can tag identifiers in source code with at least the same accuracy as the state-of-the-art Ensemble Tagger, and (2) the Scalable Tagger must significantly improve its tagging rate relative to the state-of-the-art Ensemble Tagger.
Tagger. In the long run, this will help support program comprehension research and tools. In particular it will support further research into grammar patterns which are critical to understanding the naming structures that developers use in code [64].

2.2 Research Questions

- **RQ1: What is the accuracy of this approach compared to the state of the art Ensemble Tagger?**
  This research question seeks to determine whether increasing the tagging rate is detrimental to the Scalable Tagger’s overall accuracy in annotating source code identifiers. We base the success of the Scalable Tagger on whether it achieves an equal or greater accuracy metrics.

- **RQ2: How scalable is this approach compared to the state of the art Ensemble Tagger?**
  This research question seeks to determine the increased tagging rate of the developed scalable part-of-speech tagger compared to the state-of-the-art, Ensemble Tagger. The increase in the tagging rate of the Scalable Tagger will determine its scalability, especially in contrast to the Ensemble Tagger’s inability to scale, which was its major deficiency [62].
Chapter 3

Related Work

Many studies addressed challenges to software maintenance in general [1,2,3,4, 5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31, 32,33,34,35,36,38,39,40,41,42,43,44,45,46,48,49,51,52,53,54,55,56,57,58,59, 60,61,65,66,68,69,70,71,72,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89, 90,91,94], and program comprehension in particular. Part-of-speech tagging is a commonly used method to analyze code in order to understand underlying relationships amongst identifiers. This has inspired numerous studies to be performed and papers to be written about this topic.

3.1 Part-of-Speech Taggers

Newman et al. [62] created an ensemble part-of-speech tagger for annotating identifiers in source code. The tagger comprised the SWUM, POSSE, and Stanford models, and achieved a 75% success rate at tagging identifiers and an
CHAPTER 3. RELATED WORK

86% rate tagging at the word level [62]. This tagger is used as a reference in this paper as this study is meant to address its slow tagging rate.

Yitagesu et al. [95] developed a part-of-speech tagger for security vulnerability descriptions. That study set out to create a tagger that could tag SVDs much more accurately than a word-level part-of-speech tagger. Through the use of a neural network, the tagger was able to achieve a 93% accuracy for tagging SVDs.

Gupta et al. [47] developed a part-of-speech tagger that also specializes in tagging source code identifiers. This tagger uses POSSE (POS tagger for Software Engineering) to achieve an 11-20% increase in accuracy over other traditional taggers used for the same purpose.

Toutanova et al. [92] developed a part-of-speech tagger which optimizes for maximum entropy in the text by enriching tagging information sources. This tagger improved the accuracy of tagging individual words by using additional context provided by outside information.

Partachi et al. [67] researched Software Engineers’ usage of mixed natural language and source code in their communications and developed a part-of-speech tagger, called POSIT to solve the problems of language identification and token tagging amongst this mixed text. The tagger could tag code tokens with their part-of-speech with an accuracy of 85.6%.

3.2 Part-of-speech-based Analysis of Identifiers

Newman et al. [63] describe the categorization of source code identifiers rather than their common place connotations. This study categorized source code
identifiers by their type, behavior, context, use, and meaning to provide a more descriptive definition to the identifier. This study saw a high correlation between the identifiers underlying meaning and the meaning of the words that comprised it.

Caprile et al. [37] research the intrinsic value in identifier names and how restructuring identifiers can improve their meaningfulness. The study was able to replace words in identifiers with little meaning with other words that retained the same connotation, but expressed a more significant meaning to the reader.

Peruma et al. [73] researched the importance of names of test methods and how proper use of grammar patterns can assist developers in better understanding test methods over time. The study confirmed the usefulness of grammar patterns in the understanding of test method names, and in understanding how they relate to code behavior.

Høst et al. [50] extracted rules for naming methods in order to point out "naming bugs" in code. An automatic suggestion of more suitable names was studied and presented to have a high correlation between the name given to a method and the method’s underlying functionality.

Wu et al. [93] created a process for identifying non-descriptive test method names in JUnit tests to increase the comprehension of the purpose of the test method to the reader. This was done by providing information on how the test method name can be improved with a 95% accuracy rate in determining which method names needed to be improved.
Chapter 4

Grammar Pattern Definitions

By definition, the usage of a part-of-speech tagger involves the annotation of words with their parts of speech in grammar in a given context. This tagging of identifiers in code then results in the formation of a grammar pattern. A grammar pattern is the sequence of part-of-speech tags assigned to words within an identifier. For example, an identifier called “GetDogData” would start with the tagging of individual words (Get, Dog, and Data), and result in a grammar pattern of verb noun-adjunct noun. This grammar pattern can then be used to classify identifiers that share the same grammar patterns. This then can be used to correlate identifiers that have different meanings and connotations but share similarities in their program behavior.
### Table 4.1: Part-of-Speech Categories to be Used by POS Tagger

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Expanded Form</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>noun</td>
<td>Disneyland, shoe, faucet, mother</td>
</tr>
<tr>
<td>DT</td>
<td>determiner</td>
<td>the, this, that, these, those, which</td>
</tr>
<tr>
<td>CJ</td>
<td>conjunction</td>
<td>and, for, nor, but, or, yet, so</td>
</tr>
<tr>
<td>P</td>
<td>preposition</td>
<td>behind, in front of, at, under, above</td>
</tr>
<tr>
<td>NPL</td>
<td>noun plural</td>
<td>Streets, cities, cars, people, lists</td>
</tr>
<tr>
<td>NM</td>
<td>noun modifier</td>
<td>red, cold, hot, scary, beautiful, small</td>
</tr>
<tr>
<td>V</td>
<td>Verb</td>
<td>Run, jump, spin</td>
</tr>
<tr>
<td>VM</td>
<td>Verb modifier</td>
<td>Very, loudly, seriously, impatiently</td>
</tr>
<tr>
<td>PR</td>
<td>pronoun</td>
<td>she, he, her, him, it, we, they, them</td>
</tr>
<tr>
<td>D</td>
<td>digit</td>
<td>1, 2, 10, 4.12, 0xAF</td>
</tr>
<tr>
<td>PRE</td>
<td>preamble</td>
<td>Gimp, GLEW, GL, G, p, m, b</td>
</tr>
</tbody>
</table>
4.1 Annotating Words in Identifiers

As identifiers are broken down to have their comprised vocabulary annotated, it is important to label and categorize these annotations. The categories used in the part-of-speech tagger are all included in Table I. These categories are the most common parts of speech that are recognizable from common English grammar parts of speech. However, there are two categories in Table I that are uncommon and need to be addressed, preambles and noun modifiers. In the context of code identifiers, and in this thesis, a preamble is an abbreviation that occurs at the beginning of an identifier to provide metadata, name-space an identifier, or highlight the identifier’s type without changing the reader’s understanding of the identifier. For example, the preamble p_ gives context that the identifier is being used as a pointer. Noun modifiers are defined in this thesis as an adjective or a noun that is being used as an adjective, called a noun-adjunct. An example of a noun-adjunct can be seen in the identifier "GetDogData" as the noun Dog is being used to describe the noun Data, making the word Dog a noun-adjunct in this identifier.
Chapter 5

Methodology

In order to evaluate the effectiveness of the Scalable Tagger created in this thesis against the Ensemble Tagger, both taggers will be trained on the same data-set. The process and means of sourcing the data-set used for training the Ensemble Tagger are not included in this thesis, but can be found in the paper on the Ensemble Tagger [62] if desired. The data-set is comprised of 1,335 manually annotated identifiers to label their part-of-speech, which were sourced from 20 different software systems. The sourcing of this data is broken down and shown in Table 5.1. The data-set was then broken down into a training set and test set in order to validate the results of the Scalable Tagger across different data sets. The test set was comprised of the identifiers from 5 of the software systems and the remaining 15 comprised the training set, from which more identifiers were extracted in order to maintain the 1,335 identifier basis.

In order to reduce over-fitting, 5-fold validation is used on the training
Table 5.1: Distribution of Annotations in Training and Test Sets

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Training Set</th>
<th>Unseen Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>CJ</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>DT</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>N</td>
<td>1149</td>
<td>322</td>
</tr>
<tr>
<td>NM</td>
<td>1520</td>
<td>415</td>
</tr>
<tr>
<td>NPL</td>
<td>220</td>
<td>78</td>
</tr>
<tr>
<td>P</td>
<td>91</td>
<td>32</td>
</tr>
<tr>
<td>PRE</td>
<td>83</td>
<td>33</td>
</tr>
<tr>
<td>V</td>
<td>330</td>
<td>81</td>
</tr>
<tr>
<td>VM</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3449</strong></td>
<td><strong>977</strong></td>
</tr>
</tbody>
</table>

data. The training data sourced from 15 software systems is split into 5 smaller train-test sets, or folds, which will be referred to as the "5-fold test set". The test set comprised of identifiers that are removed from the training process is referred to as the "unseen test set". The distribution of the annotations from the identifiers into the training set and unseen test set can be seen in Table 5.2.
Table 5.2: Systems used to create training (unbolded) and unseen test (bolded) sets

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (kloc)</th>
<th>Age (years)</th>
<th>Language(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>junit4</td>
<td>30</td>
<td>19</td>
<td>Java</td>
</tr>
<tr>
<td>mockito</td>
<td>46</td>
<td>9+</td>
<td>Java</td>
</tr>
<tr>
<td>okhttp</td>
<td>54</td>
<td>6</td>
<td>Java</td>
</tr>
<tr>
<td>antlr4</td>
<td>92</td>
<td>27</td>
<td>Java/C/C++/C#</td>
</tr>
<tr>
<td>openFrameworks</td>
<td>130</td>
<td>14</td>
<td>C/C++</td>
</tr>
<tr>
<td>jenkins</td>
<td>156</td>
<td>8</td>
<td>Java</td>
</tr>
<tr>
<td>irrlicht</td>
<td>250</td>
<td>13</td>
<td>C/C++</td>
</tr>
<tr>
<td>kdevelop</td>
<td>260</td>
<td>19</td>
<td>C/C++</td>
</tr>
<tr>
<td>Ogre</td>
<td>370</td>
<td>14</td>
<td>C/C++</td>
</tr>
<tr>
<td>quantlib</td>
<td>370</td>
<td>19</td>
<td>C/C++</td>
</tr>
<tr>
<td>coreNLP</td>
<td>582</td>
<td>6</td>
<td>Java</td>
</tr>
<tr>
<td>swift</td>
<td>601</td>
<td>5</td>
<td>C++/C</td>
</tr>
<tr>
<td>calligra</td>
<td>660</td>
<td>19</td>
<td>C/C++</td>
</tr>
<tr>
<td>gimp</td>
<td>777</td>
<td>23</td>
<td>C/C++</td>
</tr>
<tr>
<td>telegram</td>
<td>912</td>
<td>6</td>
<td>Java/C/C++</td>
</tr>
<tr>
<td>opencv</td>
<td>1000</td>
<td>19</td>
<td>C/C++</td>
</tr>
<tr>
<td>elasticsearch</td>
<td>1300</td>
<td>9</td>
<td>Java</td>
</tr>
<tr>
<td>bullet3</td>
<td>1300</td>
<td>10+</td>
<td>C/C++/C#</td>
</tr>
<tr>
<td>blender</td>
<td>1600</td>
<td>21</td>
<td>C/C++</td>
</tr>
<tr>
<td>grpc</td>
<td>1800</td>
<td>5</td>
<td>C++/C/C#</td>
</tr>
</tbody>
</table>
5.1 5-fold test set

Another form of validation of the Scalable Tagger is through 5-fold validation. During the training of the model, the training data comprised of 1,335 identifiers from 15 software systems are split into 5 smaller folds. This allows for a split amongst the folds for a percentage to be used as training data while the remaining is used as testing data to validate the trained model. After the data is split into the 5 folds, 70% is used as training data, and the remaining 30% is used as testing data. This is repeated 5 times until all of the sets have been used for testing. This is an application of k-fold cross-validation sets in which the value of k has been set to 5. This is appropriate with choices from researchers who often use k values of 5 or 10, from which 5 is used considering the size of our data-set and the distribution of the annotations. Each testing round is accompanied by the collection of metrics to evaluate the effectiveness of the model, which is described in the next section.

5.2 Quality Model Measurements

The quality of the Scalable Tagger is measured using common metrics for categorization problems. The metrics used are Accuracy, Precision, Recall, and F1 Score. Another metric used is the balanced accuracy of the 5-fold testing which is the average of each average accuracy for each annotation category (i.e., N, NM, CJ, etc.). This allows for each part-of-speech annotation to be equally represented, which helps with unbalanced data-sets by giving more weight to underrepresented annotations.
5.3 Choosing a machine learning approach

The Scalable Tagger uses a machine learning model in order to annotate identifiers with a part-of-speech. The model must be trained to follow a primary machine learning approach from which the following were evaluated for use: Random Forest, Decision Tree, XGBoost, Logistic Regression, Support Vector Classification, and K-Nearest Neighbors. From these Random Forest and Decision Tree were selected to be used as the primary approaches due to their out-performance of the other approaches in terms of the quality model metrics. The building of the models requires a training and testing set, which is from a splitting of the training data in which 70% is used as the training data and 30% is used as the testing data. Through further analysis of the performance of the Random Forest and Decision Tree approaches the Decision Tree approach was determined to be the better approach. Throughout the formation of the Scalable Tagger Random Forest would reach a stagnation to how well it performed, while Decision Tree continued to improve, eventually surpassing the performance of Random Forest. Random Forest is also a much slower approach than Decision Tree which is counterproductive to the goal of creating a scalable part-of-speech tagger.

5.4 Features

Machine learning algorithms must be trained on a set of features in order to derive patterns and predict outputs from given inputs. The basis for features used for the Scalable Tagger is based on relevant features used for the Ensemble
Tagger. The relevancy of features is determined by how they impact the quality model measurements detailed in Chapter 5.2. New features were developed in order to improve these quality measurements, thus improving the Scalable Tagger. Certain features did not end up being used either due to their negative impact to the Scalable Tagger. The testing of the features is detailed further in Chapter 5.4.4.

5.4.1 Existing Features

The Scalable Tagger uses the finding of the Ensemble Tagger as a basis of the features used for training. These existing features are those from the Ensemble Tagger continuing to be used due to their continued relevance. They are as follows:

1. Normalized position: We normalized the position metric described above such that the first word in the identifier is in position 1, all middle words are in position 2, and the last word is in position 3. For example, given an identifier: GetXMLReaderHandler, Get is in position 1, XML is in position 2, Reader is in position 2 and Handler is in position 3. The reason for this feature is to mitigate the sometimes-negative effect of very long identifiers.

2. Identifier size: The length, in words, of the identifier of which the word was originally part.
5.4.2 New Features

The training of the model focused on creating new features in order to better the Scalable Tagger’s quality measurement metrics. Each new feature provides new insight into understanding key similarities and differences between part-of-speech categories. The new features are as follows:

1. **Word Embeddings**: Word vectorization is applied to each word to analyze the similarities and differences between them as they relate to other words. Word vectorization is the process of converting words in text into numerical arrays, in order for their use in training the model. Gensim, a popular natural language processing library, is used to provide a pre-trained model of a plethora of word vectors. Since this feature is an array of numbers, each index in the array is used as a separate feature that comprises the whole. The length of the array depends on the Gensim model used, which is "fasttext-wiki-news-subwords-300" making there be 300 Word Embedding features.

2. **Last letter**: The ASCII value of the last letter of the word. This provides better contrast between nouns and noun plurals as the difference usually occurs with noun plurals ending in the letter "S". For example the difference between "Dog" being a noun and "Dogs" being a noun plural, in which the last letter of the word being the key differentiator.

3. **Digit**: Determines whether the word is a digit or not. Is set to a value of 0 if the word is a digit or a 1 if the word is not.

4. **Word Length**: The length of the word. This can be used to differentiate
preambles, prepositions and digits which are typically shorter in length from nouns, noun plurals and verbs which are typically longer.

5.4.3 Removed and Failed Features

The Scalable Tagger detailed in this thesis used the features of the Ensemble Tagger as a basis for its starting features. This also means that certain features were removed due to their lack of relevance. The SWUM, POSSE, and Stanford annotation features were three part-of-speech taggers that were used to comprise the ensemble nature of the Ensemble Tagger. These features are the source of the slow tagging rate experienced by the Ensemble Tagger, which is the grounds for their removal. The Word, Data Type, Position, and Context features were also removed as they had a negligible effect on the Scalable Tagger’s quality measurement metrics, which is further explained in Chapter 5.4.4. More information on these discarded features can be found in the Ensemble Tagger paper [62].

Other features were attempted to increase the quality measurement metrics besides those mentioned in Chapter 5.4.2, but they did not result in improvements. Further attempts at the use of word vectors were made through cosine correlation between word vectors. These cosine similarities were flawed in their over-fitting to given data, and when corrected presented no statistical improvements. Other features tried involved determining if a word was a commonly used conjunction or determiner. These part-of-speech categories are comprised of limited sets, however, this approach was unfruitful. These features were not useful in the end, but the process of finding statistically beneficial features
involves an arduous period of trial and error.

5.4.4 Feature Testing

Further testing was used to analyze the features to ensure which features were statistically significant to the quality measurement metrics detailed in Chapter 5.2. The analysis of the features involved two techniques: Drop-column feature importance and permutation importance. Drop-column feature importance is the training of the model on every possible subset of the available features, which can allow for the best possible subset to be determined. Permutation importance involves first training the model. Then each feature is randomly shuffled out, and the resulting importance is measured from the difference in the quality measurement metrics. This then gives a value to the importance of each feature, and whether it should be used. Each feature was run using these techniques to value their importance, from which the features stated in Chapters 5.4.1 and 5.4.2 were all validated. Testing using permutation importance was can be shown on these validated features in Tables 5.3, 5.4, and 5.5, which detail the importance of the features relating to F1 measure, balanced accuracy, and accuracy. The key exception is that of Word Length, which can be shown with no importance in all of these categories. Unlike the other features that have shown no importance, Word Length did have noticeable effects on the quality model measurements when removed. It should be noted, that these techniques do not represent non-linear relationships between features and the target outcome and do not always reflect the true importance of a feature, as can be seen in the instance of the Word Length feature.
### Table 5.3: F1 weighted importances for best features

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>F1 Weighted Importances</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized Position</td>
<td>0.36 0.37 0.36 0.35 0.36</td>
<td>0.36</td>
</tr>
<tr>
<td>Last Letter</td>
<td>0.09 0.09 0.09 0.09 0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>Identifier Size</td>
<td>0.03 0.03 0.03 0.03 0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>Digit</td>
<td>0.01 0.01 0.01 0.01 0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Word Length</td>
<td>0.00 0.00 0.00 0.00 0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Word Embeddings</td>
<td>0.54 0.54 0.55 0.54 0.54</td>
<td>0.54</td>
</tr>
</tbody>
</table>

### Table 5.4: Balanced accuracy importances for best features

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Balanced Accuracy Importances</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized Position</td>
<td>0.34 0.35 0.31 0.31 0.30</td>
<td>0.32</td>
</tr>
<tr>
<td>Last Letter</td>
<td>0.12 0.13 0.12 0.14 0.14</td>
<td>0.13</td>
</tr>
<tr>
<td>Identifier Size</td>
<td>0.02 0.03 0.02 0.02 0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>Digit</td>
<td>0.08 0.08 0.08 0.08 0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>Word Length</td>
<td>0.00 0.00 0.00 0.00 0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Word Embeddings</td>
<td>1.33 1.38 1.38 1.29 1.34</td>
<td>1.34</td>
</tr>
</tbody>
</table>

### Table 5.5: Accuracy importances for best features

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>Accuracy Importances</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized Position</td>
<td>0.37 0.36 0.37 0.36 0.35</td>
<td>0.36</td>
</tr>
<tr>
<td>Last Letter</td>
<td>0.08 0.08 0.08 0.08 0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>Identifier Size</td>
<td>0.03 0.03 0.03 0.03 0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>Digit</td>
<td>0.01 0.01 0.01 0.01 0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Word Length</td>
<td>0.00 0.00 0.00 0.00 0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Word Embeddings</td>
<td>0.53 0.51 0.51 0.49 0.51</td>
<td>0.51</td>
</tr>
</tbody>
</table>
5.5 Tagger Scalability Timing

The essential flaw of the previous state-of-the-art part-of-speech tagger for source code identifiers, the Ensemble Tagger, was its inability to scale. The Scalable Tagger will be made scalable through the removal of the ensemble nature while attempting to maintain the accuracy and other quality measurement metrics set by the Ensemble Tagger.

The tagging rate for the Scalable Tagger will be determined by timing how long the Scalable Tagger takes to tag the unseen test set. This allows for the evaluation of the Scalable Tagger’s tagging rate on a set of data that was not used in training. The tagging rate will therefore be calculated using the following equation:

\[
Tagging \ Rate = \frac{\text{\# of words tagged}}{\text{Total tagging time}} \quad (5.1)
\]

The unseen test set is comprised of 977 words from 384 identifiers, causing the tagging rate of the unseen test set to be:

\[
Tagging \ Rate = \frac{977 \ \text{words}}{\text{Total tagging time}} \quad (5.2)
\]

The tagging rate has units of part-of-speech annotation, or tag, per second (tag/sec). One thing to mention is that the time it takes for the new features created for the Scalable Tagger was not applied to the time it takes for tagging. It is assumed that the features have been applied to the data set before the tagging is initiated.
Chapter 6

Analysis & Discussion

6.1 Tagger Accuracy

RQ1: What is the accuracy of this approach compared to the state-of-the-art Ensemble Tagger?

The evaluation of the Scalable Tagger’s accuracy is determined in two ways. The first method by using the 5-fold cross validation, which uses a set of 1,335 manually-annotated identifiers. This allows for the evaluation of the model during its training. The second method is by running the trained model on the unseen test set of 384 identifiers. This provides an unbiased set of identifiers to provide assurance of the model’s accuracy and that it isn’t over-fitting on the training data.
### Table 6.1: Benchmark Tagger quality measurement metrics for each annotation category

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>CJ</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>0.89</td>
<td>1.00</td>
<td>0.94</td>
<td>8</td>
</tr>
<tr>
<td>DT</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>4</td>
</tr>
<tr>
<td>N</td>
<td>0.75</td>
<td>0.85</td>
<td>0.80</td>
<td>333</td>
</tr>
<tr>
<td>NM</td>
<td>0.83</td>
<td>0.94</td>
<td>0.88</td>
<td>509</td>
</tr>
<tr>
<td>NPL</td>
<td>0.20</td>
<td>0.05</td>
<td>0.08</td>
<td>77</td>
</tr>
<tr>
<td>P</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>24</td>
</tr>
<tr>
<td>PRE</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>31</td>
</tr>
<tr>
<td>V</td>
<td>0.57</td>
<td>0.55</td>
<td>0.56</td>
<td>76</td>
</tr>
<tr>
<td>VM</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 6.2: Benchmark Tagger’s average quality measurement metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.77</td>
</tr>
<tr>
<td>Balanced Accuracy</td>
<td>0.34</td>
</tr>
<tr>
<td>Weighted F1 Score</td>
<td>0.72</td>
</tr>
<tr>
<td>Weighted Precision</td>
<td>0.69</td>
</tr>
<tr>
<td>Weighted Recall</td>
<td>0.77</td>
</tr>
</tbody>
</table>
6.1.1 5-fold Test Results

The first run of the Scalable Tagger, before any improvements were made, involved using a stripped down version of the original Ensemble Tagger [62], excluding the SWUM, POSSE, and Stanford annotations which were the cause of the Ensemble Tagger’s detrimental slow tagging rate. This established a benchmark for improvement upon which the Scalable Tagger’s improved results are compared. The Benchmark Tagger is shown in Tables 6.1 and 6.2. The evaluation of the Scalable Tagger is run using the Decision Tree machine learning approach as detailed in Chapter 5. The results from this evaluation can be seen in Tables 6.3 and 6.4. Both tables, 6.2 and 6.4, give the averaged results of the 5-fold evaluations which are the tagger’s quality measurement metrics, which include the tagger’s accuracy, balanced accuracy, weighted F1 score, weighted precision, and weighted recall. Tables 6.1 and 6.3 detail the precision, recall, and F1 score measured for each annotation category, which characterize the tagger’s effectiveness across each category.

The Benchmark Tagger can be seen to have many inadequacies, as shown in Tables 6.1 and 6.2. The accuracy of the Benchmark Tagger, at 77%, is lower than that of the state-of-the-art Ensemble Tagger, at 86%. This needs to be improved upon in order to meet the set standards. The balanced accuracy, however, of the Benchmark is very low at 34%, as compared to the Ensemble’s balanced accuracy of 55%, indicating that many categories are being mis-annotated by the Benchmark Tagger. This can be clearly seen as 5 of the 10 annotation categories are not tagged accurately at all, with 0% measurements for precision, recall, and F1 score. This contrast between ac-
Table 6.3: Scalable Tagger quality measurement metrics for each annotation category

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>CJ</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>1.00</td>
<td>0.88</td>
<td>0.93</td>
<td>8</td>
</tr>
<tr>
<td>DT</td>
<td>0.80</td>
<td>1.00</td>
<td>0.89</td>
<td>4</td>
</tr>
<tr>
<td>N</td>
<td>0.92</td>
<td>0.86</td>
<td>0.89</td>
<td>333</td>
</tr>
<tr>
<td>NM</td>
<td>0.87</td>
<td>0.94</td>
<td>0.90</td>
<td>509</td>
</tr>
<tr>
<td>NPL</td>
<td>0.89</td>
<td>0.82</td>
<td>0.85</td>
<td>77</td>
</tr>
<tr>
<td>P</td>
<td>0.80</td>
<td>0.83</td>
<td>0.82</td>
<td>24</td>
</tr>
<tr>
<td>PRE</td>
<td>0.89</td>
<td>0.52</td>
<td>0.65</td>
<td>31</td>
</tr>
<tr>
<td>V</td>
<td>0.68</td>
<td>0.63</td>
<td>0.65</td>
<td>76</td>
</tr>
<tr>
<td>VM</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.4: Scalable Tagger’s average quality measurement metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.87</td>
</tr>
<tr>
<td>Balanced Accuracy</td>
<td>0.75</td>
</tr>
<tr>
<td>Weighted F1 Score</td>
<td>0.87</td>
</tr>
<tr>
<td>Weighted Precision</td>
<td>0.87</td>
</tr>
<tr>
<td>Weighted Recall</td>
<td>0.87</td>
</tr>
</tbody>
</table>
accuracy and balanced accuracy is due to the heavy support of nouns(N) and noun modifiers(NM), which are more accurately tagged than other annotation categories.

The Scalable Tagger improves on the inadequacies of the Benchmark Tagger in order to surpass the accuracy of the Ensemble Tagger, as can be seen in Tables 6.3 and 6.4. There are enhancements in every annotation category, except for that of VM, which can be attributed to its low support. The accuracy of the Scalable Tagger, at 87%, increased 10% from the Benchmark Tagger, as well as 1% from that of the Ensemble Tagger [62]. The biggest improvement is the Scalable Tagger’s balance accuracy, at 75% is a 41% improvement over the Benchmark Tagger, and a 20% improvement over the Ensemble Tagger. This is due to the focus on addressing the tagging failures of the conjunction(CJ), determinant(DT), noun plural(NPL), preposition(P), and preamble(PRE) annotation categories which allows for more accurate tagging on a wider variety of identifiers that aren’t comprised of the more common noun and noun modifiers.

6.1.2 Unseen Test Set Results

The unseen test set is an unbiased test set used to validate the trained model on data that hasn’t been used to train it. The model was run on the unseen test set, resulting in the accuracy evaluations shown in Table 6.5. This result validates the finding from the 5-fold test validation as the accuracy of the Scalable Tagger remains congruent in both testing methods across most of the annotations, except for conjunctions which has a low sample size. The
testing of the unseen test set resulted in a greater accuracy of 89%, which is 2% greater than that of the 5-fold validation. This demonstrates a very low chance of over-fitting in the model, and that the Scalable Tagger’s improved accuracy holds true.

### 6.2 Tagger Scalability

**RQ2:** How scalable is this approach compared to the state-of-the-art Ensemble Tagger?

The lack of scalability of the Ensemble Tagger is a major limitation. This limits its ability to perform tagging on large scale code bases, as users would
like to use a tool that is timely and effective. The lack of scalability is due to its slow tagging rate of approximately 1 part-of-speech annotation per second. At this rate, the tagging of one million words from a large scale repository with a very large amount of source code identifiers would take 12 days. The Scalable Tagger attempts to have its tagging rate be swift in order to correct for this limitation.

The scalability, through means of the tagging rate, has been evaluated by measuring the amount of time the Scalable Tagger takes to tag all the identifiers in the unseen test set. The time that the Scalable Tagger took to annotate all 977 words from 384 identifiers is 3.984 seconds. The Scalable Tagger’s tagging rate can then be calculated by using Equation 5.2 using the newly measured time:

\[ \text{Tagging Rate} = \frac{977 \text{ words}}{3.984 \text{ seconds}} \]  

This computes to a tagging rate of approximately 245 tags per second. This rate is 245 times faster than that of the Ensemble Tagger, which clearly improves upon its slow tagging rate. This allows for a much more Scalable Tagger, as in the aforementioned example of a large repository with one million words to be tagged, the Scalable Tagger would only take approximately 1.2 hours to tag every word. This is a great improvement from the 12 days that the Ensemble Tagger would take to achieve the same task.

It is also important to mention that this feat would not be as important without the achievements evaluated in Chapter 6.1. The ability for the Scalable Tagger to be much faster in its tagging rate, thus being scalable would not be as useful if the Scalable Tagger’s accuracy, balanced accuracy, precision, recall,
and F1-score suffered as a consequence.
Chapter 7

Threats to Validity

This thesis sources its datasets from those used by the Ensemble Tagger. Therefore, one concern is that the collection of 1,335 identifiers could have been improperly annotated by the authors of the ensemble tagging paper [62], leading to imperfections in the data. This was validated by the cross-validation amongst annotators for all grammar patterns, and by having the data sets publicly available allowing for corrections from outside sources.

Another concern is the confidence we have that the features selected in the training of the model have a true cause-and-effect relationship with the evaluation of the Scalable Tagger’s accuracy and other quality measurement metrics and that the positive evaluation of these measurements are not caused by other external factors. The use of drop-column and permutation importances validated that the features were directly affecting these measurements, as they were tested through 5-fold cross-validation. Outside external factors were mitigated by keeping the same training and testing sets for the Scalable
Tagger as the Ensemble Tagger.

Over-fitting is another threat that must be acknowledged. In order to ensure that the Scalable Tagger is not only valid on the data used to train it, the Scalable Tagger went through the testing method of 5-fold cross-validation, which used train-test folds to ensure that each sample of training data was also used to test the Scalable Tagger. This was compounded by the use of an external testing set, unseen to the Scalable Tagger during training. The validation that both of these testing methods proved to have positive evaluations on the Scalable Tagger’s quality measurement metrics indicates that over-fitting has been properly mitigated.

Many machine learning approaches, including Random Forest, Decision Tree, XGBoost, Logistic Regression, Support Vector Classification, and K-Nearest Neighbors, were considered for this model along with numerous features to train them. This resulted in the usage of a model implementing the Decision Tree approach and the features discussed in Chapter 5. These were then evaluated using the metrics of accuracy, balanced accuracy, recall, precision, and F1-score. This resulted in a new state-of-the-art tagger according to these metrics that is scalable, however, this should not be taken as the final result in source code identifier tagging. It would be a naive judgment that the Scalable Tagger could not be potentially improved upon since the Scalable Tagger itself is an improvement on the previous state-of-the-art tagger. To mitigate this, the data-sets and code for the Scalable Tagger have been made publicly available on GitHub.
Chapter 8

Conclusion

This thesis has detailed the creation and evaluation of a new scalable part-of-speech tagger. The Scalable Tagger was modeled using the Decision Tree machine learning algorithm and trained using various features to target the better tagging of specific annotation categories while allowing for a rapid tagging rate. This resulted in the Scalable Tagger having an 87% accuracy, which is a 1% increase over its predecessor, a balanced accuracy of 75%, which is a 20% improvement over its predecessor, and a tagging rate of 245 annotations per second, which is 245 times faster than its predecessor.

This part-of-speech tagger has, therefore, become the new state-of-the-art tagger for the annotation of source code identifiers. It has addressed the major inadequacy of the Ensemble Tagger, its predecessor, of lacking scalability due to its slow tagging rate. The Scalable Tagger detailed in this thesis is deemed successful as it has addressed this thesis’ main goal of being scalable while maintaining the accuracy set by the Ensemble Tagger.
Future work on the subject of the tagging of source code identifiers may arise around handling certain limitations of this project. Future taggers can implement other machine learning algorithms that were not tested in this thesis in order to increase the scalability and/or the accuracy of the tagger. The usage of a larger data set to train the model, whose annotations are validated by more analysts, would aid in allowing the tagger to recognize patterns that may not be present in the data set used for this thesis.
Bibliography


[4] Wajdi Aljedaani, Mohamed Wiem Mkaouer, Anthony Peruma, and Stephanie Ludi. Do the test smells assertion roulette and eager test im-


[29] Hussein Alrubaye, Stephanie Ludi, and Mohamed Wiem Mkaouer. Comparison of block-based and hybrid-based environments in transfer-


[40] Moataz Chouchen, Ali Ouni, Mohamed Wiem Mkaouer, Raula Gaikovina Kula, and Katsuro Inoue. Whoreview: A multi-objective search-based ap-


[56] Mohamed Wiem Mkaouer, Marouane Kessentini, Slim Bechikh, Kalyanmoy Deb, and Mel Ó Cinnéide. High dimensional search-based software engineering: finding tradeoffs among 15 objectives for automating software


[79] Islem Saidani, Ali Ouni, Moataz Chouchen, and Mohamed Wiem Mkaouer. On the prediction of continuous integration build failures using


[85] Islem Saidani, Ali Ouni, Mohamed Wiem Mkaouer, and Aymen Saied. Towards automated microservices extraction using muti-objective evolution-


