IMPLICATIONS OF PUNCTUATION MARK NORMALIZATION
ON TEXT RETRIEVAL

Eungi Kim, B.S., M.S.

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

August 2013

APPROVED:

William E. Moen, Major Professor
Brian C. O’Connor, Committee Member
Christina Wasson, Committee Member
Suliman Hawamdeh, Chair of the Department
of Library and Information Sciences
Herman Totten, Dean of College of
Information
Mark Wardell, Dean of the Toulouse Graduate
School
Kim, Eungi. **Implications of punctuation mark normalization on text retrieval.** Doctor of Philosophy (Information Science), August 2013, 278 pp., 41 tables, 20 figures, references, 94 titles.

This research investigated issues related to normalizing punctuation marks from a text retrieval perspective. A punctuated-centric approach was undertaken by exploring changes in meanings, whitespaces, words retrievability, and other issues related to normalizing punctuation marks. To investigate punctuation normalization issues, various frequency counts of punctuation marks and punctuation patterns were conducted using the text drawn from the Gutenberg Project archive and the Usenet Newsgroup archive. A number of useful punctuation mark types that could aid in analyzing punctuation marks were discovered.

This study identified two types of punctuation normalization procedures: (1) lexical independent (LI) punctuation normalization and (2) lexical oriented (LO) punctuation normalization. Using these two types of punctuation normalization procedures, this study discovered various effects of punctuation normalization in terms of different search query types. By analyzing the punctuation normalization problem in this manner, a wide range of issues were discovered such as: the need to define different types of searching, to disambiguate the role of punctuation marks, to normalize whitespaces, and indexing of punctuated terms.

This study concluded that to achieve the most positive effect in a text retrieval environment, normalizing punctuation marks should be based on an extensive systematic analysis of punctuation marks and punctuation patterns and their related factors. The results of this study indicate that there were many challenges due to complexity of language. Further, this study recommends avoiding a simplistic approach to punctuation normalization.
Copyright 2013

By

Eungi Kim
ACKNOWLEDGEMENTS

First and foremost, I want to thank God for giving me the strength and perseverance to finish this journey. I wish to express my sincere gratitude and appreciation to everyone who contributed to this dissertation. In particular, I would like to thank my committee members for their patience and unyielding support. I’m grateful to my dissertation chair Dr. William Moen. He assisted with every aspect of completing this dissertation. We’ve had numerous detailed discussions, and his thoughtful comments were always useful. My other committee members, Dr. Brian O’Connor and Dr. Christina Wasson, whom I have known for all these years, contributed their time generously in serving on my proposal and dissertation defense. They offered me with intellectually stimulating comments, and their valuable comments helped me to avoid making a number of potential critical mistakes in this dissertation.

I am also indebted to Dr. Shawne Miksa, Ph.D. Program Coordinator at University of North Texas. She always provided me with assistance and guidance whenever I needed. I would also like to thank Dr. Suliman Hawamdeh, Dr. Linda Schamber, and Dr. Victor Prybutok for their roles in granting extension to complete my Ph.D. dissertation. I also want to thank Malaina Mobley, other staff members, and the faculty of The Library and Information Sciences Department at the University North Texas for their assistance in various matters.

Finally, this dissertation would not have been possible without the support of my family members. I would like to thank my wife Eunhee for her personal support and great patience. My precious two daughters, Minhae and Ester, have been patience with me for all these years. My sister Eunsil has been a reliable support of my life. I cannot thank enough to my father Young Jong Kim and my mother Sook Ja Kim for their endless love and support. They helped me with everything I needed to continue my education and complete this dissertation.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td></td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td></td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td></td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCTION</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>The Notion of Punctuation Normalization</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Research Motivation</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>Theoretical Framework</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>The Linguistic (Computational) Perspective</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Information Retrieval (Text Retrieval)</td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>Information Extraction (IE)</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Integrated Theoretical Framework</td>
<td></td>
<td>22</td>
</tr>
<tr>
<td>Research Questions</td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>Research Design and Data Collection</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>Limitation of the Study</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>Organization</td>
<td></td>
<td>29</td>
</tr>
<tr>
<td>CHAPTER 2 BACKGROUND AND LITERATURE REVIEW</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>Linguistic and Computational Analysis of Punctuation Marks</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>Punctuation Mark Categorization</td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>Frequency Analysis of Punctuation Marks and Punctuation Pattern</td>
<td></td>
<td>35</td>
</tr>
<tr>
<td>Punctuation Marks in Text Processing Related Procedures</td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>Tokenization</td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>Indexing and Punctuation Marks</td>
<td></td>
<td>43</td>
</tr>
<tr>
<td>Effectiveness of Search Queries and Punctuation Normalization</td>
<td></td>
<td>45</td>
</tr>
<tr>
<td>Existing Approaches to Punctuation Normalization Practices</td>
<td></td>
<td>47</td>
</tr>
<tr>
<td>Google Search Engine</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>LexusNexis</td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>Concepts Related to Punctuation Normalization</td>
<td></td>
<td>54</td>
</tr>
<tr>
<td>Character Encoding and Character Normalization</td>
<td></td>
<td>54</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1. Jones' Inter-Lexical Categories of Punctuation Marks (1997) ..............................................33
Table 2. Panchapagesan’s List of Non-Standard Words .................................................................61
Table 3. Example of Punctuation Mark Types and Tokens ............................................................79
Table 4. Possible Interpretation of Figure 3 Scenarios ........................................................................83
Table 5. Basic Criteria for Analyzing Search Results .................................................................88
Table 6. Basic Statistics of Punctuation Marks ............................................................................96
Table 7. Top 20 Infix-Punctuated Tokens (Gutenberg to Usenet) ..................................................99
Table 8. Top 20 Infix-Punctuated Tokens (Usenet to Gutenberg) ..................................................101
Table 9. Top 4 Infix Tokens with Least Relative % Differences ..................................................103
Table 10. Frequency Count of Apostrophe-s and Z’Z Punctuation Pattern .................................105
Table 11. Hyphen and Dash Variations in the Usenet and Gutenberg Datasets ............................109
Table 12. Frequency Counts of Single-Letter-Infix-Slash Token ..................................................114
Table 13. Top 10 Tokens Infix-Period in the Usenet Dataset ..........................................................116
Table 14. A Variation of Ellipses in Affix-Based Token Types .....................................................118
Table 15. Text Lines Containing the Variation of Ellipses ..............................................................119
Table 16. The Top 20 Prefix-Punctuation Marks ............................................................................120
Table 17. The Top 20 Suffix-Punctuation Marks ..........................................................................122
Table 18. The Top 20 Punctuation-Exclusive Tokens ......................................................................126
Table 19. Hyphenated Token Categorization Comparison .............................................................136
Table 20. Selected Paired-Punctuation Marks without Post-Pre Blank Spaces ..............................137
Table 21. The Most Common Tripple Punctuation Marks ............................................................146
Table 22. Examples of Numeric Token .................................................................147
Table 23. Basic Statistics of Numeric Punctuation Marks ..........................158
Table 24. The Top 20 Tokens with x.x. Punctuation Pattern .........................150
Table 25. Common Punctuation Mark Operations ........................................162
Table 26. LO Normalization Example ............................................................167
Table 27. The Punctuated String Distance (PSD) Example .............................175
Table 28. The PSD Comparison Example ........................................................176
Table 29. An Example of LI Punctuation Normalization Results ..................181
Table 30. Exact Search Queries and Normalized Character Strings ...............185
Table 31. Approximate Search Queries and LO Normalized Character Strings 187
Table 32. Infix-Hyphen Normalized Pattern ....................................................198
Table 33. Sub-Word Indexing Involving a Compound Word .........................200
Table 34. Sub-Word Indexing Involving a Non-Standard Word .................201
Table 35. Matched Abbreviated Token “B.C.” in the Usenet Dataset ...............202
Table 36. An Example of LO Normalization Effects .......................................207
Table 37. An Example of LI Normalization Effects .......................................208
Table 38. Examples of Text Segments with Two-Letter-followed by Period ....211
Table 39. Examples of Two Letter Abbreviated Words with Periods .............212
Table 40. Lines Containing Abbreviated Words with Infix Blank Space ........216
Table 41. The Categorization of Punctuation Mark Related Concepts ............234
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.</td>
<td>Theoretical Representation of Punctuation Normalization</td>
<td>24</td>
</tr>
<tr>
<td>Figure 2.</td>
<td>The Google Search Result for the Phrase cat -v</td>
<td>50</td>
</tr>
<tr>
<td>Figure 3.</td>
<td>The Google Search Result for the Phrase cat -v Using the Exact Search Feature</td>
<td>51</td>
</tr>
<tr>
<td>Figure 4.</td>
<td>Demonstration Dataset Creation Process</td>
<td>68</td>
</tr>
<tr>
<td>Figure 5.</td>
<td>Major Methodological Procedures and Corresponding Chapters</td>
<td>74</td>
</tr>
<tr>
<td>Figure 6.</td>
<td>Possibilities of Punctuation Marks in Two Datasets</td>
<td>82</td>
</tr>
<tr>
<td>Figure 7.</td>
<td>An Example Showing Different Types of Tokens</td>
<td>94</td>
</tr>
<tr>
<td>Figure 8.</td>
<td>Text Segment Containing Infix Em Dash</td>
<td>112</td>
</tr>
<tr>
<td>Figure 9.</td>
<td>The Least Frequently Used Infix-Period Tokens in the Usenet Dataset</td>
<td>117</td>
</tr>
<tr>
<td>Figure 10.</td>
<td>Exemplary Text Lines Containing Double Dashes (Usenet)</td>
<td>128</td>
</tr>
<tr>
<td>Figure 11.</td>
<td>A Comparison Chart of Punctuation-Exclusive Tokens</td>
<td>129</td>
</tr>
<tr>
<td>Figure 12.</td>
<td>Low Ranked Punctuation-Exclusive Token in the Gutenberg Dataset</td>
<td>131</td>
</tr>
<tr>
<td>Figure 13.</td>
<td>Example of Punctuation-Exclusive Token in the Gutenberg Dataset</td>
<td>132</td>
</tr>
<tr>
<td>Figure 14.</td>
<td>Variation of Paired-Punctuation Marks Based on Post-Pre Spaces</td>
<td>139</td>
</tr>
<tr>
<td>Figure 15.</td>
<td>An Exemplary Sentence with Punctuation Marks</td>
<td>153</td>
</tr>
<tr>
<td>Figure 16.</td>
<td>LO Normalization and Orthographic Comparison Environment</td>
<td>169</td>
</tr>
<tr>
<td>Figure 17.</td>
<td>Conceptual Configuration of LO Normalization Effects</td>
<td>205</td>
</tr>
<tr>
<td>Figure 18.</td>
<td>Conceptual Punctuation Normalization Effects: A.A. and I.D.</td>
<td>214</td>
</tr>
<tr>
<td>Figure 19.</td>
<td>Text Lines Containing A.A. from the Gutenberg Dataset</td>
<td>217</td>
</tr>
<tr>
<td>Figure 20.</td>
<td>Punctuation Normalization Framework</td>
<td>219</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

Retrieval of textual based information in any type of domain is predominantly dependent upon a standard set of letters in a given language. That is, text retrieval systems that are responsible for analyzing, processing, and delivering the retrieval of text most often use alphanumeric characters. Considering only alphanumeric characters as a means to analyze corpus-based text is a standard practice in developing text retrieval applications. After all, words that are used every day are mostly represented by a standard set of letters and numbers.

Written information has been increasingly depicted by more than just alphabetical characters. Since punctuation marks in English became widely used in the 16th century, punctuation mark usages have been constantly evolving (Parkes, 1993; Hitchings, 2011). Punctuation marks have become a more integral part of our written language and they have been appearing in almost all types of texts in modern times. The spread of punctuation marks might be linked to human evolution as Holden (2008) pointed out. In light of all of these aspects of punctuation marks, the focal point of this research, however, is to determine how punctuation marks can be treated for the purpose of search and retrieval of information.

Let us first establish what punctuation marks are. The English word for punctuation was "pointing" (Nunberg, 1990). In fact, punctuation refers to various systems of these points and all types of glyphs that are found in a language. A glyph is an image that represents a character or part of a character. The word “mark” is usually attached in order to denote a physical, written form of punctuation. In this research, punctuation and punctuation marks are used synonymously. In general, punctuation marks include symbols, special characters, orthographic characters and others.
The use of punctuation marks is language dependent as they vary from one language to another in varying degrees (Santos, 1998). Despite the fact that punctuation marks are essentially similar among Indo-European languages, the focal point of study will be limited to English language texts.

Definitions of punctuation marks can be found in more academic studies. They differ slightly from the conventional understanding of punctuation marks. Nunberg (1990) defined punctuation as a set of non-alphanumeric characters that provide information about structural relations among elements of text. The notion of non-alphanumeric characters is defined later.

Jones (1997) embraced perhaps the broadest notion of punctuation. He argued that punctuation is any facet of the text that is not a letter or number. Similarly, Say and Akman (1997) defined punctuation marks as a standard set of marks and graphical devices in a language. According to their definition, a standard set of marks include the comma, colon, period, dash, etc., while graphical devices include textual features such as paragraphing, listing, and emphasizing with italics.

Incorporating the above-mentioned researchers’ notions of punctuation marks, it is reasonable to consider punctuation marks as any type of glyph that denotes punctuation, including invisible markup character such as spaces between characters. These are language-specific elements often used in conjunction with alphanumeric characters that represent another concept of its own or enhance and clarify written text.

In determining what constitutes a set of punctuation marks, the closest term that matches the punctuation mark is the term *non-alphanumeric character*. As the prefix “non” suggests, ‘non-alphanumeric character’ simply means something other than alphanumeric characters. A *character*, which is usually associated with numeric code in computer processing, refers to any
one symbol. See a tutorial on character code issues (Korpela, 2012a) for an extended description and related terms. To restrict ourselves for operational purposes, we can define alphanumeric characters as the character sets that are associated with numbers “0” to “9” and letters “A” to “Z”. This term is not only useful for understanding the notion of punctuation marks, but also it is useful for deciding what constitutes punctuation marks.

One caveat in understanding the notion of punctuation mark is the issue of *blank spaces*. Blank spaces between words are unique in that they are typically not represented by a graphical mark. Blank spaces disambiguate the meaning of characters. Functionally, blank spaces visually indicate boundaries between certain structural segments of the written language. In simple linguistic sense, they often signify a “pause” between words.

In defining punctuation, Jones (1997) takes a most liberal notion of punctuation mark and includes blank spaces as a punctuation, which he refers to as “inter-whitespaces”. However, whether blank spaces are considered as punctuation or are not is generally debated. Nunberg (1990) does not specifically state that blank space is a punctuation mark. It could be argued that blank spaces should not be considered as just “empty” despite the fact that it appears to be empty.

From a computational perspective, blank spaces correspond to characters in a given character set. For example, there is the American Standard Code for Information Interchange (ASCII) code for spaces, line breaks, etc. (Code for Information Exchange, 1968). More specifically, a blank space will correspond to ASCII decimal #32, which is labeled as a *space* in Appendix A.

Blank spaces play a special function in the text and need an independent treatment in most cases. To operationalize the notion of punctuation mark and to define other related terms based on the notion of punctuation marks, it is inconvenient to include blank spaces. For this
reason, blank spaces will be excluded from the list of punctuation marks and blank spaces will be specifically referenced in our discussion.

The uses of punctuation marks have become increasingly integral parts of text-based information. For example, the word $C++$ is more commonly used form of the word than the word $cplusplus$. In this example, the two plus signs $++$, which we refer to as punctuation marks according to our earlier definition, are used in conjunction with the alphabetical character $C$ to form another meaning. Take another example. The word $café$ contains a diacritic – a glyph added to the letter $e$. Although the word $café$ is French, it is not uncommon to find this word in English text with the presence of a diacritic mark. We may refer to the diacritic as a punctuation mark according to our operational definition.

For another example of punctuation marks, consider the following simple text segment:


This short text segment contains multiple punctuation marks: quotation mark, question mark, colon, period, parentheses, and comma. Strictly speaking, italics here can be considered as punctuation marks according to Jones (1997) and Say & Akman (1997). However, italics are excluded in our definition of punctuation marks for the purpose of this research. The above text-segment contains different types of punctuation marks that serve different purposes. Moreover, because punctuation marks appear in conjunction with alphanumeric characters, words can appear with various punctuation marks in a text. For example, consider the word $computer$. In a given text-based content, many different types of punctuation marks may surround the word $computer$. For example, we may find:

- $computer$-
- $computers$)
This research refers to these types of words to which punctuation marks are attached as a punctuated character string. For this study, we will define a character string as a continuous sequence of characters. Similarly, a punctuated character string is a character string that contains at least one punctuation mark. For instance, the word \texttt{C++} is a punctuated character string. In the above example, the word \textit{computer} can be also regarded as a term in a text retrieval system. Similar to the notion of a “word”, a term in this research is defined as a word that appropriately describes any elements of a subject matter. A lexical item is defined as a word or a series of words representing a single unit of sense or meaning. A blank space is an “empty” like character that can be a part of character string, but its purpose is to demarcate the beginning and end of a word.

Given the fact that such terms are indexed by a text retrieval system, the system’s objective is to match the indexed character strings with a user’s search query expression. Users of a text retrieval system typically use search terms as a means to describe a user’s information need. One of the prerequisites of a text retrieval system is to determine optimal ways to retrieve relevant information based on a user’s search query. With an overarching goal of determining optimal ways to retrieve relevant information, the focus of normalization is to standardize variants that are found in text.

An important term that is used in this study is “normalization”. In general term, according to Merriam-Webster Online Dictionary (Normalization, 2012), normalization means to make conform to; or it can mean to reduce to a norm or standard.. Even within the areas of computer processing, this term has been used widely depending on the types of objects. For
example, text normalization generally means standardizing terms so that terms with the same meaning but with different variations can be effectively retrieved by a user (Baeza-Yates, 2004). If one is referring to the term in the context of normalizing data, the definition of normalization is entirely different. Database normalization refers to the process of efficiently organizing data in a database with the goal of reducing redundancies in a database system (Date, 2004). In this study, only the first definition that deals with the context of text retrieval setting is directly applicable. More specifically, the primary interest of this study is in normalizing text elements that contain punctuation marks.

The Notion of Punctuation Normalization

Let us establish terminology that is associated with normalizing punctuation marks. Punctuation normalization is a new term that is defined for this research. Punctuation normalization is a type of text normalization process that is intended to normalize punctuation marks. This term enables us to focus primarily on the issues regarding punctuation marks in the text retrieval context. To be considered as a punctuation normalization operation, two basic conditions must be met:

1. Text processing only purposefully targets punctuation marks or punctuated string. Text processing, in this case, means performing any type of operations on a given text.

2. Normalization procedure must include operations that explicitly or implicitly remove, add, or substitute punctuation marks.

Typically, punctuation normalization operates prescriptively based on custom developed algorithms or rules. These algorithms or rules target and determine how punctuation marks should be transformed from one form to another. For instance, a punctuation normalization procedure might include an explicit specification to drop the internal hyphenation and convert a character case from one form to another. As a result, the character string All-you-can will be
transformed to *all you can*. In this case, a number of specific arrangements can be devised to drop hyphens and change the capitalized character string. These are as follows:

- Convert the uppercase to lower case for all occurrences and drop the hyphens
- Convert the letter *A* to lower case *a* and drop the hyphens
- Convert all the incidences of the character string *All to all* and drop the hyphens
- Convert the word *All-you-can* to *all you can*
- Convert the character string *All-you-can* to *all you can* only if certain specified conditions are satisfied

The above options are not the only ones that can be used to normalize the character string *All-you-can* since other algorithmic procedures can be imagined. All of the above statements that specify operations to change from one form to another consist of specific rules that affect punctuation marks. Such rules are referred to as punctuation normalization rules in this study. More specifically, all of the punctuation rules above can produce a normalized expression by converting the character string *All-you-can* to *all you can*. Notice that the last option in the above example is slightly different since the normalization will occur only if some types of specified conditions are satisfied (e.g., normalize if the string length greater than 3).

In the above case, any operation that results in removing the hyphen is considered as punctuation normalization. Considering the earlier example of the term *computer*, if punctuation normalization is applied to the list of related terms, then we may end up with just the character string *computer*. In this fashion, every instance of information relevant to the search term and index term *computer* can be effectively retrieved since they match each other. Changing a capitalized character string is not considered as punctuation normalization in this study; it is considered a type of text normalization.
In this study, we use the notion of punctuation normalization in reference to a specific subtype of the text normalization problem. However, it is often inter-mixed with other text normalization procedures. In the above example, changing the capitalized $A$ to lower case $a$ requires a text normalization procedure. Often such a type of text processing that involves converting capitalized character strings might be specified in conjunction with a punctuation normalization procedure. Within the framework of text normalization, often punctuation marks are removed normally for the purpose of reducing unnecessary character string variants. In certain instances, the definition of punctuation normalization can be slightly different from the definition of text normalization. In such cases, the notion of punctuation normalization may be further clarified.

There are cases where punctuation normalization could be less desirable. In certain cases, punctuation marks can be regarded as critical elements that create a special meaning in a character string (e.g., the word $C++$). Because of the varying roles punctuation marks play, a closer investigation is required to consider various forms of character strings. This is especially true for a text retrieval system; these systems are required to process a user’s search query by taking the user’s search terms and matching those to indexed terms in the system. For example, common words such as $C++$ or $AT&T$ may appear in a user’s search query. Without two plus signs $++$ and an ampersand sign $&$, the meaning of these words will be changed, and a greater number of inaccurate, irrelevant search results is likely to be produced.

In the context of punctuation normalization, two important types of punctuation marks discussed in this study are:

- Lexical independent (LI) punctuation marks
- Lexical oriented (LO) punctuation marks
The LI punctuation marks are those that have independent functional roles in a given sentence. Since the LI punctuation marks have specific word boundary, normalizing these punctuation marks do not affect the meaning of the words. Examples of LI punctuation marks include commas, semicolons, full-stop, quotation marks, and others.

On the other hand, LO punctuation marks are punctuation marks that generally have functional roles with individual lexical items in a sentence. The characteristics of LO punctuation marks are that their functional roles are generally localized to two to three words. Most often normalizing LO punctuation marks may alter the meaning of the word associated with the punctuation mark. LO punctuation marks generally include plus sign, dollar sign, hyphen, percent signs, dot, etc.

All punctuation marks may fall into either one of these two types: LI punctuation mark or LO punctuation mark. The category depends on the specific role in a sentence or a lexical item. For this reason, one type of punctuation mark could be categorized LI or LO depending on a specific situation. For example, a comma is usually considered as a LI punctuation mark, but it can be also categorized as a LO punctuation mark (e.g., a comma in 1,000). These two types of punctuation marks are important since they may indicate the type of punctuation normalization in the context of text retrieval. As discussed in Chapter 4, the exact determination of LI or LO punctuation marks require consideration of other factors.

In a text retrieval environment, punctuation normalization can occur mainly in two critical places: on search query side and on system’s content side. An example of punctuation normalization that occurs in the search query is a transformation of the search terms entered by a user. In the content side, where the text is stored and processed, punctuation normalization can occur as a part of text processing. Often, search and retrieval of a punctuated character string
such as C++ would depend on the text retrieval matching function where it attempts to match
the search terms with the index term, assuming an index was created by a system. In this case,
the search terms and the index terms will be normalized to varying degrees by taking the search
and retrieving functions into consideration. Differences in normalization rules may lead to less
than optimal retrieval.

From the standpoint of improving text retrieval, a number of procedures that are often
required in the course of designing a text retrieval system must be carefully considered. One of
the main procedures is transformation from one string to another for the indexing and retrieving
relevant to the user’s information need. If the user is not aware of the details of such
transformations, the user will not understand why certain results occur.

Essentially, depending on each type of search scenario, punctuation normalization
choices may lead to better or worse search results. How one should decide to treat punctuation
marks also depend on other factor such as the type of search queries (e.g., phrase search) and the
type of matching function (e.g., exact match) that a designer of a text retrieval system must also
consider.

Consider a situation where a user wishes to find product information based on a product
number. The user decides to search the document containing the character string x.22 to find
relevant information. However, the user discovers that many variations were retrieved and
becomes perplexed by how the retrieval system behaves in retrieving such punctuation marks.
For example, a text retrieval system produces search results that contain the following terms:

- x 22
- x-22
- x22
- $x/22$
- $17 \times 22$
- $x.22.52$

Similarly, a user might want to retrieve sentences that only contain the character string $X:Y$ algorithm. That is, the user might desire to match with the character string as the search query appears without retrieving following string variants:

- $XY$ algorithm
- $X&Y$ algorithm
- $(X,Y)$ algorithm
- $X - Y$ algorithm
- “$X Y$ algorithm”
- $X,Y,algorithm$

As can be seen from the above examples, one of the main issues is that all of these procedures and decisions are not apparent. The effects are not well understood from both designer’s and user’s perspectives. From a user’s perspective, a matching process that is more transparent and easy to understand might be important. From a designer’s perspective, attempts to satisfy information needs of such a user leads to different types of punctuation normalization choices, and each choice are linked to different effects on text retrieval.

This type of problem, however, has not been systematically approached. Various issues, or factors, as mentioned above are linked together and present a complex issue and set of decisions that may affect the outcome of text retrieval processes. At present time, we know very little about the type of punctuation normalization that is desirable and the effects of punctuation normalization on search and retrieval.
Research Motivation

From a text retrieval perspective, a research opportunity was clearly noticeable. Text retrievability issues that are closely associated with punctuation marks have not been closely examined in the past. While there has been an overwhelming amount of research that focused on various text retrieval techniques, including normalization, stemming, etc., most text retrieval related research only focus on alphanumeric character strings.

Traditionally text-based retrieval systems have viewed punctuation as extraneous since it can interfere with identifying the words that stylistically appear differently but have the same or similar meaning. It is true that in certain situations punctuation marks can be viewed as “noise” that should be eliminated. Although punctuation marks can play a critical role in disambiguating words in a text, from a text retrieval viewpoint, punctuation marks have been generally regarded simply as “extraneous” characters.

Based on this view, Brooks (1998) concluded that inconsistencies in treating and processing queries with punctuation marks seemed unavoidable in the realm of IR systems. Many types of successful text retrieval systems are in existence today, but some users of text retrieval systems may already have noticed that such systems sometimes inadequately process search queries that contain punctuation marks.

One of the problems with normalizing text is that often modifying subtle cues can change the original sense or the intended meaning of the words. While reducing text variants by removing punctuation marks in general is necessary as it often increases retrieval performance, this is not true for all cases due to the side effects of punctuation normalization. Even though a wide range of subtle problems are associated with punctuation marks, punctuation normalization remains a neglected problem.
Meanwhile, other interesting approaches have utilized punctuation marks in text retrieval related applications. Such approaches include works by Mikheev (2000); Sproat et al. (2001); and Kaufmann and Kalita (2010). One of the benefits of punctuation marks is that they can provide additional cues that are hidden in a text. In essence, many small elements within any given text can carry subtle cues to distinguish one type of meaning from another.

Despite this benefit, attempts to explore punctuation marks have been limited from a text retrieval perspective. To increase search and retrieval effectiveness, a sufficient amount of analysis on punctuation marks should be undertaken. As will be shown in the next chapter, punctuation marks can aid in reducing ambiguity as textual elements. Yet, without an empirical investigation, it is difficult to determine how punctuation marks or their normalization can improve search and retrieval effectiveness and what normalization rules to use with existing text retrieval theories.

In a nutshell, prior research has not sufficiently focused specifically on punctuation mark issues and punctuation mark normalization. Brooks (1998), in particular, included some punctuated character strings in his analysis. However, Brooks’ analysis only pointed out some of the complicated problems that are associated with punctuation marks. In addition, his analysis did not consider various relevant factors and did not adequately explore the punctuation normalization problem.

The primary motivation of this study is to determine whether there is a methodologically justifiable way of normalizing punctuation marks for the purpose of improving text retrieval systems. As of now, only fragmentary methods exist specifically for the text retrieval environment. One of the desirable features that seemed to be missing was a sound methodological approach to analyzing punctuation marks. Basically a sound approach should be
able to distinguish and utilize punctuation marks and punctuation patterns; thereby, it should be helpful in assessing the importance of punctuation marks, and punctuation patterns in terms of punctuation normalization.

For the purpose of examining punctuation marks, defining a notion of *punctuation pattern* can be useful. A punctuation pattern in this study is a term used to indicate any type of recognizable pattern formation that involves at least one punctuation mark. This term can be used rather loosely to include broad types of punctuated patterns, although a specific type of punctuation pattern was named and identified in this study. From a theoretical perspective, determining some sense of “meaningfulness” of punctuation patterns in a text retrieval context is the key issue that has not been sufficiently addressed in previous studies.

In general, this research can be an important step toward understanding the complexity of the normalization process. This research was conducted to provide insights into understanding punctuation marks and punctuation normalization. In particular, this study intended to examine the role of punctuation marks using real-world datasets so various factors that influence punctuation normalization can be examined.

In a larger sense, punctuation mark issues that are relevant to text retrieval are explored, and this will help to understand punctuation normalization related problems. This study mostly deals with investigating and analyzing normalization implications of punctuation marks from the perspective of text retrieval. Toward the end of this study, in Chapter 5, we focused on the complexity of punctuation normalization. We have provided what seems to be the most useful information to understand the effects of punctuation normalization. Some of the findings of this research should be applicable beyond the punctuation normalization problem by shedding light on text normalization and other related aspects of the text retrieval system as a whole.
Theoretical Framework

Theories developed in related disciplines provide the theoretical basis for this study, which examined punctuation normalization issues in the context of text search and retrieval. To develop a theoretical framework for this research, some essential components of existing relevant perspectives from different fields of study were incorporated. The role of punctuation marks and the application of punctuation marks span three related theoretical disciplines:

- Linguistics (computational)
- Information retrieval (text retrieval)
- Information extraction (IE)

The Linguistic (Computational) Perspective

The uses of punctuation marks has been examined from the discipline of linguistics, especially in Say and Akman (1997, p.1). The original purpose of punctuation marks in a written language was to provide additional meaning to minimize ambiguity of a unit of text (e.g., words, phrases, paragraphs, etc.). To find linguistic patterns of punctuation marks, some computational linguistic approaches have been applied.

In a broader view, the problem of punctuation marks also touches upon the category of orthography. Orthography in general is a type of language study that is concerned with letters and spelling, representation for word boundaries, stops and pauses in speech, and tonal inflections. Moreover, orthography is “the set of rules for using a script in a particular language (i.e., the English orthography for the Roman alphabet) like symbol-sound correspondences, capitalization, hyphenation, punctuation, and so on.” (Cheng, n.d., p.2).

A particular noteworthy aspect of linguistic studies is that punctuation marks indicate the structure and organization of written language. Some linguistic studies suggested that
punctuation marks play diverse roles in a language. Many interesting aspects of punctuation marks were discovered by scholars such as Nunberg (1990), Doran (1998), and Briscoe (1994). Nunberg (1990), in particular, conducted an extensive syntactical analysis based on “separating” punctuation marks and delimiting punctuation. Delimiting punctuation marks (e.g., parenthesis) are used in pairs, while separating punctuation marks (e.g., comma, semicolon) are used to separate different segments in a sentence. The efforts by previous researchers focused mostly on discovering syntactic roles of punctuation marks.

In particular, computational linguistic theories have discovered some important aspects related to punctuation marks. Jones (1997), in particular, investigated a computer-based approach to examining the roles of punctuation marks. The significance of his work is that he used real world corpora to examine punctuation marks. He carried out practical experiments based on some of the previous linguistic theories and demonstrated that the roles of punctuation marks are much more diverse than previously thought.

Individual roles of punctuation marks can be differentiated depending on the syntactic roles in a language as Jones (1997) pointed out. For example, a period can be used with an abbreviated character string or as an end of a sentence marker (full-stop). In Appendix A, it is equivalent to ASCII code decimal # 46. Determining the roles of individual punctuation marks and punctuation patterns are crucial in understanding different type of punctuation normalization effects. From this study’s point of view, this is the contributing component of the computational linguistic perspective.
A research discipline with which this study directly associates is information retrieval (IR). The central issue of IR is locating information within computerized information systems. Manning et al. (2008) defined IR as finding material of an unstructured nature that satisfies an information need from within large collections stored on computers. Jones and Willett (1997) defined IR as a subject field covering both the representation and the retrieval sides of information. Jansen and Rieh (2010) provided a similar definition. They defined IR as a process of representing, storing, and finding information objects (e.g., multimedia, documents, etc.) with information technology.

Since this research focuses on text retrieval, which is a branch of IR, some aspects of text retrieval are identical to IR. For example, for IR systems, the issues of indexing, ranking, similarity is common regardless of the type of information objects. The major difference between the two notions is that text retrieval focuses on issues related to retrieving only text-based contents, while IR deals with issues related to retrieving information beyond just text-based contents (e.g., videos, images, sound recordings, etc.). By focusing on just text retrieval issues, some important aspects of IR are incorporated in developing a theoretical framework for this study.

Key terminology associated with information retrieval need to be clarified first. A text retrieval system is a computer based information tool that supports a specific user's text-based information needs. Text retrieval systems can be associated with a wide range of information related sources such as online, automated, digital, and bibliographic data. The term retrieval will be used in the context of extracting information out of text-based content. Thus, this term is used to denote retrieval system functions. From a user’s standpoint, a typical user who needs
information issues a *search* using a search query. As opposed to denoting a system’s function, the term *search query* is used to denote a formulated query that represents a user's information need.

Central to the notion of text-retrieval is text processing. Text processing includes various text related operations such as lexical analysis, elimination of stop words, index terms selection, thesauri construction, and text normalization (Baeza-Yates & Ribeiro-Neto, 2011). Discussions of these topics are surveyed in the literature review. The specific text processing operations depend on the functionality of the underlying IR system and the underlying IR model that the system supports.

From a text retrieval perspective, text normalization is just one type of text processing. Regardless of the type of text normalization, typical text processing techniques include procedures to identify and decompose text into different types of segments: word, sentence, paragraph, etc. As pointed out earlier, since punctuation normalization is a type of text normalization, punctuation normalization can be viewed as a part of overall text processing operations.

For the most part, language is an inseparable component of searching and retrieving information. From the search side, the user's information needs must be formulated into a query that is essentially language based. At the same time, on the text processing side, underlying text must be cleaned, normalized, and indexed for retrieval. From a system’s point of view, truly understanding the user’s search request is difficult because of the complexity of human language. While language provides users with an enormous ability to specify information needs, most often systems have difficulties in understanding human languages.
In the early days of IR research, the fundamental language related problem that is associated with estimating a user's information need for content in a document was poignantly described by Blair and Maron (1990). They argued that since information can be described in so many different ways using a natural language, it is difficult for a system to figure out correct terms to retrieve the relevant documents. According to these authors, the number of different ways in which an author could write about a particular subject was unlimited and unpredictable. As an exemplary case, they pointed out that an event such as an “accident” from one person’s point of view could be described as an “unexpected event,” an “unfortunate occurrence,” an “untimely incident,” etc.

Another fundamental IR problem that is related to language is pointed out by Brooks (1998). He contended that orthography is a major impediment to online text retrieval systems. His point was that orthography, which includes punctuation marks, is a fundamental issue for text retrieval systems. He argued that text retrieval related research cannot make significant progress without solving some of the orthography issues. In effect, he demonstrated that normalizing orthographic elements can have noticeable effects on IR systems.

The problem of retrieving textual elements that are related to punctuation marks in some ways seems to be more interesting than Brooks’ pointed out. The sheer variety of punctuation marks requires different types of normalization. As noted earlier, the period at the end of a sentence, can be generally removed for a text retrieval purpose. Let us refer a period as a *full-stop*, if it is used to mark an end of a sentence. Otherwise, it is referred to as just a *period*, although it is officially labeled as a full-stop (ASCII code decimal #46) according to ASCII code in Appendix A. Each punctuation mark typically requires specific normalization rules. At the same time, punctuation marks occasionally contain important meaning that should not be simply
disregarded. From IR theorists’ point of view, assessing the types of “meaningfulness” of punctuation marks in respect to punctuation normalization can be very useful.

Information Extraction (IE)

Another influential area of study that is highly pertinent to punctuation normalization is IE. IE can be considered as a subfield of IR which focuses on extracting useful information such as entities, relationship between entities, and attributes describing entities. IE complements IR systems, and the area of IE has emerged as a research area that closely influenced the field of IR (Moens, 2006).

Using an IE technique to determine relevant information can be particularly beneficial. IE techniques are often employed where applications need to find correct information based on relationships among text elements. IE can be applied to information processing related tasks such as indexing, parsing, sentence boundary disambiguation, extraction of named entities (e.g., people, places, and organizations), and identification of relationships between named entities (Sarawagi, 2008). IE techniques rely on analyzing text in an attempt to exploit underlying semantic clues.

For example, an abbreviated string B.C., in which each letter is separated by a period, could stand for entirely different notions such as British Columbia, Before Christ, and Boston College. By examining various elements in the text, an IE algorithm might be used to correctly identify whether the abbreviation is semantically equivalent to one of the following:

- British Columbia
- Before Christ
- Boston College
On the other hand, in a text retrieval environment, if the term *B.C.* is used in a search query, an ordinary text retrieval system may retrieve only strings that are orthographically similar. Unlike the above case, in this case a text retrieval system might attempt to match strings in a search query against any of the following character strings:

- B.C.
- B.C.
- BC
- -B.C.-

What sets the above task apart from a typical IR system is the following: An IR system may only retrieve different patterns of the term *B.C.* and relies on the user to make an ultimate decision, while an IE based application or component might be able to employ additional procedures to correctly identify relevant terms. The problem of identifying information units like people, organization, location names, and numeric expressions are known as Named Entity Recognition (Grishman & Sunheim, 1996; Nadeau & Sekine, 2007).

As pointed out in this example, the difference between IR and IE is that IR focuses on retrieving a subset of documents in a corpus while IE focuses on identifying particular features of information within a document. IE often uses predefined, extracted type of information and represents specific instances found in the text. Cowie and Wilks (2000) stated that IE is more interested in the structure of the texts, whereas IR views texts as just bags of words.

The techniques and success of IR and IE applications vary depending on the type of content. For example, Vlahovic (2011) pointed out that if the content is natural language, the success of IE is limited due to requiring exhaustive pattern-matching algorithms. He further pointed out that IE from unstructured content does not guarantee accuracy. Despite these
differences, IR and IE techniques are complementary to each other and should be combined to build powerful text processing tools as suggested by Gaizauskas and Robertson (1997).

The separation of the two areas becomes less distinct if text processing techniques in each area are combined. In some ways, it may be desirable to incorporate IE techniques to an IR system. Such an experiment was carried out by Gaizauskas and Robertson (1997). Sarawagi (2008) pointed out that in addition to classical rules-based methods, statistical methods and hybrid-methods have been developed. Assuming that some type of coupling of IE and IR is desirable, it is necessary to examine IE related procedures that process or utilize punctuation marks.

In case of utilizing punctuation marks, some previous research provided practical methods to reduce ambiguity of textual elements. Sometimes punctuation marks have been vital elements that IE uses to disambiguate text. For example, Mikheeeve (2000) attempted to determine whether a period is used to indicate a sentence or an abbreviated character string. For character strings that one wants to disambiguate, IE can further aid in determining the specific appropriate technique that can be applied.

Integrated Theoretical Framework

To provide a theoretical basis for conducting this research, various concepts surrounding punctuation marks from the relevant areas (linguistic, IR, and IE) are used in this study. These research areas not only provide operational means to normalize punctuation marks but also provide opportunities to utilize the knowledge that comes along with punctuation marks. In particular, a multi-disciplinary view can be especially useful in the process of finding implications of punctuation marks and punctuation normalization effects on text retrieval.
However, it is difficult to draw clear boundaries as there is an overlapping area among different theories. In spite of distinguishable concerns of each research area, there is an area of interaction among different concepts and theoretical disciplines. Because of this reason, in building this study’s own theory, multiple perspectives from different fields of study are taken into account. By integrating the most relevant perspectives and methods into the study’s research framework, this study assumes that the necessary details can be established in observing the punctuation normalization phenomena.

Considering the different aspects of punctuation marks will be used to gain insights into the punctuation normalization problem. An integrated view that can be derived from the above mentioned perspectives (linguistic, IR, and IE) can aid in understanding the complexity of normalizing punctuation marks. The effects of normalizing punctuations can be simple in some cases but not obvious in other cases; one cannot often comprehend all the competing factors at a glance.

In addition, collectively considering IR, IE, and computation linguistics is important to answer the following overarching question: how and to what extent are punctuation marks essential in terms of text retrieval? In assessing the degree of importance, various perspectives from IR, IE, and computational linguistics need to be considered. After all, the ultimate goal of the research is to contribute knowledge about punctuation marks for the purpose of increasing search and retrieval effectiveness. Combining the best elements from this group of theories enables us to reach this goal.

In this research, an attempt is made to develop a unique theoretical framework based on the relevant previous works. The relevant theories that can contribute to building a theoretic framework are summarized as the following:
1. Linguistic theories, especially from the computational side, can offer some punctuation mark characterization, and thereby, it can offer an ability to categorize punctuation marks and punctuation patterns.

2. IR theories address issues in dealing with relevance with the goal of satisfying user’s information needs. That is, it often addresses factors and effects of punctuation normalization from the point of view of delivering the pertinent information to a user.

3. IE can bring an expertise in systematically analyzing punctuation marks for the purpose of distinguishing and identifying elements that essentially have the same meaning. Also, IE offers methods to analyze punctuation marks and relevant textual element characteristics (e.g., capitalization) surrounding each punctuation mark.

The purpose of the theoretical framework is also to draw the boundaries of this research. A typical theoretical framework provides a schematic description of relationships among different components. Figure 1 is a conceptual representation of related components that guided this research and maybe able to guide punctuation normalization decisions. From this diagram, the inter-related components that can potentially influence punctuation normalization decisions can be identified.

![Figure 1. Theoretical representation of punctuation normalization.](image)
They are:

- Categorization of punctuation marks
- Role of punctuation marks
- Content type
- Segmentation
- Indexing scheme
- Text processing
- Search query expression
- Search feature

Some of these components can be identified as factors that may have effects on punctuation normalization and will be covered in the subsequent chapter. As techniques vary widely, some necessary assumptions were made during the course of this research. In any case, under the overall framework, punctuation marks can be best traced back to the theories of IR, IE, and linguistics. In Figure 1, the tokenization and indexing schemes are the typical procedures that are found in IR applications. The procedures are indicated on the left side of the figure as they are in the realm of investigating punctuation normalization issues. Based on these core components, which have relationships to punctuation marks, the theoretical assumption in this study can be summed up as the following: *In a realm of a text retrieval environment, a punctuation-centric approach is needed to systematically analyze issues related to normalizing punctuation marks.*

To carry out this study, identifying common patterns of specific punctuation uses in the real-world datasets will be examined. The real-world datasets that will be used in this study contain a variety of punctuation marks. By comparing extracted patterns from the datasets, realistic differences and commonalities among datasets can be obtained. We will also identify
different types of punctuation marks in respect to punctuation normalization problems. This can allow future research to analyze punctuation marks more effectively based on the findings of this research. To this end, this study can be viewed as an initial exploratory and descriptive attempt to improve existing handling of punctuation marks in text retrieval systems.

The framework of typical IR systems as shown in Figure 1 is dynamic. Each type of IR related procedures can affect the search results in different ways. It is difficult to come up with causal relationship among these components that can be used to predict the effects of punctuation normalization. As a new area of research, many intricate causes and issues have to be discovered first. A careful analysis of how extracted items using IE techniques can be integrated into a general retrieval setting is needed. Occasionally extracted textual elements using IE techniques can be orthographically different from term matched using information retrieval. This study is an exploratory study that focuses on discovering such new and challenging issues.

The components in Figure 1 can be considered as factors that one has to consider in analyzing punctuation normalization effects. The analysis will likely reveal options that a system designer should take into account in choosing one approach to normalization versus another. For this reason, a careful analysis of effects of punctuation normalization along with punctuation patterns is necessary.

Research Questions

Considering the several intriguing aspects of punctuation marks, a key question that is associated with text retrieval systems is the following: *What are the implications of normalizing punctuation marks from the perspective of text retrieval?* Since this is a broad question, a set of more specific research questions guided the inquiry in pursuit of answering the main questions:
1. In what ways and to what extent are punctuation marks and punctuated character strings (i.e., character strings that contain punctuation marks) important in terms of text retrieval?

2. How does punctuation normalization affect the process of matching punctuated character strings?

3. What are some of the useful categories of punctuation marks, punctuation patterns, and punctuation normalization?

4. How can we measure similarity between terms in an orthographic sense so that it can be incorporated into normalizing punctuation marks?

5. In what ways can we use underlying punctuation patterns in a text for a text retrieval system?

These questions, not in specific order, are considered from the perspective of text retrieval.

Research Design and Data Collection

To carry out this research, texts from the following corpora were used:

- Gutenberg Project archive (http://www.gutenberg.com)
- Usenet Newsgroup archive (Shaoul & Westbury, 2011)

These corpora were readily available to download from the Internet. The Gutenberg Project archive contained digitized contents mostly from books representing a wide range of topics such as literature, cookbooks, reference works and issues of periodicals. The second dataset came from a Usenet Newsgroup archive representing posting for 2009. The Usenet Newsgroup posting archives represented a vast amount of modern linguistic expressions and vocabularies that people commonly use in their daily lives.

By examining punctuation marks in punctuated character strings in the datasets various roles of punctuation marks were methodologically analyzed, and some of the punctuation patterns were identified. Based on the analysis, different categories were developed further. Then, issues related to punctuation normalization were explored. Toward the end of this study,
we focused on finding the effects related to punctuation normalization using the datasets that we sampled from the above two corpora.

Limitation of the Study

As in the case of most research, there are some practical constraints and limitations in carrying out this research. The primary limitation of this study is that it is infeasible to exhaustively analyze every punctuation mark and patterns that result from punctuation marks. To this end, we seek to explore all types of issues surrounding normalizing punctuation marks but restrict to examine only punctuation marks and patterns appeared to have the potential for major implication on text retrieval. Also, because of extensive computational effort and time resources that are required, only two datasets were used to base this research.

Another limitation of this study is that it focused only on punctuation marks that are found in the English language. Since the uses of the punctuation marks will vary depending on the language, it is difficult to generalize these patterns to other languages without knowing some details of the language itself. Although the roles of some punctuation marks are essentially same among some languages, without an in-depth analysis it will be difficult to generalize beyond the punctuation normalization analysis based on language in most cases.

Assessing the degree of importance in regards to punctuation marks can be prone to subjectivity. Without conducting an actual user-focused study, assessments will be confined to examining existing corpus-based datasets. In interpreting punctuation mark roles, intuitive, logical explanations are sought. This can be done partly based on existing literature related to punctuation marks.

In assessing the value of punctuation marks from a text retrieval perspective, the most
plausible explanations are sought to minimize subjectivity. Consequently, in certain situations, absolute judgments and decisions, especially in categorizing punctuation mark related notions, have been avoided, as value judgments are basically inexact by nature. During the course of the analysis, additional assumptions had to be stated so that targeted demonstrations could be interpreted appropriately within the context of this study.

Organization

The organization of this research is outlined in this section. Chapter 2 presents background information and a literature review of relevant normalization related approaches in the context of text retrieval. A literature review of related works on punctuation normalization is presented. Chapter 3 provides methodological details and research design considerations. This chapter includes the methodological approaches, major procedures, issues related to designing demonstrations, and limitations of this study. Chapter 4 presents an analysis of various punctuation marks and punctuation patterns. Punctuation mark related statistics are qualitatively interpreted, and they are presented in the context of text retrieval. Chapter 5 provides the results of the normalization strategy analysis. The categorization of punctuation normalization and the effects of punctuation normalization are the focal points of Chapter 5. Chapter 6 provides the concluding remarks of this research. This chapter also re-examines the research questions to see if they are adequately answered. Future areas of research are suggested in this chapter.
CHAPTER 2
BACKGROUND AND LITERATURE REVIEW

This chapter provides additional background information and a literature review that are relevant to this study. Previous research provided various detailed information that was useful for this study. Different approaches emphasized different aspects of punctuation marks, particularly related to text processing. This literature review provided punctuation normalization related concepts. Elements from this literature review were useful in formulating a necessary strategy to conduct this study. The background concepts and literature review for this research are presented together based on these major categories:

- Linguistic and computational analysis of punctuation marks
- Punctuation marks in text processing related procedures
- Concepts related to punctuation normalization
- Punctuation marks in forms of words and expressions

Linguistic and Computational Analysis of Punctuation Marks

Some of the research in areas related to linguistics focus on analyzing syntactic and semantic roles of punctuation marks. Meyer (1983) was one of the pioneers in investigating different aspects of punctuation marks. According to Meyer, syntax and punctuation are related in a hierarchical manner. That is, the period, the question mark, and the exclamation mark are at the highest level in the hierarchy, while the comma is at the lowest level. According to Meyer, by placing appropriate punctuation marks, different text segments (e.g., clause) form a hierarchy within in a sentence. This, in turn, provides different levels of sentence structure. He argued that the rules of punctuation marks are the principles that specify appropriate punctuation mark usage in a particular text.
Nunberg (1990) identified interesting aspects of punctuation marks from a linguistic perspective by focusing on the syntactic rules of punctuation marks. Nunberg argued that punctuation systems are highly systematic and linguistic, obeying a set of linguistic rules. He argued that punctuation marks can be examined independently from lexical grammar. Nunberg referred to commas, dashes, semicolons, colons and periods as points. According to Nunberg’s analysis, when two points occur in a sentence one is absorbed by the other, according to the strength of the two points. In sum, Nunberg focused on constraints on the grammatical sequences of punctuation marks.

Then, a number of other researchers followed Nunberg’s theory and examined punctuation marks from the field of linguistics. Doran (1998) focused on the structural role of punctuation by examining adjunct and subordinating clause constructions in sentences. Dale (1992) identified discourse structure primarily in the context of natural language generation. He pointed out certain punctuation marks such as the comma, the colon, the semicolon, the dash, and parentheses can be interpreted as signals of discourse structure. Osborne (1996) demonstrated that punctuation marks play an important role for machines to learn grammar automatically. Briscoe and Caroll (1997) applied Nunberg’s method to develop a method to analyze parts-of-speech from a text-based corpus.

Linguistic theories provided useful information especially in understanding the nature of punctuation marks. The applicability of the linguistic perspective, however, had to be limited due to its inherent complexity in dealing with human language. Also another limitation was that only a handful of punctuation marks were examined in the context of linguistics. Considering this, general aspects of punctuation marks, including useful categorization schemes provided by the previous research, are subsequently described in detail.
A number of researchers have attempted to categorize punctuation marks. This categorization was constructed from the perspectives of linguistics rather than text retrieval. Here, we mention some of the notable punctuation mark categories.

Jones (1997) performed a computational analysis of the punctuation marks in various corpora. He compared different types of corpora based on the frequency count of punctuation marks. Furthermore, he extracted punctuation patterns from a corpus to generalize the functions of punctuation marks. For our purposes, whether the function of punctuation marks is at a lexical level or a sentential level, we refer to the function of punctuation marks as a “role” and use the terms synonymously. Nevertheless, although his main application was on parsing sentences, Jones’ general characterization of punctuation marks can be particularly useful in terms of understanding characteristics of punctuation marks that have syntactical roles in a sentence.

Jones’ (1997) categorization of punctuation marks is comprehensive and especially applicable in an initial attempt to understand the punctuation mark related issues. Jones categorized punctuation marks into different types mainly on a computational linguistic basis. Jones provided three basic categories:

- Sub-lexical
- Inter-lexical
- Super-lexical

According to the Jones’ research, sub-lexical punctuation marks are marks that could change the meaning of the words. For example, the punctuation marks in character strings such as X-Ray and she’s are considered sub-lexical. The sub-lexical punctuation mark can be source-specific or source-independent. However, whether a punctuation mark is source-specific or
source-independent, it can be less distinguishable sometimes. The category of sub-lexical punctuation marks appears within lexical and numerical entities. These include periods (dots), hyphens, apostrophes, decimal points, multiplication signs, and power signs, etc.

Jones’ (1997) inter-lexical punctuation marks are the typical marks that visually can occur between words and not within words. These typically include punctuation marks such as the colon, comma, and full-stop. He noted that most conventional punctuation marks fall into this category. To this extent, investigating inter-lexical punctuation marks was the primary focus of Jones’ study. Table 1 is a summary of Jones’ inter-lexical punctuation mark categories. An example of the inter-lexical punctuation mark is the question mark in the sentence.

Table 1

*Jones’ Inter-Lexical Categories of Punctuation Marks (1997)*

<table>
<thead>
<tr>
<th>Inter-Lexical Types</th>
<th>Syntactical Functions (Roles)</th>
<th>Punctuation Marks</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source-specific</td>
<td>● decimal point,</td>
<td>• multiplication sign,</td>
<td>Some of these inter-lexical punctuation marks are also considered as sub-lexical.</td>
</tr>
<tr>
<td></td>
<td>● power,</td>
<td>• etc.</td>
<td></td>
</tr>
<tr>
<td>Stress markers</td>
<td>● exclamation mark,</td>
<td>• question mark,</td>
<td></td>
</tr>
<tr>
<td>Conjunctive (Separating)</td>
<td>● semicolon,</td>
<td>• dash, and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● comma,</td>
<td>• comma,</td>
<td>According to Jones, capitals (e.g., capitalized letters) are considered as a punctuation mark because of its pairing role with another punctuation mark.</td>
</tr>
<tr>
<td>Source-independent</td>
<td>● dash,</td>
<td>• brackets,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● colon,</td>
<td>• quotes,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● full-stop,</td>
<td>• capitals,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● etc.</td>
<td>• etc.</td>
<td></td>
</tr>
</tbody>
</table>
Jones further categorized inter-lexical punctuation marks as the following two types:

- Source-specific
- Source-independent

According to Jones (1997), if the punctuation mark usage is a generally accepted rule in the English language, then it is considered as source-independent. The source-independent types are elements such as stress markers, conjunctive, and adjunctive marks. While incorporating Jones’ framework for parsing a text, Doran (1998) acknowledged that Jones’ categorization of source-independent types closely resembled Nunberg’s category, which dealt with delimiting and separating punctuation marks. Doran pointed out that Nunberg’s work mostly dealt with source-independent punctuation mark related issues.

Jones (1997) pointed out that delimiting punctuation marks refer to instances in which certain text fragments are conceptually separated from the rest of the sentence. Jones’ argued that delimiting punctuation marks appear in pairs but not necessary same exact ones. For example, consider the following three sentences:

1. *John has been misinformed, not ignorant.*
2. *Obviously, he slept; she continued working.*
3. *They had a plan—leave the chaotic city as soon as possible.*

In the first sentence, the comma is paired with the ending period. In the second sentence, the comma is paired with the semicolon. Also, the semicolon is paired with ending period. In the third sentences, the dash is paired with the ending period. Thus, delimiting punctuation marks are ones that are used as a pair in a sentence. According to Jones, even the capitalization of first sentence is paired with the period at the end of a sentence, although we have not included capitalization as punctuation marks in this research.
Separating punctuation marks are usually ones that are used to list items. For example, consider the following two sentences:

- *I want to buy apples, oranges, and some vegetables.*
- *I have been to China, Brazil, and Russia.*

In the above cases, the commas are the separating punctuation marks since they are merely used to list the items in a sentence.

According to Jones (1997), if the punctuation mark usage is more dependent upon its content source to figure out its specific punctuation mark role then it is considered as source-specific. Jones’ super-lexical punctuation marks include various structural related mark-up punctuations such as paragraphing, underlining, italicizing, and superscripting.

While both Nunberg (1990) and Jones (1997) provided rich linguistic accounts of punctuation marks, one of the noticeable problems that are associated with generalizing the punctuation marks is that some punctuation marks can appear in more than one category. For example, the ASCII code decimal #46 can be a *full-stop* or a *dot* according to Jones’ punctuation mark category. The limitation of Jones’ punctuation mark category is that it is based on the syntactic and semantic functions of punctuation marks from a linguistic perspective. Jones’ category scheme is neither entirely applicable nor efficient in terms of addressing the needs of text retrieval.

**Frequency Analysis of Punctuation Marks and Punctuation Pattern**

There have been a small number of previous attempts to interpret punctuation marks based on the punctuation mark frequency counts. Although the frequency counts of punctuations marks were performed mainly to discover linguistic aspects of the punctuation marks, they
provide useful information in terms of understanding different roles of punctuation marks. Some of the key aspects of punctuation mark frequency analysis are summarized here.

Meyer (1983) reported the frequency counts of punctuation marks from the Brown corpus (Kucera & Francis, 1967). The Brown corpus consists of more than 1 million words, but in Meyer’s study, only 72,000 words were used as a sample, which comprised approximately 4,000 sentences. Based in his sample, Meyer reported the following:

- Comma (47%)
- Period (45%)
- Dashes (2%)
- Parenthesis (2%)
- Semicolons (2%)
- Question marks (1%)
- Colon (1%)

The percentage of inside the parentheses is a relative percentage of punctuation marks found in Meyer’s sample data collection. The sum of the above punctuation marks percentage equals 100%. According to Meyer, the most common punctuation marks are the comma and the period.

Tavecchio (2010) compared punctuation marks (colons, semicolons, and dashes) in Dutch to English language. His findings were based on four different genres: academic prose, newspaper articles, short stories, and leaflets. Tavecchio also reported findings in terms of the percent that each sentence contained a particular punctuation mark. Pertaining to English, his findings are summarized as follows:

- Use of colons (51.4%) was highest in leaflets
- Use of semicolons (46.4%) was highest in short stories
• Use of dashes (69.6%) was highest in newspaper articles
• Use of semicolon (9.5%) was lowest in the leaflets

From the highest uses to lowest uses, the overall ranked order was dashes, colons, and semicolons.

As mentioned earlier, Jones (1997) extensively examined punctuation mark frequencies among corpora. He reported some of the notable characteristics of punctuation marks based on a number of corpora including Philosophy, Project Gutenberg, and Usenet. His stated that the corpora he examined contain a vast amount of punctuation marks as shown in (1).

\[ . ? ! , ; - ( ) ] < > \{ \} ' \# \* ^ \^ \| / - & \% \$ + = @ ~ \] (1)

Some of his analyses of punctuation marks are summarized as the following:

• The punctuation mark frequency distribution seem to spread considerably among the corpora.
• Some of the stylistic differences regarding punctuation marks were noticeable. For example, one of his corpus used the single quote more extensively than the double quote.
• The most popular punctuation marks were the period and the comma.
• The opening and closing single quotation marks were not evenly balanced.
• Underscore and asterisk appeared more in Usenet than in Project Gutenberg.
• Some corpora contained more than one period per sentence.

In addition, from the corpora Jones also attempted to count sentence based punctuation patterns based on different structural segments. Jones also counted punctuation pattern in a sentences by substituting series of words with the letter e. Punctuation marks inside each word were disregarded. According to Jones (1997), the top 10 punctuation patterns that appeared in Project Gutenberg were the following:

1. e.
2. e.e.
3. e,e,e.
4. e,e,e,e.
5. e
6. e?
7. e,e,e,e,e.
8. e,e.
9. “e,”e.
10. “e?”

The fifth pattern, the letter e without any surrounding punctuation marks, represents “unstopped” sentences. This means that the series of words were not delimited by a punctuation mark. As Jones pointed out, the frequency count of “unstopped” sentences revealed problems that were associated with his frequency counting method.

Regardless, Jones’ calculation of punctuation patterns seems to be a reasonable way to indicate underlying structure of sentences in corpora. In the context of text retrieval, a more pertinent question is how such punctuation patterns help to reduce ambiguity that sometimes comes along with particular types of character strings (e.g., abbreviation). Although only sentence-based punctuation patterns were investigated, his research was the only research that extensively reported punctuation mark related statistics based on corpora.

A less extensive yet still interesting frequency count of punctuation marks was reported by Ling and Baron (2007). They examined linguistic characteristics in text messaging and instant messaging based on a survey. They noted that stylistic features were evident particularly in the form of abbreviations, acronyms, emoticons, misspellings, and omission of vowels, subject
pronouns, and punctuation marks. They reported that instant messaging had more punctuation marks overall. Based on their categories of punctuation marks, the instant messaging punctuation marks were the following:

- Sentence punctuation mark (45%)
- Transmission-final punctuation mark (35%)
- Transmission-internal punctuation mark (78%)
- Use of period question mark (100%)
- Use of required period (41%)

Regarding punctuation marks, their analysis suggested that contractions (e.g., *can’t* instead of *cannot*) appeared more in casual or informal speech and writing. In this context, they concluded that omitting the punctuation mark, such as an apostrophe, was useful for users because less keystrokes were required to send messages. The broader implication of their study was that some punctuation mark uses might be associated with a domain-specific context. That is, not only stylistic differences depended upon a more narrowly defined type of content but also, in a larger sense, stylistic differences of punctuation marks might be different dependent upon a particular type of domain.

In addition, all of the previous works seemed to suggest that frequency counts are useful in detecting punctuation patterns. The types of punctuation patterns and their applicability are a separate issue, since only some of the punctuation patterns were reported from these studies. Nevertheless, the previous studies implied that more frequency counts involving punctuation patterns are needed using different corpora. Although all of the mentioned works on punctuation marks frequency analysis are useful to varying degrees, for the purpose of discovering
punctuation patterns, more varieties of frequency analysis of punctuation patterns are clearly needed.

Punctuation Marks in Text Processing Related Procedures

For a text retrieval system, text processing procedures may affect how punctuation marks are normalized. Some of components of text processing have to be taken into account in normalizing punctuation marks. Although not all text processing details are described, still, some of the basic concepts that are pertinent to normalizing punctuation marks are discussed below.

Tokenization

The notion of tokenization can be defined as a process of converting a stream of text into tokens. Thus a token can be defined as a resulting set of character strings that are produced during the process of tokenization. If undesirable tokens are produced for some reason, constituents of the tokens can be further defined for the purpose of producing desirable character strings. One of the common approaches is to attempt to tokenize text into tokens that closely resemble a more intuitive notion of words (Palmer, 2004). Yet difficulties arise since a notion of a word is often imprecise. Based on the general notion of a word, a word is a morphological unit that carries some amount of meaning.

Another way to tokenize text is to simply use blank space as a separator. In such a case, a character string or a lexical item could be split into multiple tokens. For example, a lexical item health care could be split into two separate units: health and care. Manning et al. (2008)
pointed out that one solution in a text retrieval system for such a tokenization problem is to use phrase indexing and handle searching by a phrase-based search query.

Other research indicates that tokenization often needs to carefully consider punctuation marks. Schmid (2007) pointed out some of the punctuation mark related problems that are linked to tokenization. The author claimed that periods and hyphens are particularly problematic. Since periods are used in marking the end of sentences and in abbreviations, it is often difficult to distinguish the difference. Applied to this research in a broader sense, defining words or tokens can affect punctuation normalization procedures.

Palmer (2004) also pointed out that these issues can commonly appear during a typical tokenization process. Palmer also stated that the tokenization decision in these cases is context dependent. According to Palmer, there are two types of words: *multi-word expression*, and *multi-part word*.

According to Palmer, a word can be viewed as a multi-part word as in the case of the term *end-of-line*. Also a word that can be regarded as one word is *in spite of*. On the other hand, multiple word expression can contain multiple words that are different from a multi-part word. For example, *Nov. 18, 2011* can be considered as just one word. Such a multi-word expression can contain both punctuation marks and contain a numeric expression. Palmer’s example suggests that various punctuation mark practices should be examined for a system to effectively distinguish these types of character strings.

A tokenization strategy could be a domain-specific issue. For example, Jiang and Zhai (2007) used a heuristic-based tokenization strategy for a biomedical information retrieval application. For example, their tokenization rules included the following:

1. Replace the following characters with spaces: ! "#$% & $ < > ? @
2. Remove the following characters if they are followed by a space: | ; ; ,

3. Remove the following pairs of brackets if the open bracket is preceded by a space and the close bracket is followed by a space: ( ]

4. Replace all hyphens with spaces

As a result, a biomedical term such as (MIP)-1alpha was tokenized as mip 1alpha. Additionally, above normalization rules were modified so that the tokenized text could become mip1alpha and mip 1 alpha. Their experiment showed that proper tokenization can significantly affect retrieval accuracy. The importance of their approach is that they demonstrated that punctuation marks are critical components of heuristics based tokenization rules.

Sometimes tokenization is viewed as a required step in disambiguating underlying textual elements. For this reason, tokenization is sometimes considered as a procedure in a lexical analyzer from an IR perspective (Baeza-Yates & Ribeiro-Neto, 2011). Mikheeve (2000) pointed out three different types of disambiguation problems: capitalization, sentence boundary, and abbreviation. He took a unified approach to solving this type of problem. Similarly, issues related to sentence segmentation, which is described as a process of breaking up the text to identify sentences instead of words, were examined by Grefenstette and Tapanainen (1994). Kiss and Strunk (2006) also described a method to identify abbreviation words and sentence boundaries. They identified abbreviations based on utilizing three features of punctuation marks: a final period, shortness, and internal periods.

An indication from all of these previous works is that punctuation marks can play a critical role in disambiguating text elements. Still, there has not been a sufficient amount of research in the area of tokenization as Jiang and Zhai (2007) pointed out. Also, despite the previous works, the tokenization problems did not thoroughly consider punctuation patterns. From a perspective of text retrieval, tokenization that involves punctuation marks has certain
effects on text retrieval. This type of issue was not thoroughly investigated by the previous research in this area.

Indexing and Punctuation Marks

Indexing is a common procedure in a typical text retrieval environment. A word-based indexing strategy is frequently used since a word is the smallest logical unit that carries a basic concept. In spite of its imprecision, it is natural for humans to describe subject matters in terms of words. The words in this type of indexing strategy are referred to as index terms.

One of the reasons for using index terms is that text retrieval systems can use them to match against search query terms. Hence, proper index terms are required for a system to conveniently match these character strings against terms in a search query. In general, if individual concepts are well represented by index terms, then there is a greater chance for increasing the retrievability and accessibility of underlying text. Based on this intuitive reasoning, variants of words and phrases need to be reduced to a reasonable degree so that query search terms would be properly matched with indexed terms.

Furthermore, index terms can be based on controlled vocabularies that contain a pre-determined list of terms relevant to a particular domain (Mackenzie-Robb, 2010). Otherwise, the indexing scheme can be full-text indexing, which is also sometimes referred to as free-text indexing or automatic text indexing. Savoy (2005) compared automatic text indexing and controlled vocabularies and found out that overall using one method over another did not significantly improve search precision.

Another relevant consideration is that indexing can be done manually by a human or automatically processed by a machine. A good comparison of automatic indexing was pointed
out by Anderson and Perez-Carballo (2001). In the case of human indexing schemes, professional indexers follow punctuation marks usage generally under an explicitly specified stylistic guideline, usually in the form of metadata (Moen, 2001). In a case where an index is generated automatically, a more dynamic and substantial volume of indexing terms is possible. How character strings that contain punctuation marks are indexed differs depending on the type of indexing scheme.

Regardless, punctuation normalization can be viewed as a procedure from an indexing point of view. There are common approaches that may change the way punctuation marks become indexed internally in a text retrieval system. Inverted indexing is a common word-based indexing scheme. This allows a free text searching capability even for a large corpus (Baeza-Yates & Riberio-Neto, 2011). Because an inverted indexing method is basically word-based, a more accurate identification of word is required. As a consequence, punctuation marks that surround each word need to be normalized according to the operational definition of a word. The weakness of inverted indexing is that it is not efficient for all string patterns since indexing is restricted to word or phrase queries as Russo et al. (2007) pointed out.

The other notable approach is to index subsequences of long characters, regardless of the unit size of segmentation. Wiegel (2002) reviewed these types of indexing approaches that find substrings of a text instead of word based indexing. Methods such as suffix trees and suffix arrays indexes all into this category as one long string is represented using a tree form instead of each word. Unlike inverted indexing, this technique allows searching of subsets of single long strings. A more recent indexing scheme named the Word Permuterm Index was designed to support a wide range of wild card queries (Chubak & Rafiei, 2010). The major advantage of this type of techniques is that finding character patterns can be more easily facilitated. A
disadvantage is that these methods are not suitable for a large corpus such as the Web (Baeza-Yates & Riberio-Neto, 2011).

Whether an indexing scheme is word-based or just based on a sequence of long characters, punctuation marks must be normalized by a system. For example, for indexing multi-part words that contain hyphens, important design decisions must be made. One has to make technical decisions such as how to index these words in a system and how to retrieve them. Considering the search side, what to expect from a user who may search such multi-part words must be roughly estimated to a degree.

Effectiveness of Search Queries and Punctuation Normalization

Effectiveness of punctuation normalization and search queries can be measured using different type of conditions (e.g., punctuation normalization versus un-normalized punctuation marks). The classical performance evaluation method of IR experiments is to use precision and recall measurements (Cleverdon, 1972). Precision and recall, which is often used in classic measurement of search and retrieval effectiveness, can be defined as equations (2) and (3) (Manning, 2008).

\[
\text{Precision} = \frac{\text{# of Relevant Items Retrieved}}{\text{# of Retrieved Items}} \quad (2)
\]

\[
\text{Recall} = \frac{\text{# of Relevant Items Retrieved}}{\text{# of Relevant Items}} \quad (3)
\]

Precision is a measurement of how many of the retrieved items are relevant. Recall is the ratio of relevant items returned by a retrieval system to the total number of relevant items within the collection of data. Items can be defined as any type of sensible linguistic unit (e.g., words or
phrases). The above formula basically states that in a randomly selected condition, precision is the proportion of relevant search results to all search results while recall is the proportion of relevant search results to all items matching the query. In other words, precision is the probability that retrieved items are relevant, while recall is the probability that relevant items are retrieved in a search.

Various attempts have been made to critically analyze the precision and recall measurements (Blair & Maron, 1985; Raghavan et al., 1989; Davis & Goadrich, 2006), and they are widely known in the IR community. The most notable characteristic of the precision and recall measurements is that one has an inverse relationship to another. For example, to increase a level of recall, one generally needs to perform this at the expense of precision. A goal in general is to increase precision and recall at the same time. This is generally difficult to achieve in a text retrieval environment.

In a limited sense, the precision and recall measurements can be useful in terms of understanding some of the aspects of punctuation normalization phenomena. However, the usefulness of precision and recall is not clear in terms of measuring the effectiveness of punctuation normalization. The reason is that the precision and recall measurements have not been used specifically in the context of measuring punctuation normalization.

From a system’s standpoint, it is not easy to identify the correct context of a character string. For example, if the user submits a search query that contains BC, a system has to decipher the user’s intended meaning of character string BC. The character string BC could be an abbreviation that stands for a number of items including person’s name, time (e.g., Before Christ), entity (e.g., Boston College), location (e.g., British Columbia), etc.
In general, it is difficult to determine a user’s information need solely based on the user’s query terms. Because of the large amount of text we worked with in this study, the precision and recall measurements were less a practical means to investigate the effects of punctuation normalization.

Existing Approaches to Punctuation Normalization Practices

Brooks (1998) observed how some text retrieval systems normalized punctuation marks and the effects that they had on text retrieval. His research included reviewing the practices of text retrieval system vendors such as ERIC (http://www.eric.ed.gov/) and Dialog (http://www.dialog.com/dialog/). In his research, Brooks reported diverse practices of treating and processing queries with punctuation marks. His review of normalizing punctuation marks was conducted more than a decade ago. Yet, based on the review of current punctuation normalization practices, substantial progress in this area has not been made.

First, some pertinent questions need to be raised in assessing the current practices of normalizing punctuation marks. As Brooks implied in his research, a text retrieval system designer needs to make decisions regarding normalizing punctuation marks. For example, a text retrieval system needs to answer these types of questions:

- What type of character strings that contain punctuation marks in the expression should be indexed?
- What type of punctuation marks or punctuation mark patterns should be indexed?
- How should a text retrieval system normalize user’s search query that contains punctuation marks?

Furthermore, once punctuation marks are normalized, based upon normalized character strings and un-normalized character strings, in some cases, we need to know the specific
procedures to support the matching functions. Specifically, we need to know how a text retrieval system matches terms based on the following pairs:

- Un-normalized term in the search query to normalized term in the content
- Un-normalized term in the search query to un-normalized term in the content
- Normalized term in the search query to un-normalized term in the content
- Normalized term in the search query to normalized term in the content

These questions can serve as a guideline in investigating the punctuation normalization practices. However, the above mentioned questions are not easy to answer since most text retrieval systems do not describe their treatment of punctuation marks in great detail.

There are two main ways to obtain information regarding how the punctuation marks are processed and normalized. The first source of information is the documentation that is provided by a text retrieval system vendor. However, as mentioned punctuation marks generally are not discussed in detail. The second way to obtain information is to analyze a text retrieval system based on search query results. Brooks (1998) used this method to assess the vendor practices of text retrieval systems. Some retrieval patterns should be obviously noticeable especially when documents related to punctuation marks are available. Still, knowing the full details of punctuation normalization procedures and matching functions might be still limited. Through extensive testing, we might be able accurately describe full effects of normalizing punctuation marks. Considering all of these factors and limitations, we will review two existing text retrieval systems as an example.

Google Search Engine

It is worthwhile to examine the punctuation normalization policy of Google search engine (http://google.com) based on its popularity and size. Google (2012), which has the largest
indexed web pages to date, announced that some punctuation marks will be continuously
considered for indexing. In the official Google Blog Post, it stated the following:

Based on analysis of our query stream, we’ve now started to index the following heavily
used symbols: “%”, “$”, “\”, “.”, “[“, “@”, “#”, and “+”... (Blogger, 2012).

According to Search Engine Showdown Website (2012), the changes make searching
possible for only certain cases such as “@bios” and “#cilde”. It is apparent that indexing
punctuation marks has to keep up with changes in order to meet changes to these types of
character strings. However, to what extent Google indexes and match punctuated character
strings are not widely known. It is difficult to know its methodology since Google is a “closed”
commercial system. Its detailed methods of indexing and matching and the ranking of search
results that pertains to punctuation marks are not well-documented. Answering the above
mentioned questions is severely limited, but it is still possible obtain a sense of how punctuation
marks are possibly normalized by running some search queries.

Google apparently indexes some punctuated character strings such as C++. Running a
search query that contains such punctuated character strings does not seem to be a major issue.
However, based on some punctuated queries, the effect of punctuation marks in the text retrieval
process seems to be rather evident. Figure 2 and is an example of Google search result based on
*cat -v* as a search query. Using the default search option, the Google search engine produced
more than 5 billion matched results for the search term *cat -v*. In this figure, a few top ranked
search results on the first search result page are shown. None of the first page search results is
relevant to the -v Unix command option. The titles of the retrieved search results seem to
suggest that only the character string *cat* is critical in determining the relevant search rank.
Such a troublesome problem that seems to be challenging to overcome could be partly resolved by using the exact searching query feature. That is, it is possible to match the character strings as it appears in the search query using the double quotation marks. The search results based on exact search method for the search term `cat -v` is shown in Figure 3. More than 1 billion matched results were produced in contrast to the 5 billion matched results using the default search method. As shown in the figure, the result seems to be a slight improvement compared to that of the previous search query. While the first search result page contains some non-relevant search results, the first page also contains some relevant search results. Although
some relevant search results are shown in the first search result page, it is difficult to know how punctuation marks are normalized and how they matched the terms in the content. The problem is that we attempted to assess a “closed” text retrieval system based on retrieved search results alone. To this end, we can only gain a limited sense of how internal processes such as indexing, matching and ranking work and how punctuation marks are normalized as a part of such processes.

**Figure 3.** The Google search result for the phrase *cat -v* using the exact search feature.

Punctuation marks also affect information viewing tools such as Google Ngram Viewer (Google Ngram Viewer, 2012). Ngram Viewer is program that provides the user with a visual representation of the relative frequencies of lexical bundles, or Ngrams, that have occurred in
published texts over the years (Corpus Linguistics - The Google Ngram Viewer, 2011).

Punctuation normalization policy is available from the Google’s Ngram Viewer support web page (About Ngram Viewer, 2012). The web page stated the following:

We apply a set of tokenization rules specific to the particular language. In English, contractions become two words (they're becomes the bigram they 're, we'll becomes we'll, and so on). The possessive 's is also split off, but R'n'B remains one token. Negations (n't) are normalized so that don't becomes do not. In Russian, the diacritic ё is normalized to e, and so on. The same rules are applied to parse both the ngrams typed by users and the ngrams extracted from the corpora, which means that if you're searching for don't, don't be alarmed by the fact that the Ngram Viewer rewrites it to do not; it is accurately depicting usages of both don't and do not in the corpus. However, this means there is no way to search explicitly for the specific forms can't (or cannot): you get can't and can not and cannot all at once.

Still, this type of policy information is inadequate from some user’s point of view as the details of the types of punctuation marks are adequately stated. The need for methodologically justifiable punctuation normalization policy is clearly apparent.

_LexisNexis_

LexisNexis is (http://www.lexisnexis.com) is a corporation that provides legal research services and products to legal professionals, businesses, academics, and government agencies or employees. LexisNexis maintains more than 37 billion records from more than 10,000 sources that include public, private, regulated, and derived data. The policy and guides on punctuation marks are available on the LexisNexis web page (Searching Symbols and Punctuation Marks, n.d.). The web page indicates specific policies on various punctuation marks including the section symbol §, percent symbol %, registered trademark ®, slashes /, and dollar sign $.

According to the punctuation mark policy document, some punctuation marks are ignored entirely. This seems to suggest that some punctuation marks are normalized by removing them in its entirety. For example, searches on LexisNexis ignore the question mark. The policy
document states that “The LexisNexis services do not read question marks.” (Searching Symbols and Punctuation Marks, n.d.). Even when LexisNexis search was attempted using an exact search method, the system was unable to locate terms that contained a question mark.

The policy document did not cover all punctuation marks. The plus signs were also examined, but no such punctuation marks were mentioned in the punctuation mark policy document. When we attempted to run a search query containing the term C++, the system converted the query as C, meaning that ++ were disregarded from the search. We could see that the punctuation marks were processed same way even when the exact search using the term C++ was performed. Same as previously, such punctuated character string produced an unnecessarily large amount of unrelated search results.

In certain cases, the system attempted to index and match punctuated character strings as it exactly appears. Because of multiple punctuation mark roles in a sentence, the period should need a more elaborate scheme. LexisNexis states the following policy regarding the period:

The LexisNexis® services read a decimal point when it precedes one or more numbers. The LexisNexis services read a period only when it follows just one letter and precedes a series of alternating letters and periods. Examples:

- Term: 1.2 Read as: 1.2
- Term: 100. Read as: 100
- Term: .1a Read as:.1a
- Term: F.R.C.P. Read as: F.R.C.P.
- Term: www.lexisnexis.com Read as: www lexisnexis com (Searching Symbols and Punctuation Marks, n.d.).

Thus, some terms with periods are ignored while other terms with periods are not ignored. Ignoring the period at the end of a character string produced odd results:

- For an exact search query using the term P. O., the search result showed the instances containing P&O.
- For an exact search query using the term B.C, the search result showed the instances containing B.C Without the period after the letter C.
For an exact search query using the term B. C., the search result showed the instances containing B&C.

Such inadequate search results seem to indicate that some punctuation marks might be normalized arbitrarily without a careful analysis. These examples support Brooks’ (1998) finding that punctuation normalization issues are fundamental impediment to text retrieval systems. A methodologically justifiable way of normalizing punctuation marks is clearly needed from the perspective of text retrieval.

Concepts Related to Punctuation Normalization

In the literature that covered text normalization concepts in general, issues related to punctuation normalization were mentioned frequently. Sometimes text normalization techniques can be integrated within the system or the concepts overlap one another. The concepts related to punctuation normalization can be categorized into the following categories:

- Character encoding and character normalization
- Normalization of capitalized character strings
- Normalization of non-standard words and abbreviation

Each of these concepts is reviewed from a text retrieval perspective in this section.

Character Encoding and Character Normalization

Punctuation mark normalization problem can be associated with character encoding and character normalization problems. Different character encoding schemes (e.g., ASCII, Unicode, etc.) are available. Useful information regarding different character encoding schemes was provided by Korpela (2012a). Basically, each encoding scheme offers a different set of characters. Some characters sets in these encoding schemes are often incompatible. The primary
reason is that different character sets were formed at different times for different reasons. Also, each encoding system offers different sets of characters in order to accommodate a particular language’s representation requirement.

Essentially the problem is that a character can be represented by multiple forms and codes. To reduce the variants of character forms, character normalization has been suggested in the literature (W3C website, 2012). For instances, a̧ can be represented as a single character, or it can be represented as two combining sequences of characters â and ̋. Character normalization is mostly concerned with issues related to normalizing different characters and typographic character variations. The notion of character normalization is closely associated punctuation normalization. The problem domain also involves normalizing one type of punctuation mark to another.

All available characters in a character set are assigned a specific numbered code due to slight typographic font differences. Since the ASCII character set is one of the oldest and the most basic typographic representation of characters, for this study, we considered only the punctuation marks listed in Appendix A. Note that there are some typographic variations in this table. ASCII code decimal number 39 ′ is assigned to the apostrophe or the single quote. ASCII code decimal #145 refers to the left single quote ′, while ASCII code decimal #145 refers to the right single quote ″.

A useful source where character set normalization issues are discussed is the Unicode normalization documentation (The Unicode Consortium, 2012). Unicode is a character encoding system that offers a richer set of characters than ASCII. The technical report published by Unicode describes problems specifically related to a Unicode environment. Because of the size of the character set that it supports, there are fundamental issues in dealing with canonical forms
within a Unicode character set. Regardless, a character string that is not in English can be represented by different sequences of Unicode characters.

For example, the character string café has the letter é. The letter é in Unicode is (U+00E9). It is a Latin small letter e with acute, but the same letter can be also represented by (U+0065). In this case, the letter is a Latin small letter e followed by (U+0301) which is “combining acute accent or a diacritic mark”. The first form is known as precomposed, while the latter is known as decomposed form. Four normalization forms are possible according to the Unicode Standard:

- NFC (normalization form canonical composition)
- NFD (normalization form canonical decomposition)
- NFKC (normalization form compatibility composition)
- NFKD (normalization form compatibility decomposition)

Because of the variant forms and characters that Unicode attempts to support, character normalization issues arise during the course of combining or decomposing character from one to another equivalent character form. Adhering to the Unicode normalized forms allows programmers to follow standard ways of dealing with individual characters.

Character normalization is concerned with normalization pertaining to individual characters due to the variations in a character encoding scheme. The issue of character normalization is not directed at examining functions (e.g., syntactical roles) of individual punctuation marks. Instead, the issues are the character differences and representation issues that can occur on individual character level. This is not to say the specific character strings that are affected by a character normalization procedure are any less important. If a character normalization problem affects a single character for whatever reason, then it can also affect the
entire character string since it contains the particular character. EUROSEC GmbH (2005) pointed out that understanding character set differences and their conversion issues is essential in an effort to reduce unintended normalization effects.

Moen (2005) identified similar problems that are associated with character normalization. Moen demonstrated that the issues related to normalization of certain characters concern not only the proper representation of characters, but also proper retrievability of an equivalent character. For instance, the letter á, which is a Latin small letter with a tilde, is supported by Unicode with the character (U+00E3). However, this character is not present in the original ASCII code. See Appendix A for the list of ASCII codes that represent this study’s punctuation marks. A proper conversion is necessary if one wants to capture the same type of meaning in an ASCII form. Otherwise, the character must be converted to a plain lower case letter a. While this normalization of character might be better suited for matching the letter á, the original character information has been altered and the meaning of the character has changed.

Normalization of Capitalized Words

Traditionally, case sensitive searches are not provided in many text retrieval systems. That is, text is usually converted to all upper case or lower case in the preprocessing stage of IR (Baeza-Yates & Riberiro-Neto, 2011). The simplest method to normalize capitalized character strings is to change capitalization to either all in upper case or all in lower case. Manning et al. (2008) pointed out that in practice usually all upper case characters are converted to a lower case. They referred to this process as a case-folding (e.g., changing the character string CAT to lower case cat). It should be pointed out that another alternative, although inefficient, is to have a system to retrieve the character string both ways and merge the results. Retrieving both ways
achieves the same goal since from a user’s standpoint the returned results may appear to be insensitive. In either way, it relieves a user from searching both ways—one in upper case and one in lower case.

Although capitalization issues seem to be a straightforward problem in this sense, the character strings that are already capitalized or not is a useful characteristics that a system can use for a variety of applications. For example, capitalized character strings often denote the start of a new sentence. They can also denote proper names such as names of organizations, locations, people, artifacts, etc. (Quirk et al, 1985). As pointed out earlier, from an IE standpoint, Mikheev’s work (2000) suggested a method to disambiguate capitalization in conjunction to identifying abbreviations. The significance of Mikheev’s work is that normalization of an abbreviation should utilize underlying capitalized character strings in a text since capitalization can also play an important role in identifying abbreviation marks.

Punctuation Marks in Forms of Words and Expressions

Punctuation marks can appear in various forms of character strings and expressions. Some of the previous works in this area have pointed out various categorizations of forms of character strings and expressions. They were particularly useful from the view of analyzing the punctuation mark roles. Punctuation marks usage in two particular forms of character strings and expressions that were useful to observe are: abbreviations and non-standard words, and non-standard expressions.

*Abbreviations and Non-Standard Words*

Abbreviations and non-standard words often contain punctuation marks. Non-standard
words do not typically appear in a dictionary. Xydas et al. (2004) stated that non-standard words do not have a regular entry into a lexicon and their pronunciation needs to be generated by a more complicated natural language process. Apart from this, natural languages often contain a vast number of non-standard words that a typical text retrieval system must deal with. These include categories based on textual characterization such as numbers, dates, currency amounts, and acronyms.

From an IE side, the Named Entity Recognition problem often involves matching abbreviated character strings to their non-abbreviated character strings. Since abbreviated character strings often present a unique problem, solving the problem generally requires a domain specific solution. A number of researchers attempted to identify abbreviation in the text. For example, Potamites (2007) examined the nature of abbreviation and claimed that his system is fairly accurate in determining whether the textual element is an abbreviation or not. He focused on identifying abbreviations based on punctuation marks and boundaries between words.

Other studies have shown various ways to normalize non-standard words. Interesting textual patterns involving punctuation marks were observed by Yeates (2002). Yeates focused on the automatic extraction of acronyms within text. He pointed out the differences between an acronym and an abbreviation by arguing that there is a lack of symbols such as apostrophes and or periods in acronyms. On the other hand, he pointed out that an abbreviation conforms to more standard construction and uses capital letters. As an example, he claimed that the word can’t and etc. are contractions and abbreviations respectively; he argued that these words are not acronyms.

Like others, Yeates (2002) took a heuristic approach in finding acronyms in the text. According to Yeates, acronyms have the following characteristics:
• Acronyms are shorter than their original specification
• Acronyms contain initials of most of the words in their definitions
• Acronyms are usually in upper case
• Shorter acronyms tend to have longer words in their definition
• Longer acronyms tend to have more stop words such as the word the and a

His work is particularly useful in devising heuristic based rules for identifying acronyms.

Sproat et al. (2001) investigated various non-standard words in the context of speech recognition. Although some non-standard words do not involve punctuation marks, his contribution was mainly on identifying various non-standard word types. The non-standard words are often categorized into different types and specific models are designed to tackle each type. His definition of non-standard word is a word that falls into one of his predefined types of word. For instance, Sproat et al. suggested four categories for alphabetical character based non-standard words: word sequence containing abbreviation (EXPN), letter sequence (LSEQ), standard word (ASWD) and misspelling (MSPL). Examples of each category are the following:

• EXPN: adv, N.Y., mph, gov’t
• LSEQ: CIA, DC, CDs
• ASWD: CAT, proper names
• MSPL: geography

Sproat et al. (2001) indicated that not all non-standard expressions contain punctuation marks. For example, adv might stand for advanced. However, a character string such as gov’t, which contains an apostrophe, is an abbreviation but a non-standard one. Nevertheless, his categorization is useful in terms of recognizing punctuation marks that are associated with non-standard words since these words often contain punctuation marks as well.
Likewise, for a medical domain, Pakhomov (2002) demonstrated a novel approach to normalize acronyms and abbreviations. As shown in Table 2, Panchapagesan et al. (2004) identified non-standard words into different types. Some of these types appeared to be useful in addressing punctuation normalization issues. Most of these words contain punctuation marks in one form or another.

Table 2

Panchapagesan’s List of Non-Standard Words

<table>
<thead>
<tr>
<th>Non-standard Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Cardinal numbers and literal strings</td>
</tr>
<tr>
<td>• Ordinal numbers</td>
</tr>
<tr>
<td>• Roman numerals</td>
</tr>
<tr>
<td>• Fractions</td>
</tr>
<tr>
<td>• Ratios</td>
</tr>
<tr>
<td>• Decimal numbers</td>
</tr>
<tr>
<td>• Time</td>
</tr>
<tr>
<td>• Telephone numbers</td>
</tr>
<tr>
<td>• Date</td>
</tr>
<tr>
<td>• Year</td>
</tr>
<tr>
<td>• URL, E-mail, etc.</td>
</tr>
<tr>
<td>• Range</td>
</tr>
<tr>
<td>• Percentage</td>
</tr>
<tr>
<td>• Alphanumeric strings</td>
</tr>
</tbody>
</table>

The primary limitation of previous studies, however, is that they did not conduct their research in the context of text retrieval. For example, none of these studies examined abbreviations with search queries that contain punctuation marks. These studies predominantly focused on correctly identifying certain types of abbreviated character strings. Because of this reason, the effects of searching abbreviated character strings in a typical full-text retrieval setting have not been adequately addressed.
Non-Standard Expressions

Occasionally, a text may contain non-standard expressions rather than non-standard words. Such expressions may contain non-standard capitalization and punctuation marks. For example, texts that are based on casual conversions (e.g., online posting) may lack standard capitalization. Rasulo (2008) reported the efficiency of time and effort is the main reason why people do not use conventional capitalization and punctuation marks. For example, he reported that a sentence such as (4) is non-standard because it lacks appropriate capitalization and uses multiple question marks.

\[ \text{can i phone anyone, need help????} \]  

(4)

The extent of how non-standard expressions affect text retrieval is difficult to estimate since it is a complicated issue that is often domain specific. Also, the extent that non-standard expressions are used in real-world corpora is not widely known. Rennie (2007) stated that capitalization and punctuation features are noisy in informal communication. Nevertheless, from a text retrieval view, an indication based on Rasulo’s study is that domain-specific knowledge in conjunction to IE techniques is most likely needed.

Chapter Summary

By providing the literature review, a number of important aspects were highlighted in the context of punctuation normalization in this chapter. The theories from linguistic studies revealed a range of punctuations mark roles. Although constructed from a perspective of computational linguistic theory, Jones provided a useful categorization of punctuation marks. The previous works related to punctuation normalization dealt with issues such as character encoding and character normalization, capitalization, and non-standard words and abbreviations.
Some of the previous works focused on analyzing punctuation marks in different contexts. These studies suggested that a underlying complex relationship exists among different text segments (e.g., character, word, phrase, etc.) and that punctuation marks can be a part of this complex relationship.

Although issues related to punctuation marks were abundant, the previous studies did not conduct their research in the context of text retrieval. Consequentially, the previous research related to this area did not provide effects of punctuation normalizations from a text retrieval perspective. Still, the implication of the previous studies is that there are a number of important aspects, such as normalizing abbreviations that need to be addressed in investigating punctuation normalization effects. The previous studies suggested that for an effective text retrieval of character strings such as abbreviations, punctuation patterns should be examined thoroughly.
CHAPTER 3
METHODOLOGY AND RESEARCH DESIGN

This chapter presents methodological approaches, detailed procedures of the study, and issues related to analyzing the data. In essence, this research could be roughly characterized as an exploratory and analytic study. The attempt was made to analyze relevant factors for the purpose of exploring punctuation normalization issues and implications for text retrieval. Several key procedures were used in carrying out this research. These procedures are discussed in detail. The chapter ends with a discussion on the limitation of the study.

Overview of Research Strategy

Once again, the main research question that this study attempted to answer was the following: What are the implications of normalizing punctuation marks from the perspective of text retrieval? To answer this research question, a punctuation frequency analysis was conducted to examine real-world corpora. Along with the results of the punctuation frequency analysis, we applied additional analytical procedures to explore the issues surrounding punctuation normalization.

In this study, various factors had to be considered including type of users, search and retrieval types, indexing methods, content type, etc. Since it was not easy to quantify any of these factors for the study, using a holistic, integrated approach to the problem was suitable to answer the research questions.

1. In what ways and to what extent are punctuation marks and punctuated character strings (i.e., character strings that contain punctuation marks) important in terms of text retrieval?

2. How does punctuation normalization affect the process of matching punctuated character strings?
3. What are some of the useful categories of punctuation marks, punctuation patterns, and punctuation normalization?

4. How can we measure similarity between terms in an orthographic sense so that it can be incorporated into normalizing punctuation marks?

5. In what ways can we use underlying punctuation patterns in a text for a text retrieval system?

However, relying on a punctuation frequency analysis alone was not the most effective way to answer the research questions. Additional analytical procedures seemed to be appropriate to answer the research questions. For this reason punctuation frequency analysis was initially used to explore the underlying text for gaining insights into issues related to punctuation mark normalization. Then, the normalization strategy analysis procedures were incorporated. Often, these analytical procedures were based on hypothetical search queries to explore punctuation normalization issues as they were necessary components in finding the effects of punctuation normalization. The latter parts of the study incorporate this type of analysis, and the details are presented in Chapter 5.

In terms of methodological approach, this study is also exploratory and descriptive in nature. Punctuation mark normalization is a complex problem that needs to be addressed in a holistic and systematic way. Also, since prior research did not specifically address this issue, there has been a lack of research directly dealing with punctuation normalization issues in the context of text retrieval. The methodology is partly descriptive since this study describes phenomena systematically to reveal patterns and their types.

In addition, a comparative analysis approach underpinned the research design. For example, frequency counts from two different datasets were useful for identifying domain- and genre-specific characteristics of punctuations. Then, the overall analysis results could become a focal point for understanding of the normalization decision making process.
The Analysis Setting

As in most research studies, an appropriate type of data was needed for this research. It was desirable to select datasets that represent different types of genres or different types of domains. After viewing various types of text-based contents, this research decided to use two different datasets: the Gutenberg Project archive (http://www.gutenberg.org) and the Usenet Newsgroup archive (Shaoul & Westbury, 2011).

Coincidentally, after deciding to use the Gutenberg archive and the Usenet archive for this research, we found out that Jones (1997) also used these two datasets for his research. Our research can be used to confirm the Jones’ statistical results. However, for Jones’ research, the focus was on functional roles of inter-lexical punctuation marks. The purpose of his research was much different. Since our purpose was to examine punctuation normalization issues, Jones punctuation mark frequency analysis results were useful to a limited extent.

The Gutenberg Project archive (http://www.gutenberg.org) is a corpus that contains contents mostly from e-books. These are the full-texts of books from publications that are out of a copyright. The Gutenberg Project e-books represent a wide range of topics such as literature, cookbooks, reference works and issues of periodicals.

For the Gutenberg Project, we downloaded a large amount of e-books that were in plaintext file format. We downloaded all available files from the Gutenberg archive ftp site, and a total of 24,657 files were downloaded. Varying in length, each file was a complete book or an issue of periodical. In total, the books downloaded were approximately 12 Gigabytes in the form of uncompressed text files. The Gutenberg e-books were organized according to different file numbers. The lowest numbered file was 10.txt, while the highest numbered file was 36444.txt.
The Usenet Newsgroup archive (Shaoul & Westbury, 2011) is a corpus that consists of newsgroup posting collection for the year 2009. The 2009 Usenet Newsgroup archive exceeded more than 30 Gigabytes of data. The primary reason for including the Usenet Newsgroup posting archives was that it contained a vast amount of modern linguistic expressions and vocabularies that people commonly use in daily lives.

The Usenet was organized into different discussion groups. The compressed file we downloaded contained 51 files. These files are the collections of 47,860 discussion groups, containing only parts of the discussion that were posted by the Usenet’s users. The actual names of files are shown in Appendix B. It was unclear as to the number of discussion groups that each file contained, but by browsing the content of each file, we determined that each file contained a mixture of different discussion groups.

To clarify, in the case of Usenet, the term “dataset” refers to the collected, sampled and reduced collections, from the archive in this study. Two terms, “archive” and “corpus”, can be used synonymously as they refer to the same object in this study. In reducing the corpus to each dataset, balance between the performance of processing the large of text and the varieties of linguistic patterns including punctuation marks that each dataset could contain had to be carefully weighed. The goal was to reduce the corpus into a modest size to speed up the processing of the data, but still sufficiently large enough so that they could be used to contain sufficient quantities and varieties of punctuation patterns. To estimate the processing speed, we ran a few Unix commands such as sort. Based on our preliminary examination with the data we downloaded, we decided to use approximately 200mb of data for this study. Figure 4 illustrates the overall process of creating the two datasets.
For each archive, the number of files and the size of the each file were different. Having two datasets with equivalent size made more sense. In this manner, comparing the punctuation mark analysis results from two archives could become much more perceptive during the course
of analysis. To reduce the corpus into the datasets, different procedures were needed for each archive. Random selection seemed necessary to reduce potential selection bias.

To create two equal size datasets, one for Usenet and one for Gutenberg, the file size was used as an initial guide. Working with individual files was inconvenient in many ways so we decided to create two files: a file for the Usenet archive and another file for the Gutenberg archive. They will be referred to g.temp for the Gutenberg dataset file and news.temp as the Usenet dataset file. A news.temp was created first and then a benchmark was used to create the file g.temp in order to create equivalent file sizes.

First, in creating the file news.temp, the file size needed to be approximately equivalent to 193,105,778 bytes since this was the file size of g.temp. The average file size of 51 files in the Usenet archive was approximate 500mb (500,000,000 bytes). This meant that only a small amount of each of 51 Usenet file was necessary in creating the file news.temp. The top 100,000 lines of each of 51 files used in creating the news.temp file were extracted. In other words, the extracted lines from each of the 51 files were “appended” to news.temp. Data cleaning was necessary since irrelevant and spurious text that was present in the dataset. The cleaning process reduced a fraction of the file size and the final file size of 218,807,347 bytes for the file news.temp.

To create the file g.temp, a program Unix shell script was written to extract text segments from 24,657 files that were mentioned. Each file contained some header that included descriptions of the content that was not part of the content. For each Gutenberg e-book text file, 165 lines were extracted from each file to create g.temp, starting from the line number 401. To determine the exact lines to extract for g.temp, the dataset creation procedure was performed interactively. Then some data cleaning was necessary due to some irrelevant content and errors.
There were more issues with cleaning the dataset in the Usenet than the Gutenberg dataset. Problems detected in the Usenet dataset were:

- Metadata type of information that is irrelevant to the actual content. For example, the Usenet dataset contained the line “---END.OF.DOCUMENT---” at the end of each posting.
- Most often the responses of to someone’s Usenet posts started with the right arrow signs and it was decided to remove these arrows.
- Duplicate posting was detected throughout the Usenet corpus.
- Some single arrows > were also placed as a soft line break.
- A fair amount long sequences of hyphens ---- and double arrow right arrow marks >> were embedded in the text. Sometimes an equal sign = was placed at the end of each line. These types of punctuation marks were removed.

It appeared that the double arrows >> in the Usenet dataset could be referred to as an embarrassing line wrap according to Gellens (2004). Since the archive was in a plain text format, the embarrassing line wrap up were created during the conversion to 80 characters per line. Thus, it denotes a continuation of the previous line. The double arrows were used as quoted messages as replies or forwarded messages. It is unclear how they were accumulated, but the report written by Gellens (2004) provided various issues related to these types of characters.

A complete and extremely accurate cleaning was difficult to achieve. For the purpose of carrying out this research, these types of metadata information and punctuation marks were judiciously removed. Also, in the Usenet dataset, there were a very few non-printing and control character that were not specified in Appendix A. Since an ASCII character such as a TAB was not considered as a punctuation mark, they were removed as they appeared. These types of unrelated information could be viewed as anomalies for understanding the effects of punctuation...
mark normalization. It also reflected a real world scenario where the text might contain irrelevant meta-information and erroneous strings that needed a thorough cleaning procedure.

The Gutenberg dataset also contained a slight amount of idiosyncratic text characteristics. Some patterns of punctuation marks were “local” to the Gutenberg dataset. For example, periods were used extensively to fill certain spaces. These punctuation marks were left in the datasets as they clearly required further investigation for the purpose of this research. Such punctuation marks were regarded as formatting punctuation marks since such punctuation marks were mainly used for structural purposes in a given text. In a technical sense, removing these types of strings from the text could be considered as punctuation normalization. It became apparent that removing formatting punctuation marks could be done in different stages in the process of developing a text retrieval system.

At the end, the total file size of g.temp was 193,105,778 bytes. Once g.temp and news.temp were created, the tokenization procedures for both files were applied. A blank space based tokenization was applied and this created a tokenized dataset–g.data. The same blank space based tokenization was applied to news.temp to create a tokenized file–news.1. Although two tokenized datasets were used most of the time, sometimes g.temp and news.temp, which are un-tokenized dataset files, were needed to retrieve text lines that contain particular character strings. A trial and error process was necessary to obtain files of approximately equal size. The file size of news.1 was 188,080,560 bytes, while the file size of g.data was 184,825,251 bytes.

The tokenization process is described later in this chapter. More detailed tokenization procedures along with basic statistics of tokenized character strings are provided in the next chapter.

These two datasets were available in a plaintext file format. Although the downloaded
plaintext file was retrieved first, the exact character set encoding schemes (e.g., ASCII, Unicode, etc.) and the extent that character set differences might become an issue were not determined at first. In retrospect, perhaps the issues should have been investigated in the beginning. In the subsequent chapter, these character set differences are described more in detail.

Both datasets contained textual contents that we can commonly describe as a natural language. This meant that text retrieval issues pertained to natural language instead of metadata or controlled vocabularies. Although the text in both datasets is based on natural language, each dataset reflected very different genres. Since the language used in the Usenet dataset was mostly informal in terms of language usage, it contained unrestricted language, with non-standard expressions. The text in Usenet did not conform to any specific rules of writing or a stylistic guideline as Jones (1997) pointed out. At the same time, it contained more grammatical errors due to the nature of unfiltered information.

In contrast, e-books from the Gutenberg Project had gone through typical publication procedures. As far as genres of the contents are concerned, a considerable amount of the Gutenberg books could be labeled as classics, implying that the dataset probably contained lesser amount of contemporary slang, idioms, and abbreviations. In other words, the Gutenberg Project e-books reflected more traditional, common expressions that typically appear in standard published books. To this end, the punctuation mark usages were expected to contain fewer errors since a typical published work goes through editing process.

The primary reason for using the two datasets for this study was that a comparative analysis strategy could be employed. This strategy was adopted based on the presumption that the multiplicity and variety will offer a greater, richer mixture of contents and language usages,
and a variety of punctuation mark types and usages. As a consequence, interpreting analysis results based on comparative observations was expected to strengthen this study’s findings.

The examination of punctuation marks using the Usenet and Gutenberg corpora for examining punctuation marks is not entirely new. Among other corpus, Jones (1997) used datasets from Usenet and Gutenberg Project for his study. As indicated in the earlier chapter, he previously reported basic frequency counts of different punctuation marks based on his sample. This study was an extension of research done by Jones (1997) and Meyer (1983). However, the overall extent, goals, and applications of punctuation marks in the current study were entirely different from these previous studies on punctuation marks.

Lastly, this study used an analysis that required some computer programming. Frequency counts of different text-based units (e.g., punctuation mark, punctuated token, etc.) were conducted by custom programming since a software tool to automatically analyze punctuation marks was not available. A combination of a Unix-based scripting program (e.g., grep) and the C++ programming language were employed in this study. To operate in a Unix-like environment, Cygwin (http://www.cygwin.com/) was actually used inside the Microsoft Windows operating system.

Methodological Procedures

Figure 5 illustrates the overall methodological stages that were involved in this research and their corresponding dissertation chapters. Successful completion of all these procedures of the study was necessary to achieve the objective of this study. The methodological procedures were “iterative” to a certain extent. The flows of these iterative steps are indicated as the “feedback” process in Figure 5. In this study, tokenization was viewed as an initial starting point
of categorization for the punctuation frequency analysis. In the course of following the methodological procedures, it was necessary to revisit a previous procedure occasionally to improve the study’s findings.

Figure 5. Major methodological procedures and corresponding chapters.

Highlights of Methodological Procedures

The rest of this chapter describes the methodological procedures and any issues that were encountered. In particular, the basic issues are discussed in respect to the following specific sub-components:
• Tokenization (segmentation)
• Frequency counts and punctuation mark types
• Additional types of punctuation
• Punctuation normalization types
• Types of search queries and punctuation normalization effects
• The punctuation normalization demonstration

Tokenization (Segmentation)

Punctuation frequency analysis in this study starts with a tokenization procedure. The tokenization procedure in this study was to automatically convert the underlying text into different tokenized elements (e.g., punctuated token). The basic strategy was to count occurrences of tokens based on different punctuation mark types.

In this study, tokenization was carried out based on blank spaces for practical reasons:

• The use of blank space was the most simple means to identify tokens, and a token in this study is a fundamental unit that needs to be identified.

• Categorizing different types of punctuation marks based on their string position could be consistently observed.

The strategy was to consider a more sophisticated tokenization scheme only after understanding the implication of the blank space based tokenization procedure. As mentioned in an earlier chapter, previous works addressed some of the issues related to tokenization. However, the intention of this study was to investigate the punctuation patterns along with punctuation normalization phenomenon by focusing on blank space based generated tokens as a starting point of investigation.
In determining the fundamental unit of analysis, defining a notion of *punctuated tokens* was useful. It is also similar to punctuated character strings in a sense that each token contains alphabetical characters and punctuation marks. For example, the token `C++` is a punctuated-token. In this case, the letter `C` is an alphabetical character component and double plus signs `++` are the punctuation mark component.

If the punctuated-token is defined based on a blank space as a separator, the character string such as `B. C`, which has a blank space in the middle, could be decomposed into two separate punctuated tokens:

- `B.`
- `C.`

Using only blank space based tokenization might not be efficient in terms of developing an effective punctuation normalization scheme. Often treatment of separated tokens can be permitted since they can be collectively searched as a phrase. For this reason, generally the individual token positions of the text needed to be recorded if they are treated separately as tokens.

In particular, the decision to use punctuated tokens was twofold. First, punctuation normalization issues could be explored based on affix-based types as punctuation mark defined in the subsequent section. Second, some of the punctuation mark characteristics and their roles could be discovered using a blank space based tokenization procedure. Unless noted otherwise, we assume a blank space based tokenization, and punctuated token refers to a blank space based token.
Tokenization also could be applied to process a user’s search query. For example, a user’s search query may contain any one the following abbreviated terms: \textit{B.C.}, \textit{B.C.}, and \textit{BC}. Assuming that all of these character strings are considered semantically identical, only stylistic differences in terms of punctuation marks were evident. Yet notice how a blank space based tokenization can become a relevant issue. The first character string \textit{B. C.} consists of two individual punctuated tokens \textit{B.} and \textit{C.} according to our rule of tokenization. If these tokens were treated individually, semantically they do not have the same meaning as \textit{B. C.}. Only combined tokens could properly hold the intended meaning term. Such issues had to be further probed during the tokenization process.

In many cases, a punctuation mark was not a distinguishable feature in terms of identifying its role in the text. Only if the punctuation marks are used in conjunction with alphabetical characters, are punctuation marks able to provide a more definitive and proper context. The underlying rules of punctuation mark usages were flexible enough so that many combinations could be formed but each might represent a slightly different sense of a character string. This rich flexible set of features was also a factor in text retrieval since they require proper normalization to effectively retrieve relevant items.

\textit{The Affix-Based Punctuation Mark Types}

Based on the notion of the punctuated tokens, individual punctuation marks were categorized according to their relative text character position. Initially, identifying different types of punctuation marks and punctuated character string based on their operational characteristics allowed a succinct and accurate referencing capability. By categorizing or grouping punctuation marks based on their locality, the punctuation normalization problem was
closely examined. As a result, the consequences and retrievability related issues could be better understood. For the rest of the study, these terms along with additional punctuation mark types are discussed. The punctuation mark types are important attributes in this study since they enabled us to address a variety of punctuation normalization problems. Three fundamental affix-based types of punctuation marks can be formed if a blank space was used as a delimiter and a character string separator. These are the following:

- Prefix-punctuation mark
- Suffix-punctuation mark
- Infix-punctuation mark

These terms are created for this study to refer to specific locations of punctuation marks. Prefix-punctuation mark appears in the beginning of each token. More than one prefix-punctuation mark can be generated by concatenating other prefix-punctuation marks. Suffix-punctuation mark appears at the end of each character string. More than one suffix-punctuation mark can be generated by concatenating with suffix-punctuation marks.

Infix-punctuation marks can appear in any token position except with beginning and at the end of each character string. Moreover, infix-punctuation marks must not be in the position of prefix and suffix. More than one infix-punctuation mark can be generated by concatenating other infix-punctuation marks. Occasionally there can be more than one disjointed infix-punctuation mark in a given token (e.g., *all-you-can*). These individual infix-punctuation marks can be referred to as *disjointed infix-punctuation marks*. Regardless of the type of infix-punctuation marks, they are just collectively referred to as infix-punctuation marks.

Table 3 provides an example of basic affix types of punctuation marks. Note that some marks can be associated with more than one affix types. As shown in Table 3, if a decomposed
Token is [1989-1999], then this decomposed token contain all three affix-based token types: prefix-, infix-, and suffix-punctuation marks. In this case, the prefix-punctuation is [, the infix-punctuation is -, and suffix-punctuation mark is ].

Table 3

Example of Punctuation Mark Types and Tokens

<table>
<thead>
<tr>
<th>Textual Element</th>
<th>Punctuated Token</th>
<th>Punctuation Mark</th>
<th>Punctuation Mark Affix Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>question?</td>
<td>question?</td>
<td>?</td>
<td>suffix</td>
</tr>
<tr>
<td>AT&amp;T</td>
<td>AT&amp;T</td>
<td>&amp;</td>
<td>infix</td>
</tr>
<tr>
<td>12:00</td>
<td>12:00</td>
<td>:</td>
<td>infix</td>
</tr>
<tr>
<td>C++</td>
<td>C++</td>
<td>+</td>
<td>suffix</td>
</tr>
<tr>
<td>Easy.</td>
<td>Easy.</td>
<td>.</td>
<td>suffix</td>
</tr>
<tr>
<td>O’Brien</td>
<td>O’Brien</td>
<td>’</td>
<td>infix</td>
</tr>
<tr>
<td>&lt;email address omitted&gt;</td>
<td>&lt;email omitted&gt;</td>
<td>&lt;</td>
<td>prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;</td>
<td>suffix</td>
</tr>
<tr>
<td>B.C.</td>
<td>B.C.</td>
<td>.</td>
<td>infix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.</td>
<td>suffix</td>
</tr>
<tr>
<td>[1989-1999]</td>
<td>[1989-1999]</td>
<td>[</td>
<td>prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>infix</td>
</tr>
<tr>
<td></td>
<td></td>
<td>]</td>
<td>suffix</td>
</tr>
</tbody>
</table>

A punctuation mark such as a period can become ambiguous if other surrounding textual elements are not closely examined. In Table 3, the token *Easy* contains a period at the end. This period, most likely, signifies a full-stop. However, the two periods in the token *B.C.* do not contain full-stop, an end of sentence marker. If such an abbreviated token were found at the end of a sentence, then the role of the ending period after the letter *C*, whether it is a full-stop or a period after an abbreviation, might be more difficult to determine computationally. We would have to craft an exception rule since the period after the *C* should be categorized as both period and full-stop.
Such diverse positions and uses of punctuation marks are linked to the complexity of punctuation mark roles in our language. One of the initial observation made from associating text into different affix-based tokens was that disambiguation of the role of the period was necessary. In figuring out the contextual meaning of each token, unlike a human judgment, a computer-based processing program often requires heuristic based algorithms. However, such algorithms tend to be ad-hoc in most cases. Even with some type of disambiguation procedures, unpredictable behavior could be expected during the text retrieval process. This was another reason that a systematic method to identify and categorize relevant punctuation patterns had a practical benefit in understanding punctuation normalization issues.

**Frequency Counts and Punctuation Mark Types**

One of the reasons for performing various frequency counts was that the results could expose hidden characteristics of punctuation marks that were particular to a dataset. Jones (1997) reported overall count of punctuation marks using samples drawn from Usenet and Gutenberg. This study, however, employed a frequency count strategy by examining the affix-based token types and punctuation patterns. Also, based on frequency counts of punctuation patterns, additional useful types could be further identified. During the course of this study, attempts were made to categorize and detect some of the ambiguous punctuation marks.

Once different types of frequency counts were obtained, the results had to be sorted and ranked. The next step was to investigate the implication of frequency counts. In general, a higher frequency count indicated the greater uses of particular punctuation marks, punctuation patterns, or punctuated character strings. The punctuation mark frequency count strategy was
basically used to obtain underlying characteristics of the text of the datasets. Often we were able to further develop punctuation mark categories by employing additional frequency counts.

Initially, through the frequency counts of affixed based punctuation marks, a certain understanding of problems associated with punctuation mark roles could be obtained. The frequency counts also pointed toward locating punctuation patterns and examining more specific patterns in datasets. Examining punctuation marks in conjunction with punctuated character string was necessary in certain cases. In other cases, an inspection of an entire sentence by retrieving the lines that contain the character strings was necessary to determine the particular role of a punctuation mark. In this respect, the frequency count assisted in identifying potentially ambiguous roles of punctuation marks. In most cases, it often called for additional inspection into the punctuation mark types. In a larger sense, factors that potentially influence punctuation normalization and effects of the normalization were recognized in this procedure.

*Additional Types of Punctuation*

Additional categories of punctuation marks and punctuation patterns could be developed based on the datasets. The purpose was to gain further insights into the punctuation patterns. The categories proved to be an effective means to analyze the punctuation normalization effects. One possible approach in handling the punctuation normalization problem was to consider a finite number of plausible punctuation patterns in conjunction with search queries. These variant categories were constructed logically based on noticeable punctuation patterns.

Certain textual characteristics associated with punctuation patterns were obtained through various frequency counts within one dataset. The underlying research strategy was to compare two datasets as mentioned earlier. The purpose of using the comparison of datasets was to
further strengthen the research findings. The frequency count results from the two datasets were varied in different ways.

Since we wanted to compare two datasets, we conceptualized different outcomes that a comparison of punctuation marks could yield. As shown in Figure 6, the strategy to compare and contrast frequency results could be represented by using a two dimensional space. In each of the analysis scenarios, plausible explanations have to be sought, triggering a further investigation to the deeper problem domain.

![Figure 6. Possibilities of punctuation marks in two datasets.](image)

As shown in the figure, if we label each frequency count as a binary result “high” or “low”, the combination lead to four distinct possibilities. Descriptive interpretation of Figure 6 is shown in Table 4.

One of possibilities is that the frequency occurrences of punctuation patterns can be high in one dataset but low in another dataset. Region 2 and 3 represents the areas where such conditions can be observed. In these cases, a plausible explanation was further sought. These
cases needed to be further investigated since it could explain reasons for substantial differences between frequency count results. In most cases, substantial frequency count differences were linked to underlying domain characteristics. Another possibility was that the frequency occurrences of punctuation patterns can be both high and both low. Region 1 and 4 represents the areas where such conditions can be observed.

Table 4

Possible Interpretation of Figure 3 Scenarios

<table>
<thead>
<tr>
<th>Region</th>
<th>Dataset A</th>
<th>Dataset B</th>
<th>Possible Interpretation of Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High</td>
<td>High</td>
<td>The occurrences of the punctuation mark in both datasets could be high.</td>
</tr>
<tr>
<td>4</td>
<td>Low</td>
<td>Low</td>
<td>The occurrences of the punctuation marks in both datasets could be low.</td>
</tr>
<tr>
<td>2</td>
<td>High</td>
<td>Low</td>
<td>The underlying reason for a punctuation mark count difference between dataset A and B need to be investigated.</td>
</tr>
<tr>
<td>3</td>
<td>Low</td>
<td>High</td>
<td></td>
</tr>
</tbody>
</table>

The sense of “low” and “high” in this figure is a rather subjective, relative term that one has to decide. Even when only two datasets are compared based on these possibilities, when using this strategy, a plausible explanation needed to be pursued in interpreting the punctuation mark frequency results.

The initial assumption was that a frequency count of punctuation patterns could easily indicate the role of punctuation marks. This, however, turned out to be a naive way of viewing comparative analysis. The reason was that sometimes a punctuation mark had multiple roles in a text. However, it did call for more in-depth analysis in most cases. Also, identifying the scenarios as pointed out Table 4 was useful in a sense that it served as an entry point to a deeper understanding of the different functions and types of punctuation marks.
In sum, the frequency count strategy used in this study was an inductive approach. The inductive approach is so-called a bottom up approach. Considering that there was not a sufficient amount of prior research that focused on punctuation normalization issues, as a result of applying punctuation frequency analysis, some of the more meaningful categories can emerge by using the inductive approach as Thomas (2004) pointed out. Because an inductive approach was employed in this study, no formal hypothesis was formed initially. A category development (e.g., punctuation normalization types), on the other hand, was made through a combination of deductive and inductive punctuation frequency analysis steps, partly relying on the previous works such as Jones (1997).

**Punctuation Normalization Types**

To examine punctuation mark normalization, it was necessary to carefully identify normalized terms in conjunction with types of punctuation normalization. The previous procedures focused mainly on discovering punctuation patterns. Unlike the previous procedures, this procedure attempted to answer the research questions more directly. Essentially, this was an observational analysis based on punctuation normalization phenomena and punctuation patterns. The major difference between normalized and un-normalized terms is that in the normalized case, terms typically undergo a punctuation mark normalization process. On the other hand, un-normalized terms refer to text in the original state, having punctuation marks intact. In effect, the normalization process itself was categorized into a number of possible types. Qualitatively analyzing these types was expected to help in answering the research questions.

In the context of normalizing punctuation marks, the discovered punctuation patterns needed to address the following questions were:
• What type of punctuation normalization operation could be applied?
• How did the token change due to applying the punctuation normalization operations?
• In the context of text retrieval, what were the effects of normalizing different types of punctuation patterns?

In examining the punctuation normalization problem, it was also important to note that the procedure relied on a statistical account of various punctuation patterns that were identified in the previous procedures. That is, qualitatively, punctuation mark normalization was examined with a collection of facts that was mostly based on the interpreted statistical results of punctuation patterns. At this point, previous works on punctuation mark frequency analysis such as Jones’ work (1997) had been taken into account as well.

One of the tasks was to distinguish punctuation patterns that depended upon a specific domain or genre. The frequency count procedures might indicate the domain- or genre-specific characteristics. In turn, some of these frequency count results helped in defining categorization for punctuation normalization rule types and normalized punctuated mark patterns. However, identifying characteristics of normalized punctuation pattern required a holistic approach that considered several related factors.

Various factors were considered in measuring the similarity between potentially relevant terms. For example, we could have two terms where one is normalized and the other is un-normalized. Let us assume \textit{B.C.} as a search query term, and \textit{BC} as a potential term that should be matched. We can clearly see that these two character strings are orthographically similar. Considering this type of situation, we considered different schemes to assess the orthographic similarity of search query term and potentially matchable term.

To this end, the concept of orthographic similarity had led to an orthographic difference measurement. Devising formulas to measure orthographic differences had to consider textual
attributes such as alphabetical string differences and punctuation mark string differences. An orthogonal distance formula was considered since it was useful in terms of understanding an orthographic difference between two terms.

Types of Search Queries and Punctuation Normalization Effects

Queries could be categorized into two basic types: exact search and approximate search. A search query that is used in an exact search is referred to as an exact search query. As the name suggests, an exact search requires matching strings in a dataset against the terms in search query exactly as they appear. Orthographically, all the spellings, individual letters, and punctuation marks need to be identical.

On the other hand, for an approximate search method, a text retrieval system needs to closely match the terms based on a search query. A search query that is used to perform approximate searches is referred to as an approximate search query. A more elaborate arrangement was necessary for an approximate search query type. For example, a regular expression search (Friedl, 2006), which normally represented by a wild card * operator, could be used to match any types of characters. Wild card usually means to match any number of characters 0 or more times. For example, B*C* could match any of the following:

- B.C.
- B.C.;
- BC
- BCDE
- B-C-D
- BEACON
In this case, the letter $B$ is matched first. Then, any number of characters can be matched as long as the letter $C$ is matched at some point after. Optionally, after this point, any number of characters can be matched again. For another example, if the search query term is expressed as *$B$*, none of the above list would be matched since the search query states that all terms that end with the letter $B$ needs to be matched. The shortcoming of applying such a wild card operator was that too many unrelated items could be retrieved as a search result.

Finally, we need to consider the length of search query. Considering various domains, it is difficult to know the average length of a search query. If we consider only the Web environment, previous studies indicate that search queries are generally short as Gabrilovich et al. (2009) pointed out. They estimated that the average length of a search query is between 2.4 and 2.7 words.

*The Normalization Demonstration*

Performing text retrieval demonstrations could provide additional insights to punctuation normalization issues. Assuming that the previous procedures provided sufficient grounds for identifying punctuation patterns, the normalization demonstrations were conducted as a means to test specific cases involving search terms. To this end, the main purpose of this procedure was to find the effects of the normalization in the context of text retrieval by running different type of search query procedures and comparing their search results.

After examining the basic token types, hypothetical search queries had to be formulated to use in the demonstration. In terms of finding answers to the proposed research questions, demonstrations with hypothetical search terms were used. A hypothetical search query is a “what-if” type of query that was useful for investigating the effects of punctuation normalization.
The purpose of running these types of queries was to find out specific types of punctuation normalization phenomena.

Table 5 shows basic search criteria for comparing search query results. The two-dimensional table shows basic test criteria for examining punctuation normalization effects. The simplest normalization demonstration was to compare and analyze two fundamental search criteria: *normalized text* versus *un-normalized text*. By comparing the search results based on these two criteria, search queries that utilized punctuation marks could be identified. Initially, the study relied on examining normalized results and alternatively normalized results on a comparative basis. In this manner, multiple normalization options or rules could be compared to each other.

As shown in Table 5, the search queries were divided into two basic types: *Un-Normalized Search Query* and a *Normalized Search Query*. Using these two searches as our basic query, Gutenberg and Usenet datasets search results were repeatedly compared. Additionally, we had to consider other punctuation normalization demonstration parameters (e.g, exact vs approximate search queries) as we mentioned earlier.

Table 5

<table>
<thead>
<tr>
<th>Basic Criteria for Analyzing Search Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Un-normalized Search Query</td>
</tr>
<tr>
<td>Normalized Search Query</td>
</tr>
</tbody>
</table>

In most cases, the issue was explored qualitatively to better understand the punctuation normalization phenomena. For practical reasons, the number of demonstration trials that
involved a different type of searches had to be limited. Although many interesting punctuation patterns were expected to be found, only demonstrations that seemed to be most important were carried out.

For now, without going into the specifics of the demonstration results, for preliminary discussion purposes, some of the demonstration interpretations can be discussed. At the time of designing the procedures, it was believed that the results from the demonstrations could indicate common semantic problems that are associated with punctuation normalization. In interpreting quantitative results, other frequency counts had to be examined collectively. Also, the relationship between similar punctuated character strings had to be examined in detail. The pertinent question in terms of punctuation marks, or in terms of orthography as a whole, was how close are the terms in a search query to the terms in the datasets? Collectively, an attempt was made to discover the semantic effects of normalizing punctuation marks.

Methodological Limitation

One of the limitations of this study was that we relied only on the two datasets as primary sources of data. Although the datasets reflect natural language that are used in the real-world, generalization based on only two datasets represent limited types of genres (e.g., classic novel, children, etc.). There are many different types of domains (e.g., bibliographical text, SMS messaging, etc.). All of these domains may reflect distinctive punctuation mark usages.

In this sense, punctuation patterns that are domain- or genre-specific are generalizable only in the context of a specific domain. Although the dataset was large, the size of the data potentially was a relevant issue due to the fact that the majority of punctuation patterns appeared infrequently in the dataset. This was an indication that other datasets could contain different
types of frequently or infrequently used punctuation patterns that might be useful in other special circumstances.

Also, qualitatively judging relevance was based on this researcher’s analytical reasoning alone; users were not involved as part of the study. Because of this, it was sometimes difficult to seek absolute certainty in estimating the contextual sense of the character strings in the datasets, especially when they were normalized.

Lastly, even with identified demonstration types, in majority cases, conducting a larger number of demonstrations were not feasible for practical reasons. There was a clear limitation to the number of tests that could be performed. Punctuation normalization demonstrations had to be significantly limited because of time and resources.

Chapter Summary

This chapter presented methodological details and research design issues. With the overarching goal of improving text retrieval practices, the challenge was to address punctuation normalization issues in a holistic and systematic manner. This chapter presented six detailed procedures of the study. The procedures were: tokenization, frequency counts, additional punctuation, punctuation normalization types, types of search queries and punctuation normalization effects, and the normalization demonstrations. All of the procedures were designed to find implications of punctuation normalization from a text retrieval viewpoint.

The first part of this research could be labeled as punctuation frequency analysis. The methodology largely relied on the results from the punctuation frequency analysis. The punctuation frequency analysis was based on the sampled datasets from two corpora: the Gutenberg Project archive and the Usenet Newsgroup archive. The punctuation frequency
analysis procedures focused for the most part on discovering generalizable punctuation patterns and punctuated tokens. The particular advantage of the punctuation frequency analysis was that various roles of punctuation marks could be characterized and quantified. From this study’s view, qualitatively determining punctuation patterns and the effects that they have on text retrieval were necessary.

Based on the result from the first three procedures, the latter procedures focused on analyzing the effects of punctuation normalization. It included an appropriate search and retrieval demonstration. Through the demonstration, we attempted find additional implications of normalizing punctuation marks. Some of the procedural details were highlighted in this chapter. This chapter ended with the note on the methodological limitations.
CHAPTER 4
THE PUNCTUATION FREQUENCY ANALYSIS

This chapter describes the results of the analysis of punctuation marks and their related categories based on the method. Frequency analysis was applied based on affix types: prefix, infix, and suffix. For this study, identifying different types of punctuation marks and punctuation patterns was an important pre-requisite to observing punctuation normalization phenomena. Once patterns were identified, additional useful punctuation mark types were further identified. Particularly based on the locality of punctuation marks, the focus of this analysis were to generalize punctuation marks behavior for the purpose of characterizing the effects of punctuation normalization.

The descriptive statistical approach, relying on frequency count specifically, was used to quantify and explain the pattern of punctuation marks and punctuated tokens. The approach taken was that through this initial analysis of punctuation marks, foundational knowledge related to punctuation normalization could be developed. This chapter covers the first part of the study which utilized a punctuation frequency analysis of the two datasets. Three major punctuation frequency analysis procedures in this chapter are the following:

- Tokenization
- Frequency counts and punctuation mark types
- Additional types of punctuation

This chapter is organized in such a way that the topics discussed roughly correspond to the punctuation frequency analysis procedures that were outlined in the previous chapter.
Tokenization

Tokenization is the first punctuation frequency analysis procedure of this study. Tokenization based on blank space was examined initially. Other options were considered after investigating blank space based tokenization. Although tokenization seemed to be a simple text processing procedure in the beginning, it became a basis for finding further issues related to punctuation normalization in this study. Tokenization based on a blank space was a trivial process and straightforward. In Unix like systems, a simple command `tr` could be used to transform inputs to an output. This command was used to translate a space to a newline is shown in (4).

\[ \text{tr ' ' 'n'} \]  

(4)

The first two apostrophes indicate that there is a blank space. A character length blank space between the first two apostrophes `' ' `is necessary. The next element `\'n\' `stands for a newline. The above `tr` command was used to replace all occurrences of the blank spaces with a new line `\n`. Another alternative attempt was to use a word-based tokenization. In such a case, a chain of tokens could be regarded as a single word. Since a word has to be precisely defined, this was a separate topic of its own. If a blank space was used as a basis for tokenization, the result would be more consistent and comparable for future studies. Nevertheless, using a blank space tokenization procedure, a single punctuation mark or a string of punctuation marks could be produced as a result of tokenization.

In defining punctuation mark types, notice that infix-, prefix-, and suffix-punctuation marks only refer to the specific type of affix-based punctuation marks. Based on the above punctuation mark affixes, three primitive token types can be defined as the following:

- A token that contains one or more prefix-punctuation marks is defined as a prefix-punctuated token.
• A token that contains one or more infix-punctuation marks is defined as an *infix-punctuated token*.

• A token that contains one or more suffix-punctuation marks is defined as a *suffix-punctuated token*.

In addition, other related terms needed to be established for the purpose of analyzing the punctuation marks. They were the following:

• *Punctuated token* denotes a token that contains any type of punctuation mark.

• *Punctuation-exclusive token* denotes a stand-alone punctuation mark that is non-affixed. That means a punctuation-exclusive token comprises only punctuation marks and does not contain alphanumeric characters.

The above defined affix-based token categories were useful for observing punctuation mark normalization phenomena since they were used to understand the changes that occur with normalized tokens as a whole. Figure 7 is an example that shows different types of tokens. Note that a token may fall into more than one punctuated token category. As in case of the punctuated token *B.C.*, the token is referred to as infix-suffix punctuation token since it could be categorized as both infix punctuated token and suffix punctuated token. Thus, in this figure, a punctuated token could be any one of the following: prefix-punctuated token, infix-punctuated token, suffix-punctuated token, infix-suffix punctuated token, and punctuation-exclusive token.

*Figure 7.* An example showing different types of tokens.
Frequency Counts and Punctuation Mark Types

In this study, *frequency counts and punctuation mark types* is the second procedure of the punctuation frequency analysis. Initially, the use of frequency counts allowed us to discover punctuation mark characteristics. Through the process of conducting frequency counts based on affix-based tokens, we were able to identify different types of punctuation marks. The section describes the results.

*Basic Statistics of Punctuation Marks*

In the previous chapter, we described the two separate datasets that were sampled from each corpus. In the end, each dataset purposefully contained an approximately equal number of tokens. Because a simple blank space was used to tokenize character strings, any sequence of character that is surrounded by a blank space could be considered as a token. A token could be comprised of alphanumeric characters, numeric characters, punctuation marks or combinational mixture of all these characters.

The basic statistics of tokens based on different types of punctuation marks are shown in Table 6. As shown in this table, each dataset shows approximately 32 million tokens. For the purpose of this study, each corpus was reduced to this size as discussed in the previous chapter. Table 6 also describes the frequency counts based on punctuation mark affix based (infix, prefix, and suffix) tokens and non-affix based (punctuation-exclusive) tokens.

The *grep* command was mainly used to match a given text pattern against the input files. Other commands that were used included *sed*. Sometimes UNIX shell scripting, containing multiple combinations of commands, was used to obtain the frequency counts. See Appendix C for more detailed descriptions of the *grep* commands that were used in this study.
For the purpose of conducting different types of frequency count, an equation to calculate the relative percentage differences between two frequency counts was necessary. It was a useful scheme to detect common and uncommon punctuation patterns. Since there were two datasets A and B, then equation (5) could be established as a basis for finding a relative percentage difference.

\[
\frac{\text{MAX}(A, B) - \text{MIN}(A, B)}{\text{MAX}(A, B)}
\]

Relative Percentage Difference = \(\frac{\text{MAX}(A, B) - \text{MIN}(A, B)}{\text{MAX}(A, B)}\)  
(5)

Table 6

Basic Statistics of Punctuation Marks

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Usenet</th>
<th>Gutenberg</th>
<th>Relative % Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Count of Tokens</td>
<td>32,255,176</td>
<td>32,271,957</td>
<td>0.05%</td>
</tr>
<tr>
<td>Total Count of Punctuated Tokens</td>
<td>4,429,504 (13.73%)</td>
<td>5,820,177 (18.03%)</td>
<td>23.89%</td>
</tr>
<tr>
<td>Total Count of Infix-Punctuated Tokens</td>
<td>852,057 (2.64%)</td>
<td>822,507 (2.55%)</td>
<td>3.47%</td>
</tr>
<tr>
<td>Total Count of Prefix-Punctuated Tokens</td>
<td>539,177 (1.67%)</td>
<td>690,334 (2.14%)</td>
<td>21.90%</td>
</tr>
<tr>
<td>Total Count of Suffix-Punctuated Tokens</td>
<td>3,232,149 (10.02%)</td>
<td>4,655,397 (14.43%)</td>
<td>30.57%</td>
</tr>
<tr>
<td>Total Count of Punctuation-Exclusive Tokens</td>
<td>274,128 (0.85%)</td>
<td>146,087 (0.45%)</td>
<td>46.71%</td>
</tr>
</tbody>
</table>

Although the type of language found in the Usenet dataset was much more informal than the type of language found in the Gutenberg dataset, the frequency count of infix-punctuation marks was remarkably similar. Between the two datasets, the relative percentage difference between the two sets was only 3.47%. The greater difference was found in the suffix-punctuated tokens frequency occurrences. The total counts were considerably higher in the Gutenberg
dataset. The greatest difference was found in the frequency count of punctuation-exclusive tokens.

As shown in this table, for each corresponding dataset, the percentage value indicated inside the parenthesis is its relative percentage against the total token counts. For example, for the Usenet dataset, the infix-punctuated token count is 852,057, which is 2.64% of 32,255,176. Similarly, for the Gutenberg dataset, the infix-punctuated token count is 822,507, which is 2.55% of 32,271,957. When compared, Table 6 indicates that the relative percentage difference is only 3.47%, which is lower to other relative percentage difference in the same column.

Overall, the total frequency counts of punctuation marks indicate that large amounts of punctuation marks exist in the text in one form or another. By comparison, the suffix-punctuation marks occur more times than the infix and prefix-punctuation marks. The most logical, reasonable explanation for this result is that inter-lexical punctuation marks such as a period and a comma often occur at the end of each character string.

Note that punctuation marks can occur in a combination. Thus, one character string can belong to more than one category. For example, the character string \textit{B.C.} would be categorized as both infix- punctuated token and suffix-punctuated token. Also, it is common for punctuation marks to occur in a sequence. In such cases, the punctuation marks that occur in the outer affix-based positions take the precedence in categorizing the punctuation marks. To illustrate, because of the two plus signs ++ in the suffix position, the character string \textit{C++} was categorized as a suffix-punctuated token rather than an infix-punctuated token.

For each dataset, each punctuated token was retrieved by using the \textit{grep} command (6).

\texttt{grep -c '^[^0-9a-zA-Z]*$' filename} 

(6)
In this case, the *filename* is simply a name of the file that contains a token list based on a blank space. The `grep` command inside the bracket was used to match only one of more non-alphanumeric characters that ends with non-alphanumeric characters. The `-c` is a UNIX command option that was used to perform the frequency count. Then, the frequency count list was exported to a spreadsheet for analysis purposes.

The relative percentage difference of punctuation-exclusive tokens between two datasets was moderately high, yielding a 46.71% difference. Once again, source-independent punctuation marks have the delimiting and separating roles according to Jones (1997). Since most source-independent punctuation marks are in the prefix and suffix positions, it prompted a investigation of punctuation-exclusive tokens in-depth. Later in this chapter, we will discuss what the frequency count of punctuation-exclusive tokens implies.

**Infix-Punctuated Token**

The types of infix-tokens found in the dataset varied greatly. As shown in Table 7, the frequency count of individual infix-punctuated tokens varied according to the types of infix-punctuated tokens. Appendix D shows more complete listing of commonly found infix-hyphen tokens. As previously noted, there was a wide range in the relative percentage difference. A lower relative percentage difference indicates that the two datasets have a similar amount of infix-punctuated tokens. In contrast, a higher relative percentage difference indicates that the amount of infix-punctuated tokens found in the two datasets differ greatly.
### Table 7

**Top 20 Infix-Punctuated Tokens (Gutenberg to Usenet)**

<table>
<thead>
<tr>
<th>Infix-Punctuated Token</th>
<th>Gutenberg</th>
<th>Usenet</th>
<th>Relative % Difference</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>z’z</td>
<td>273,112</td>
<td>445,472</td>
<td>38.69%</td>
<td>Apostrophe ASCII code dec. # 39</td>
</tr>
<tr>
<td>z-z</td>
<td>145,903</td>
<td>112,948</td>
<td>22.59%</td>
<td></td>
</tr>
<tr>
<td>z--z</td>
<td>132,745</td>
<td>3,576</td>
<td>97.31%</td>
<td></td>
</tr>
<tr>
<td>z-z,</td>
<td>36,160</td>
<td>8,862</td>
<td>75.49%</td>
<td></td>
</tr>
<tr>
<td>z. --z</td>
<td>25,435</td>
<td>29</td>
<td>99.89%</td>
<td></td>
</tr>
<tr>
<td>&quot;z'z</td>
<td>24,282</td>
<td>8,828</td>
<td>63.64%</td>
<td></td>
</tr>
<tr>
<td>z-z.</td>
<td>17,735</td>
<td>7,949</td>
<td>55.18%</td>
<td></td>
</tr>
<tr>
<td>z.--z</td>
<td>9,647</td>
<td>7</td>
<td>99.93%</td>
<td></td>
</tr>
<tr>
<td>z.z.</td>
<td>6,837</td>
<td>13,032</td>
<td>47.54%</td>
<td></td>
</tr>
<tr>
<td>z’z.</td>
<td>6,601</td>
<td>2,950</td>
<td>55.31%</td>
<td></td>
</tr>
<tr>
<td>z--z,</td>
<td>6,346</td>
<td>126</td>
<td>98.01%</td>
<td></td>
</tr>
<tr>
<td>z-z-z</td>
<td>5,288</td>
<td>7,827</td>
<td>32.44%</td>
<td></td>
</tr>
<tr>
<td>z.z</td>
<td>4,017</td>
<td>27,614</td>
<td>85.45%</td>
<td></td>
</tr>
<tr>
<td>z-z;</td>
<td>3,538</td>
<td>285</td>
<td>91.94%</td>
<td></td>
</tr>
<tr>
<td>z--z’z</td>
<td>3,382</td>
<td>182</td>
<td>94.62%</td>
<td></td>
</tr>
<tr>
<td>z’z.</td>
<td>3,245</td>
<td>4,558</td>
<td>28.81%</td>
<td></td>
</tr>
<tr>
<td>z--z--z</td>
<td>2,903</td>
<td>23</td>
<td>99.21%</td>
<td></td>
</tr>
<tr>
<td>z.z.z.</td>
<td>2,885</td>
<td>1,658</td>
<td>42.53%</td>
<td></td>
</tr>
<tr>
<td>z.z</td>
<td>2,728</td>
<td>10,325</td>
<td>73.58%</td>
<td></td>
</tr>
<tr>
<td>z-z--z</td>
<td>2,356</td>
<td>18</td>
<td>99.24%</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The letter *z* was used to replace all matching instances of alphanumerical characters.

As shown in Table 7, to properly represent infix-punctuation marks, the letter *z* was used to replace all alphanumerical characters. For example, by using the *z* notation, both *she’ll* and *O’Brien* will be represented as *z’z*, which is the top ranked infix-punctuated token. This means that single quote was the most frequently used infix-punctuation mark. The frequency count of this infix-punctuated token *z’z* was almost twice the number of the next top ranked infix-punctuated token *z-z*. As shown in this table, some patterns share same infix-punctuation mark
but can be counted in different ways. For instance, the punctuation pattern \( z'z \) and \( \text{"z'z"} \) have a common infix-punctuation mark which is the apostrophe \'.

The reason this infix-frequency count shows a prefix punctuation mark such as \( \text{"z'z"} \) is that the remaining affix-based punctuation marks were not normalized before counting them. The remaining affix-based punctuation marks in this case are the prefix- and suffix-punctuation marks. There were two ways affix-based punctuation marks could be counted:

- Without normalizing remaining punctuation marks
- With normalizing remaining punctuation marks

In this case, we conducted punctuation mark frequency count without normalizing remaining punctuation marks. For this reason, prefix-punctuation marks such as a double quote \" were not normalized prior to conducting a frequency count.

Table 8 was generated by reversing the order of comparison. This table shows top 20 infix-punctuation mark tokens when the Gutenberg dataset is compared to the Usenet dataset. Since the top rank of the Usenet dataset is compared to the Gutenberg dataset in Table 8, new infix-punctuated tokens are listed in this table. As a result, the punctuation-token \( z\ldots\ldots z \) was not listed in the previous table, but it is now listed in Table 8.

This procedure has both an advantage and a disadvantage. The advantage is that more variations of remaining punctuation marks can be recognized. If the remaining parts are normalized then such a pattern would not be exposed in the list. The disadvantage is that, for the frequency count of punctuation marks, other variations including the remaining punctuation marks might not show up in this top 20 frequency count list. For example, hypothetically the infix punctuation mark \( z'z \) could be in the top 20 list, but \( z'z! \) might not show up in the list. However, most of the frequently used patterns are displayed in this list. Considering the size of
the distribution of frequency counts, some of the small unnoticed variation might not be considered as a major factor at this point.

Table 8

Top 20 Infix-Punctuated Tokens (Usenet to Gutenberg)

<table>
<thead>
<tr>
<th>Infix-Punctuated Token</th>
<th>Usenet</th>
<th>Gutenberg</th>
<th>Relative % Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>z'z</td>
<td>445,472</td>
<td>273,112</td>
<td>38.69%</td>
</tr>
<tr>
<td>z-z</td>
<td>112,948</td>
<td>145,903</td>
<td>22.59%</td>
</tr>
<tr>
<td>z.z</td>
<td>27,614</td>
<td>4,017</td>
<td>85.45%</td>
</tr>
<tr>
<td>z/z</td>
<td>21,281</td>
<td>821</td>
<td>96.14%</td>
</tr>
<tr>
<td>z.z-z</td>
<td>13,032</td>
<td>6,837</td>
<td>47.54%</td>
</tr>
<tr>
<td>z.zz</td>
<td>10,325</td>
<td>2,728</td>
<td>73.58%</td>
</tr>
<tr>
<td>z;z</td>
<td>10,273</td>
<td>298</td>
<td>97.10%</td>
</tr>
<tr>
<td>z...z</td>
<td>9,058</td>
<td>35</td>
<td>99.61%</td>
</tr>
<tr>
<td>z-z,</td>
<td>8,862</td>
<td>36,160</td>
<td>75.49%</td>
</tr>
<tr>
<td>&quot;z'z</td>
<td>8,828</td>
<td>24,282</td>
<td>63.64%</td>
</tr>
<tr>
<td>z-z.</td>
<td>7,949</td>
<td>17,735</td>
<td>55.18%</td>
</tr>
<tr>
<td>z-z-z</td>
<td>7,827</td>
<td>5,288</td>
<td>32.44%</td>
</tr>
<tr>
<td>z'z.</td>
<td>4,558</td>
<td>3,245</td>
<td>28.81%</td>
</tr>
<tr>
<td>z---z</td>
<td>3,576</td>
<td>132,745</td>
<td>97.31%</td>
</tr>
<tr>
<td>z/z/z</td>
<td>3,547</td>
<td>33</td>
<td>99.07%</td>
</tr>
<tr>
<td>z.z.z</td>
<td>3,283</td>
<td>1,036</td>
<td>68.44%</td>
</tr>
<tr>
<td>z'z.</td>
<td>2,950</td>
<td>6,601</td>
<td>55.31%</td>
</tr>
<tr>
<td>z(z)</td>
<td>2,828</td>
<td>170</td>
<td>93.99%</td>
</tr>
<tr>
<td>$z.z</td>
<td>2,746</td>
<td>270</td>
<td>90.17%</td>
</tr>
<tr>
<td>z._</td>
<td>2710</td>
<td>47</td>
<td>90.17</td>
</tr>
</tbody>
</table>

| Mean                   | 70.54% |

It was possible to obtain an exact count of infix apostrophe only by simply adding all the variations of remaining punctuation marks. It was also possible to perform the frequency count in both ways: without normalizing remaining punctuation marks and with normalizing remaining punctuation marks. Yet, performing frequency counts both ways would be more time consuming and unnecessarily complicated for our purposes of analyzing punctuation marks.
In Table 8, the particular order of datasets needs a further explanation. In this case, the Gutenberg dataset is compared to the Usenet dataset. In Table 7, in contrast, the order of the datasets is “Gutenberg” first and “Usenet” second. The Gutenberg dataset is shown on the left side and the Usenet dataset is shown on the right side. This means the ranking of top infix-punctuated tokens for this table is based on the Gutenberg dataset. For the each type of infix-punctuated tokens that were found in the Gutenberg dataset, frequency count was also performed against the Usenet dataset. Since the list does not include the top 20 infix-punctuation tokens of the Usenet dataset, some of the top infix punctuated token in the Usenet dataset do not show up this list.

Another generated table that shows the top 20 infix-punctuation token for the Usenet dataset was desirable if we want to see the differences in greater detail. That means we would have to reverse the order of comparison. In this manner, the top most frequently used infix- tokens in the Usenet dataset could be shown first on the left side instead of right side.

As can be seen in Table 8, 8 of the 20 top ranked infix-punctuation marks had a 90% or higher relative percentage difference. The mean value was calculated by adding the relative percentage difference values and dividing by the total number of infix-punctuation patterns. In Table 8, when the Gutenberg dataset was compared to the Usenet dataset, the average of the relative percentage difference was 70.54%. By reversing the order, when the Gutenberg dataset was compared to the Usenet dataset, the average of relative percentage differences was 70.07%. Calculated either way, there was not a great difference in the mean of relative percentage difference. From this table, the most common punctuation pattern between two sets can be derived based on the relative percentage difference alone.
Table 9, on the other hand, shows only the top 4 infix-punctuated tokens with the least relative percentage differences. Table 9 was extracted from Table 8. In this table, notice that relative percentage is relatively low for all listed infix-punctuated token. The infix-punctuation mark that shows a lowest relative percentage difference is the infix-apostrophe $z'z$. The low relative percentage difference indicates that a frequency count difference was minimal. However, the low relative percentage difference does not necessary indicate that the role of punctuation marks in one dataset is equivalent to another dataset. In this case, we can only state that the quantity of hyphens (ASCII code decimal #45) is relatively similar between the Usenet dataset and Gutenberg dataset.

Furthermore, notice that index #1 $z'z$ and index #4 $z'z$. show different punctuation patterns due to ending period in index #4. However, both have a hyphen as an infix-punctuation mark. Because of this reason, when counting a specific affix type, its variations must be combined based on the particular affix type since remaining punctuation marks are not normalized. In this case, frequency counts of both #1 and #4 needed to be combined if we are attempting to count the infix-apostrophes.

Table 9

Top 4 Infix Tokens with Least Relative % Differences

<table>
<thead>
<tr>
<th>Index</th>
<th>Infix-Punctuated Token</th>
<th>Usenet Newsgroup Dataset</th>
<th>Gutenberg Dataset</th>
<th>Relative % Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$z'z$</td>
<td>445,472</td>
<td>273,112</td>
<td>38.69%</td>
</tr>
<tr>
<td>2</td>
<td>$z-z$</td>
<td>112,948</td>
<td>145,903</td>
<td>22.59%</td>
</tr>
<tr>
<td>3</td>
<td>$z-z-z$</td>
<td>7,827</td>
<td>5,288</td>
<td>32.44%</td>
</tr>
<tr>
<td>4</td>
<td>$z'z.$</td>
<td>4,558</td>
<td>3,245</td>
<td>28.81%</td>
</tr>
</tbody>
</table>
From this point forward, a new naming convention, *affix type-punctuation mark* or *affix type-punctuation pattern*, will be used. For example, infix-hyphen simply means the infix-punctuation mark in the token is the hyphen.

**Infix-Apostrophe**

Generally, apostrophes indicate contraction, possessives, and plurals. Typical grammar reference books such as Quirk et al. (1985) and Straus & Fogarty (2007) provide details on uses of the apostrophe. For now, this discussion will be on different types of apostrophes and their frequency counts. Referring back to Table 8, infix-apostrophes were the highest occurrence for both the Usenet dataset and the Gutenberg dataset. There were 445,472 instances of infix-apostrophe $'$z$ for the Usenet dataset, while in the Gutenberg dataset there were 342,936 instances of infix-apostrophe $'$z$.

Given the distinctive roles that the apostrophes play in English grammar, there were several ways to count apostrophes and examine the character strings that contain apostrophes. One possibility was to investigate the uses of apostrophe before the letter s. The $'$s indicates a possessive. The number of terms that end with $'$s could be easily obtained by using the grep command (7).

```
grep -c "s$" filename
```

(7)

Table 10 shows a frequency count of apostrophe-s $'$s and the punctuation pattern $'$z$ that exist in the Usenet dataset and the Gutenberg Dataset. There were 206,980 instances of apostrophe - s $'$s in the Usenet dataset and 168,821 instances of apostrophe-s $'$s in Gutenberg dataset. The relative percentage difference is 18.4%.
When the frequency count of apostrophe-s’s is compared to the punctuation pattern z’z, we see that apostrophe-s’s roughly accounts for about 50% of z’z punctuation pattern for both datasets. This result was a clear indication that substantial variations of apostrophe uses were still present in the datasets. For this reason, more in-depth frequency counts were needed to predict the specific uses of apostrophes.

Table 10

<table>
<thead>
<tr>
<th>Infix-Punctuated Pattern</th>
<th>Usenet Newsgroup Dataset</th>
<th>Gutenberg Dataset</th>
<th>Relative % Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>’s</td>
<td>206,980</td>
<td>168,821</td>
<td>18.4%</td>
</tr>
<tr>
<td>z’z</td>
<td>445,472</td>
<td>273,112</td>
<td>38.69%</td>
</tr>
</tbody>
</table>

Although an in-depth frequency analysis was not performed, some patterns of the infix-apostrophe token can be learned by observing Appendix E. This appendix displays the top 40 most frequently used infix-apostrophe tokens that appeared in the Gutenberg dataset. In contrast, Appendix G is a list of infrequent infix-apostrophe tokens in the Gutenberg dataset. In comparison, both lists show character strings that contain apostrophe-s’s. While the list in Appendix E contains mostly contractions, the list in Appendix G contains both possessives and contractions. It is evident that the Appendix G list contains more unique tokens which might be helpful in terms of retrieving relevant text. In the case of infix-apostrophes, infrequent punctuation patterns are likely to be associated with additional unique terms.

As shown in Appendix G, expressions such as Ne’er and ye’ve are not standard nor commonly used contractions. If capitalization is normalized and the apostrophe is removed, Ne’er and ye’ve could be rewritten as neer and yeve. An interesting observation that could be made was that without indexing internal apostrophe marks a user would have extreme difficulties
in searching these types of punctuated character strings. Here it is clear that *neer*, if normalized in this way, is semantically different from the original *Ne'er*. Similarly, the token *ye've* is semantically different from *yeve*.

Thus removing an apostrophe and combining remaining textual elements in this case was not a suitable punctuation normalization approach. Another contraction that was not in the list was *I'd*. In case of the term *I'd* is a contraction for *I would*. Similar to the above case, a user would not want to retrieve the term *Id* if *I'd* was used as a search query term.

In contrast, simply removing apostrophes and concatenating remaining sub-parts of the character string was desirable in some cases. For example, in the Usenet dataset, there were 29 instances of the character string *Shi’ite* and 53 instances of the character string *Shiite*. In this case, unless a user is aware of searching the term both ways, character strings with such typographic differences would not be retrieved by a single retrieval process using exact match search.

These examples are based on unique instances, and the need to retrieve text segments that contain such punctuated character string might be extremely rare. However, because the same type of infix-punctuation marks is used with other character strings in a text, it is easy to overlook the differences before actual punctuation normalization is performed. To what degree all of these instances have an accumulating effect in the end is a legitimate question to ask but difficult to estimate. Yet the example demonstrated how sensitive punctuation normalization may need to be and raised the question of punctuation mark roles in search queries. In the subsequent chapter, these issues are analyzed in more depth.
Infix-Hyphens and Dashes

Hyphenated character strings are very common in English. The majority of hyphenated character strings contain infix-hyphens. The frequency count showed various uses of hyphens and dashes. As will be discussed later, a distinct ASCII code is assigned to each of the characters. Typographically, however, hyphens and dashes may appear identical to users. Because of this, the fact that users may alternatively use hyphens in place of dashes in casual conversational writing should be taken into account. The implication for a frequency count result is that a careful examination of hyphenated character strings should be done prior to applying punctuation normalization procedures.

Let us refer back to Table 9. Hyphen (ASCII code decimal #45) was the second highest infix-punctuation mark that appeared in both datasets. There were 112,948 instances of infix-hyphenated tokens which appeared in the Usenet dataset, and 176,884 instances of infix-hyphenated tokens which appeared in the Gutenberg dataset. There were almost twice as many infix-apostrophe tokens as there were infix-hyphen tokens in the Gutenberg dataset. There were almost four times more infix-apostrophe tokens as infix-hyphenated tokens in the Usenet dataset. Appendix H is the top ranked hyphenated token list for the Gutenberg datasets. When Appendix E is compared to Appendix H, the occurrences of top frequently used infix-hyphen tokens were much lower than the occurring of the top frequently used infix-apostrophe tokens.

The infix-hyphen indicates that two strings of alphabetical characters exist on each side. One of the most common uses of hyphen is to connect different character strings to a new multipart character string. In Appendix H, two types of hyphenated tokens can be realized from this list:

1. Compounded word tokens that comprise of free morphemes
2. Affix-based word tokens that contain at least one bound morpheme (affix)

A free morpheme is a single linguistic unit which carries meaning, whereas bound morpheme carries meaning but depend on another free morpheme. That is, a free morpheme can stand by itself, whereas bound morpheme cannot stand alone as a word.

In the case of a compounded word token, a token such as health-care can be decomposed into two independent free morphemes: health and care. Each word can function as an independent word, but they have been combined together to form a new word: health-care. Jones (1997) referred to most of these types of words as inter-lexical words as mentioned in Chapter 2.

In contrast, in the case of the affix-based word tokens, at least one sub-component may be dependent upon another lexical component. For example, for the word re-read, re is considered as morpheme but bound by another morpheme read. The bound morpheme re cannot function as an independent lexical token. Appendix J is a random sample of words taken from the Gutenberg dataset. The dataset contained only two first letters with infix-hyphens. Because bound morphemes are usually short lengthwise, they are regarded as prefixed words. In contrast to multi-part words having an inter-lexical hyphen, Jones may have considered some of these hyphens as a sub-lexical punctuation mark. Thus, for different types of hyphenated words, one of the considerations for punctuation normalization was the type of morphemes.

Nevertheless, for a simple analysis case, infix-punctuated tokens that comprise two prefix characters were retrieved using the grep command (8)

\[
grep '^\^[a-zA-Z0-9][a-zA-Z0-9]-[a-zA-Z0-9]' \text{filename} \quad (8)
\]

As a result, 11,662 such tokens were retrieved. The produced list contains only the first two alphanumeric characters that precede the hyphen-infix-punctuation mark. Note that only
two characters were retrieved in this case. However, more combinations of alphanumeric letters can be analyzed and not just two alphanumeric characters. In any case, as indicated in Appendix J, most of these words, with the exception of the words such as *on* or *so*, contained bound morphemes. Normalization of an infix-hyphen in an affix-based token is necessary so that the token *co-worker* could be matched with either *co-worker* or *coworker*.

In addition, a character set variation was associated with hyphens. In the Gutenberg dataset, it was evident that certain hyphens were already normalized to a degree. However, in the Usenet dataset, hyphen-like characters that have an appearance of a normal hyphen were not normalized. The difference was negligible considering the total size of tokens that exist in the Usenet dataset. However, such a difference could cause a problem if it is overlooked. As shown in Table 11, a large number of conventional hyphens (ASCII code decimal #45) was found in both datasets. Although the overall count was relatively small, slight varying types of hyphens (ASCII code decimal #45 and ASCII code decimal #173) and dashes (ASCII code decimal #150 and ASCII code decimal #151) were found in the Usenet dataset. Note that the frequency counts in this table were calculated based on a total count of punctuation marks and not just an infix-punctuation mark frequency count.

Table 11

*Hyphen and Dash Variations in the Usenet and Gutenberg Datasets*

<table>
<thead>
<tr>
<th>Punctuation Mark</th>
<th>DEC</th>
<th>OCT</th>
<th>Description</th>
<th>Usenet Dataset</th>
<th>Gutenberg Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>173</td>
<td>225</td>
<td>soft hyphen</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>45</td>
<td>055</td>
<td>hyphen</td>
<td>269,572</td>
<td>408,934</td>
</tr>
<tr>
<td>–</td>
<td>150</td>
<td>226</td>
<td>en dash</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>—</td>
<td>151</td>
<td>227</td>
<td>em dash</td>
<td>19</td>
<td>0</td>
</tr>
</tbody>
</table>
On the other hand, the hyphen, which is represented by the ASCII code decimal #173, is a soft hyphen. A soft hyphen is generally inserted by software. Often some words end up at the end of line, and a soft hyphen is used to split the word onto two lines of text. There were 5 instances of the soft hyphen in the Usenet dataset, and no soft hyphen was found in the Gutenberg dataset. Refer to Appendix A again for the complete list of punctuation marks that are considered by this study.

To generate such hyphen variations as shown in Table 11, the grep command (9) could be used to calculate the occurrences of all tokens that contain a hyphen.

\[
grep -c "$055' filename \tag{9}\]

Here, 55 is an octal value for the ASCII character hyphen. The command (9) was used to generate Table 11. By using the same type of command but with different octal values, the frequency counts of Usenet dataset were obtained for the soft hyphen, en dash, and em dash. By searching for octal value 227 and using the above command (9) but without -c option, we can examine the actual text in the Usenet dataset that contain the em dash.

This type of character set related punctuation normalization problem showed that the frequency count strategy was feasible in identifying a punctuation mark that is represented by different character set code. The frequency count demonstrated that the potential problems that are associated with different character set related issues can be identified beforehand. Thus, it was important to realize the difference in character set and typographic representation before attempting to compare and analyze the punctuation mark differences. By performing the frequency count of infix-punctuation marks, we can see the detailed issues related character sets and typographic representation.
At this point, we further examined character encoding scheme differences. We discovered that the Usenet dataset contained non-ISO extended-ASCII text, meaning that the file contained some very long lines, line terminators, with escape sequences that are not typically associated with extended ASCII characters. While the ASCII code refers to the original dataset that is represented by characters having the decimal numbers 0 to 127, the extended ASCII table refers to the characters having the decimal numbers from 128 to 255. See Appendix A for the discussion on ASCII codes.

Unlike the Usenet dataset, we confirmed that all of the files that we downloaded from the Gutenberg website were in the original ASCII format. This means that some characters such as the left quote “ and right quote ” would not be found in the Gutenberg dataset since these punctuation marks, unlike the apostrophe ’, are found only in the 8-bit extended ASCII table. A combined frequency count of less than 100 for the left quote “ and right quote ” was found in the Usenet dataset when we decided to examine the character set differences. Although the quantity of variations was small, these types of typographic varieties do indicate that careful attention should be made in dealing with punctuation marks. Figure 8 is a text segment that contained the em dash in the Usenet dataset. Note that em dashes are not infix hyphens. Although there were 19 lines that contained the em dash, only 16 lines are retrieved and shown here since some lines contained more than one em dashes. Not all em dashes spaced out evenly in the text. For example, line 2 contains an em dash that is not surrounded by blank spaces, suggesting that in a real world setting, a variety of hyphens and dashes can be expected.
CAVE CREEK, Ariz. — Arizona is a modern place, among the fastest-growing identity—and it is responsive to treatment. insisting that homosexuality is a psychological condition—not a fixed lifted a card — a king of hearts — and the crowd roared. Cave Creek had shuffled it — having removed the jokers — six times. "It’s a hell of a way to win — or lose — an election," Mr. McGuire said. Constitution — not D.C., as Democrats are trying to switch it permanently. Education for the Disadvantaged: $651 million Dept. of Education — Political power in America should remain A.C. — According to the ST. PETERSBURG — Flanked by St. Petersburg Mayor Rick Baker and Tampa candidate — can burnish their conservative credentials, knowing all the governors — some of whom are said to be eyeing White House bids in 2012 it also means governors like Sanford and Louisiana's Bobby Jindal — a the parent's claims — and backed years of science that found no risk. years when the federal dollars disappear — a worry also cited by Jindal — are putting their own interests first.

Figure 8. Text segment containing infix em dash.

**Infix-Slash**

Slashes had become widely used since the medieval period as researched by Parkes (1993). There is an interesting piece of history how the slash was used in the medieval period. Parkes pointed out that double slashes // are used to denote a long pause or the start of a new section of text. It appears that in modern days, slashes seem to have many uses in the English language. Wikipedia (“Slash (punctuation),” 2013) shows a fairly good list of slashes that are used in the more modern period. From inspecting the Usenet text that contains the slashes, we attempted to categorize some of the common uses in the Usenet dataset. These include:

- Programming language (e.g., // as comments in C++ language)
- Fractions (e.g., ¼)
- Dates (e.g., 1/1/2012)
- Alternatives or versus (e.g, stars/galaxies)
- Separate numbers  (e.g, 907/09400)
- Abbreviations (e.g., i/o)
• File locations (e.g., c://temp)

• Internet addresses (e.g., http://www.myaddress.com)

• “Per” or “to” type of convention in measurement (e.g., km/h)

Although an in-depth analysis for every case was not performed, comparing frequency counts of the slash revealed interesting aspects of the punctuation mark. As shown in Table 8, an infix-slash was heavily used in the Usenet dataset. There were 21,281 infix-punctuated tokens that contained a slash. Compared with the Usenet dataset, the Gutenberg dataset contained a considerably smaller number of infix-slash tokens. There were only 821 instances of infix-slash tokens in the Gutenberg dataset; the difference was 96.14%.

More slashes appeared to be used in abbreviating character strings in the Usenet dataset than in the Gutenberg dataset. To further probe into this characteristic, a frequency count was performed based on infix-slash tokens with one alphabetical character on each side. To further classify an infix-slash type, we will use the term single-letter infix-slash for token types that consists of two alphabetical characters: one before the slash and after the slash.

The frequency counts of single-letter infix-slash tokens for the Usenet dataset is shown in Appendix K. I/O was the most frequently used single-letter infix-slash token. One hundred sixty eight of such instances were found in the Usenet dataset. The token I/O usually stands for the expression input and output. This expression is often used in the application domain that deals with computers and technology. However, hypothetically I/O also could stand for I divide by O. Similarly, B/C could stand for B or C. The token B/C could also stand for B divided by C.

For another example, there were 91 instances of w/o. However, the meaning of w/o also depends on the context. Intuitively w/o is an abbreviated character string could be found easily in a text that contain an online type of conversations. While we can expect a user to specify such
terms in a search query, to provide the correct context of such terms, further analysis of abbreviated character strings were clearly required on the part of the system.

The command (10) is used for the purpose of identifying single-letter-infix-slash.

\[
grep -c '^[a-zA-Z]/[a-zA-Z]$' filename
\]

When the frequency count based on single-letter-infix-slash token was performed, a greater difference between the two datasets was found. A frequency count of single-letter-infix-slash tokens is shown in Table 12. The Usenet dataset had a total of 1,019 instances of a single-letter-infix-slash token. On the other hand, in the Gutenberg dataset, there were only 10 instances of single-letter-infix-slash tokens. Since even 1,019 counts is still a relative small number in comparison to the total token counts, this was another indication that abbreviations perhaps are more likely to contain periods than slashes. The total token counts from the datasets was approximately 32 million.

For the Usenet dataset, some of the frequency counts and their corresponding tokens for a single-letter-infix-slash token are shown in Appendix K. The most likely cause of the high frequency of infix-slash tokens and single-letter-infix-slash tokens is the fact that the Usenet dataset contains a modern, informal language. Because of this, abbreviations using slashes were more evident in the Usenet dataset.

Table 12

<table>
<thead>
<tr>
<th></th>
<th>Usenet</th>
<th>Gutenberg</th>
<th>Relative % Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1,019</td>
<td>10</td>
<td>99.02%</td>
</tr>
</tbody>
</table>

It is difficult to estimate how users might use some of these punctuated character strings in formulating a search query. Although some character strings might be less frequently used
than another does not necessarily indicate that they are any less frequently used as a search term. Nevertheless, for the Usenet dataset in particular, single-letter-infix-slash tokens were a punctuation pattern that was worth investigating. In short, the examples demonstrate that the frequency count is an effective strategy where the particular objective is a discovering a lexical pattern that is associated with punctuation marks.

**Infix-Periods**

There were a variety of non-standard uses associated with infix-periods. Comparing the two datasets, the occurrences of the infix-period were much less in the Gutenberg dataset than in the Usenet dataset. In the Usenet dataset, there were many incidences of computer related terminologies that contributed to the high number of period occurrences. Infix-periods are used commonly in specifying website addresses, file names, etc. Referring back to Table 8, there are only 4,017 instances of an infix-period \( z.z \) found in the Gutenberg dataset whereas 27,614 instances of infix-period \( z.z \) were found in the Usenet dataset. Table 8 includes frequency count of other variations of infix-periods (e.g., \( z.z \) and \( z.z.z \ ).

Overall, the result from the Usenet dataset showed a considerable number of infix-periods. An infix-period is most often sub-lexical because of the locality of the punctuation mark in a character string. The amount of infix-periods could be a legitimate indication of domain-specific related punctuation usage. For example, the character string \textit{alt.bigfoot} refers to a particular newsgroup in the Usenet. This type of infix-period usage is much more likely to appear in a computer and technology domain related corpus. Beyond this generalization, we could state that certain type of genre such as the Usenet dataset contains a vast number of infix-periods.
Table 13 shows the top 10 frequently used tokens with infix-periods in the Usenet dataset. As the table indicates most tokens comprised numbers that had an infix-period. The result seemed somewhat surprising since the first top tokens with infix-period were just numbers, without any alphabetical characters. In other character strings, a punctuated token such as `alt.bigfoot` did not appear in the top list.

Table 13

*Top 10 Tokens Infix-Period in the Usenet Dataset*

<table>
<thead>
<tr>
<th>Frequency Count</th>
<th>Token Containing Infix Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>304</td>
<td>1.5</td>
</tr>
<tr>
<td>286</td>
<td>2.5</td>
</tr>
<tr>
<td>275</td>
<td>2.0</td>
</tr>
<tr>
<td>223</td>
<td>3.5</td>
</tr>
<tr>
<td>197</td>
<td>5.0</td>
</tr>
<tr>
<td>178</td>
<td>10.5</td>
</tr>
<tr>
<td>160</td>
<td>0.5</td>
</tr>
<tr>
<td>155</td>
<td>10.4</td>
</tr>
<tr>
<td>154</td>
<td>3.0</td>
</tr>
<tr>
<td>150</td>
<td>1.0</td>
</tr>
</tbody>
</table>

To explore infix-periods even further, it was necessary to examine some of the lower ranked tokens that contained infix-periods. Figure 9 shows some of the least frequently used terms for the Usenet dataset. There were 9,778 occurrences of unique tokens that had an infix-period. The total count of infix-period tokens in the Usenet dataset was 27,614 as previously shown. In terms of percentage, 35.41% of the infix-period `z.z` had unique instances.

Since the infix-period token list that was produced was fairly long, only a snapshot is shown in this figure. As shown, this list is alphanumerically sorted. A certain amount of periods having “lexical dependency” can be observed from this list. For example, for tokens which start with the alphabet letter “p”, there were numerous incidences of tokens which referred to page...
numbers. On the other hand, for the tokens that start with the letter “r”, many tokens were in reference to Usenet newsgroup categories. These instances are shown in this figure.

<table>
<thead>
<tr>
<th>FREQ TOKEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
</tr>
<tr>
<td>…</td>
</tr>
<tr>
<td>NewsMax.com</td>
</tr>
<tr>
<td>Newsmax.com</td>
</tr>
<tr>
<td>NewsGuy.Com</td>
</tr>
<tr>
<td>newsgroup.s</td>
</tr>
<tr>
<td>Newsday.com</td>
</tr>
<tr>
<td>newscast.Sometimes</td>
</tr>
<tr>
<td>NewsAndOpinion.com</td>
</tr>
<tr>
<td>news.Least</td>
</tr>
<tr>
<td>News.Following</td>
</tr>
<tr>
<td>news.com</td>
</tr>
<tr>
<td>…</td>
</tr>
<tr>
<td>…</td>
</tr>
<tr>
<td>p.33</td>
</tr>
<tr>
<td>p.306</td>
</tr>
<tr>
<td>p.263</td>
</tr>
<tr>
<td>p.252</td>
</tr>
<tr>
<td>p.229</td>
</tr>
<tr>
<td>p.217</td>
</tr>
<tr>
<td>p.2</td>
</tr>
<tr>
<td>p.1991</td>
</tr>
<tr>
<td>p.179</td>
</tr>
<tr>
<td>p.163</td>
</tr>
<tr>
<td>p.15</td>
</tr>
<tr>
<td>…</td>
</tr>
<tr>
<td>…</td>
</tr>
<tr>
<td>rec.music</td>
</tr>
<tr>
<td>rec.juggling</td>
</tr>
<tr>
<td>rec.heraldry</td>
</tr>
<tr>
<td>Rec.Gambling</td>
</tr>
<tr>
<td>rec.equers</td>
</tr>
<tr>
<td>Rec.Climbing</td>
</tr>
<tr>
<td>rec.climber</td>
</tr>
<tr>
<td>rec.bloats</td>
</tr>
<tr>
<td>rec.bird</td>
</tr>
<tr>
<td>rec.bike</td>
</tr>
<tr>
<td>…</td>
</tr>
<tr>
<td>…</td>
</tr>
<tr>
<td>0.0000000000000</td>
</tr>
<tr>
<td>0.000000000000000000000000000000000001</td>
</tr>
<tr>
<td>0.000000000000000000000001355</td>
</tr>
<tr>
<td>0.00000000000000015</td>
</tr>
<tr>
<td>0.00000000001</td>
</tr>
<tr>
<td>0.0000000002</td>
</tr>
<tr>
<td>0.0000000132091</td>
</tr>
<tr>
<td>0.0000004</td>
</tr>
<tr>
<td>0.000000e0</td>
</tr>
<tr>
<td>0.0000001</td>
</tr>
</tbody>
</table>

**Figure 9.** The least frequently used infix-period tokens in the Usenet dataset.
In general, an ellipsis denotes a consecutive series of 3 periods. In the extended character ASCII table, typographically represented ellipses are also available, which is ASCII code decimal #135. Only one instance of such ASCII code decimal #135 in the Usenet dataset was found. For the Gutenberg dataset, no record was found since the Gutenberg dataset conformed to the basic ASCII character set. For this study, ellipses mentioned here and forward, we will refer to a consecutive series of 3 periods (ASCII code decimal #46).

In any case, a large number of ellipses was found in the Gutenberg dataset. The exact reasons are unclear, but one of the possibilities is the fact that the Gutenberg dataset contains publications that are rather old. There were technical issues associated with counting the ellipses. The reason is that each surrounding blank spaces seemed to be inconsistent and can become a problem with counting ellipses. As shown in Table 14, four distinctive variations of ellipses pattern can occur given that an ellipsis consists of exactly three periods. All of these variations are possible in a given dataset. Ideally, these stylistic differences should be standardized. In addition, the possibilities of users using ellipses other than 3 periods (e.g., 2 or 4 periods) must also be considered. These and other issues with inconsistent blank spaces with certain punctuation marks need further investigation.

Table 14

A Variation of Ellipses in Affix-Based Token Types

<table>
<thead>
<tr>
<th>Ellipse Variations</th>
<th>Affix-Based Token Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>□...□</td>
<td>infix-punctuation marks</td>
</tr>
<tr>
<td>□... □</td>
<td>suffix-punctuation marks</td>
</tr>
<tr>
<td>□ ...□</td>
<td>prefix-punctuation marks</td>
</tr>
<tr>
<td>□ ... □</td>
<td>punctuation-exclusive punctuation marks</td>
</tr>
</tbody>
</table>

Note: the symbol □ indicates any repeatable character such as a combinational punctuation pattern or alphanumeric characters.
Table 15 is an example of text segments that contain the ellipses variations. To provide an example, the lines that contain variations of affix-based ellipses tokens were matched and retrieved from the un-tokenized Gutenberg dataset. The variations of ellipses shown in the table appear to be valid forms of ellipse. Meanwhile, the table shows possible stylistic differences of ellipses due to surrounding blank spaces.

Table 15

*Text Lines Containing the Variation of Ellipses*

<table>
<thead>
<tr>
<th>Ellipse Variations</th>
<th>Text Lines Containing the Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>□…□</td>
<td>B.J...Bibliophile Jacob, i.e. Paul Lacroix. The laws of nations, and are subject to no other limitation...It hypothesis, however much beloved...as soon as facts are shown to be But he really must find that word. Curves curves...Those little valleys phrase, &quot;J'ai enonce les memes idees...que M. Darwin&quot; (volume ii.) is the ruins...and worse... Haven't I seen and heard things enough on</td>
</tr>
<tr>
<td>□...□</td>
<td>stiffened fingers... they were slender white fingers... had sought for &quot;Well... I haven't found out anything yet except that he is dead, and &quot;Did... did you know father very well?&quot; Jeff asked tremulously. East until I am over that, you know... Suppose I never get over it? couldn't do it I... But I can stay out there long enough to bring Glenn</td>
</tr>
<tr>
<td>□ ...□</td>
<td>...SCHOOL BOOKS... [Illustration: ...and the doughboy...] &quot;Noiselessly he stepped to her side and ...stood in silent prayer&quot;...232 32: ...as a further testimony to the majestic ... 8: ...The news of the Memorial Service you ... 12: ...He is eagerly awaiting to see the friends ...</td>
</tr>
<tr>
<td>□ ...□</td>
<td>&quot;Ah ... yes.&quot; Rockford was regarding him with disturbing amusement. &quot;You &quot;Ah ... good afternoon.&quot; 4. Be careful in the use of &quot;not ... and,&quot; &quot;any,&quot; &quot;but,&quot; &quot;only,&quot; &quot;not your head ... Only the opening paragraph at present, please!&quot; in the highest degree ... disfigured by an obscurity of diction which day. And the kids.... It couldn't have happened ... fate COULDN'T do</td>
</tr>
</tbody>
</table>

*Note:* the symbol □ indicates any repeatable character such as a combinational punctuation pattern or alphanumeric characters.
Prefix-Punctuation Marks

The frequency counts of the prefix-punctuation marks for the two datasets are shown in Table 16. Although some punctuation marks are placed in the prefix position, often the punctuation mark such as quotation marks and parenthesis are also used in pairs. To simplify the process, these types of punctuation marks were counted as prefix-punctuation marks and were not distinguished for now. Since some punctuation marks such as quotation marks are mostly used in pairs, this type of calculation is investigated later in this chapter.

Table 16

The Top 20 Prefix-Punctuation Marks

<table>
<thead>
<tr>
<th>Prefix-Punctuated-Marks</th>
<th>Usenet</th>
<th>Gutenberg</th>
<th>Relative Percentage Difference</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>188,990</td>
<td>326,505</td>
<td>42.12%</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>122,945</td>
<td>44,600</td>
<td>63.72%</td>
<td></td>
</tr>
<tr>
<td>'</td>
<td>26,975</td>
<td>40,310</td>
<td>33.08%</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>18,578</td>
<td>959</td>
<td>94.84%</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>17,485</td>
<td>1,808</td>
<td>89.66%</td>
<td></td>
</tr>
<tr>
<td>[</td>
<td>12,977</td>
<td>19,684</td>
<td>34.07%</td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>4,707</td>
<td>104,461</td>
<td>95.49%</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>3,575</td>
<td>556</td>
<td>84.45%</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>3,570</td>
<td>91</td>
<td>97.45%</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>3,245</td>
<td>524</td>
<td>83.85%</td>
<td></td>
</tr>
<tr>
<td>/</td>
<td>3,233</td>
<td>50</td>
<td>98.45%</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>2,194</td>
<td>9</td>
<td>99.59%</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>1,619</td>
<td>41</td>
<td>97.47%</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>1,498</td>
<td>426</td>
<td>71.56%</td>
<td></td>
</tr>
<tr>
<td>\</td>
<td>1,314</td>
<td>560</td>
<td>57.38% Graven accent (ASCII code decimal #96)</td>
<td></td>
</tr>
<tr>
<td>%</td>
<td>1,265</td>
<td>1</td>
<td>99.92%</td>
<td></td>
</tr>
<tr>
<td>&quot;</td>
<td>1,155</td>
<td>747</td>
<td>35.32%</td>
<td></td>
</tr>
<tr>
<td>--</td>
<td>1,101</td>
<td>2,260</td>
<td>51.28%</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>1,090</td>
<td>2,306</td>
<td>52.73%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>766</td>
<td>682</td>
<td>10.97%</td>
</tr>
</tbody>
</table>
The most frequently used punctuation marks for both datasets were the double quotation marks. Other more frequently used prefix-punctuation marks included a left parenthesis and a apostrophe (ASCII code decimal #39). More interestingly, in the Usenet dataset, a greater number of tokens surrounded by prefix-period was found.

Also, a large number of the prefix-asterisk * was identified in the Usenet dataset. On the other hand, in the Gutenberg dataset, a much smaller number were identified as prefix-asterisk punctuation. The relative percentage difference for the prefix-asterisk punctuation mark was 94.84%.

Some of the prefix-punctuation mark frequency occurrences were rather obvious. For example, conventionally some punctuation marks such as dollar signs are always placed in front of a number. For this reason, there were high frequency occurrences of prefix-dollar sign $. In the Gutenberg dataset, only a modest number of dollar signs was found. The difference further suggests the genre or domain difference between the two sets.

Much higher occurrences of prefix-colon punctuation patterns were found in the Usenet dataset than in the Gutenberg dataset. A total of 2,194 occurrences was found in the Usenet dataset, whereas only 9 occurrences of prefix-colon were found in the Gutenberg dataset. The relative percentage difference was 99.59%. This high percentage difference implies that the specific nature of colon usage might be associated with the Usenet dataset. Colons are commonly used as a suffix-punctuation mark in English.

**Suffix-Punctuation Marks**

The frequency counts for the suffix-punctuation marks for the two datasets are shown in Table 17. Here, we discuss the result of the frequency count and interesting aspects of suffix-
punctuation marks found in both datasets. The top frequently used suffix-punctuation mark was
the period in the Usenet dataset, while the top ranked frequently used suffix-punctuation mark
was the comma in the Gutenberg dataset. Between the two datasets, they had a relative
percentage difference of 15.91% for the period and 47.85% for the comma.

Table 17

*The Top 20 Suffix-punctuation Marks*

<table>
<thead>
<tr>
<th>Suffix Punctuated Token</th>
<th>Usenet</th>
<th>Gutenberg</th>
<th>Relative Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>1,203,987</td>
<td>1,431,821</td>
<td>15.91%</td>
</tr>
<tr>
<td>,</td>
<td>1,174,079</td>
<td>2,251,172</td>
<td>47.85%</td>
</tr>
<tr>
<td>?</td>
<td>97,917</td>
<td>42,886</td>
<td>56.20%</td>
</tr>
<tr>
<td>&quot;</td>
<td>80,223</td>
<td>30,974</td>
<td>61.39%</td>
</tr>
<tr>
<td>:</td>
<td>77,125</td>
<td>61,106</td>
<td>20.77%</td>
</tr>
<tr>
<td>)</td>
<td>76,450</td>
<td>21,963</td>
<td>71.27%</td>
</tr>
<tr>
<td>.&quot;</td>
<td>39,966</td>
<td>105,465</td>
<td>62.10%</td>
</tr>
<tr>
<td>!</td>
<td>38,206</td>
<td>52,848</td>
<td>27.71%</td>
</tr>
<tr>
<td>;</td>
<td>32,067</td>
<td>207,088</td>
<td>84.52%</td>
</tr>
<tr>
<td>'</td>
<td>26,531</td>
<td>41,892</td>
<td>36.67%</td>
</tr>
<tr>
<td>...</td>
<td>25,805</td>
<td>1,062</td>
<td>95.88%</td>
</tr>
<tr>
<td>),</td>
<td>24,555</td>
<td>5,373</td>
<td>78.12%</td>
</tr>
<tr>
<td>&quot;,</td>
<td>22,714</td>
<td>89,137</td>
<td>74.52%</td>
</tr>
<tr>
<td>*</td>
<td>15,993</td>
<td>328</td>
<td>97.95%</td>
</tr>
<tr>
<td>),</td>
<td>14,922</td>
<td>8,182</td>
<td>45.17%</td>
</tr>
<tr>
<td>&quot;.</td>
<td>12,450</td>
<td>567</td>
<td>95.45%</td>
</tr>
<tr>
<td>-</td>
<td>11,990</td>
<td>933</td>
<td>92.22%</td>
</tr>
<tr>
<td>]</td>
<td>11,904</td>
<td>16,115</td>
<td>26.13%</td>
</tr>
<tr>
<td>?&quot;</td>
<td>9,296</td>
<td>43,716</td>
<td>78.74%</td>
</tr>
<tr>
<td>%</td>
<td>8,785</td>
<td>248</td>
<td>97.18%</td>
</tr>
</tbody>
</table>

Although conventional grammatical usage of punctuation marks indicate where the
punctuation marks should be physically placed when in use, the behavior pattern in terms of
physical locality of punctuation marks was less predictable sometimes. In the case of comma,
the conventional locality of punctuation mark is clearly predictable since commas are typically
placed at the end of a character string. In contrast, in the case of a period, the pattern might be less predictable as the period can be used numerous ways other than an end of sentence.

Because of multiple roles that one particular punctuation mark can play in a language, occasionally examining additional surrounding character strings or a sentence to determine its linguistic roles are required. The disambiguation of the syntactic role is important. It would clearly indicate whether the period is used as a source-independent or sub-lexical punctuation mark. As mentioned in Chapter 2, various disambiguations of textual features can be considered including the length of a character string, known list of abbreviated character strings, position of the capitalization of a character string, etc.

There were other punctuation marks that had a high number of occurrences and high relative percentage difference. More ellipses, which consisted of three periods ..., were also detected as suffix-punctuation marks in the Usenet dataset. The implication of this frequency count is that in counting punctuation marks, often inconsistent variants are likely to exist in the corpus. The uses of ellipses clearly indicate stylistic differences as discussed earlier. That is, we have to consider how blank spaces are used with some of these punctuation marks.

Unlike the Gutenberg dataset, a greater number of hyphens was found as a suffix-punctuated token in the Usenet dataset. Since the role of a hyphen can differ, it can become rather a complex problem to correctly count based on the individual role of a hyphen. As discussed before, there are similar typographic fonts such as en dashes and em dashes. However, the main issue is that hyphens and dashes have a variety of functions as described by Straus and Fogarty (2007). Considering the frequency counts and variety of roles that a hyphen has in the English language, normalization of hyphens obviously has to be carefully examined. Moreover, since hyphens and dashes may look alike typographically, we also suspected hyphens might be
used interchangeably with dashes. These issues will be discussed in more detail later in this chapter.

Comparing suffix-punctuation marks to prefix punctuation marks, the overall frequency counts of suffix-punctuation marks were much greater than prefix-punctuation marks. We can also revert back to Table 6 and clearly notice the differences between the two. One of the possible reasons for having a greater frequency count in the suffix position is that most punctuation marks that have separating roles are placed in the suffix position of character strings. For example, punctuation marks such as a comma are usually placed at the end of a character string. Also, other punctuation marks that are associated with stress markers such as the exclamation mark is typically used at the end of the character string. There are rare exceptions where an exclamation mark is used in a prefix position (e.g., in a computer language). As shown in Table 17, a moderate number of exclamation marks was found in both datasets. Nevertheless, the positions of punctuation marks are a reasonable predictor of the particular role of punctuation marks.

**Punctuation-Exclusive Tokens**

As discussed previously, a punctuation-exclusive token is a non-affix-based token that is bound by blank spaces. For example, punctuation marks such as ++ is a punctuation-exclusive token if no alphanumeric characters are attached to the punctuation marks. Calculating punctuation-exclusive token frequencies was another useful means to discover issues related to punctuation normalization. The frequency of punctuation-exclusive tokens was particularly useful for two reasons:

- It provided means to gain insights into the problems that are associated with normalizing various types of punctuation marks; and
• It provided additional means to characterize the particular dataset, especially in terms of particular uses of punctuation marks.

Most of the punctuation-exclusive tokens represent punctuation marks that were not source-independent; they were source-specific instead. From the examination, it was clear that punctuation-exclusive tokens were most associated with the following types:

• Source-independent punctuation marks that have a delimiting punctuation mark role (e.g., bracket)

• Punctuation marks that have a structural formatting role (e.g., consecutive periods in the table of contents)

• Source-specific punctuation marks such as slash, asterisk, etc. that often require examination of neighboring character strings to determine the punctuation marks function (role)

Table 18 shows the top 20 most frequently occurring punctuation-exclusive tokens found in the Usenet dataset compared to the Gutenberg dataset. Note that reversing the order of datasets can reveal some of the additional noteworthy punctuation-exclusive tokens. However, the table with reversed-order is omitted since Table 18 is sufficient for discussion purposes.

As shown in this table, the most frequently used punctuation-exclusive token in the Usenet dataset was hyphen. Hyphen-exclusive token means a single token that consists of hyphen only. To be categorized as a hyphen-exclusive token, it needs to be bounded by blank spaces. Thus, a hyphen-exclusive token implies that no other punctuation mark or alphanumeric characters are attached to the hyphen. From this point and on, this type of naming convention (e.g., hyphen-exclusive) will be used for all tokens that are punctuation-exclusive.

All of the following lines contain hyphen-exclusive punctuation tokens:

• December 23, 2009 - Congressional Apportionment...

• Driving conditions were treacherous - lots of ice around and heavy sleet,...

• Embattled Garden" (Lilith) and "Part Real - Part Dream."...
- Every time - every time - we publish an article about Mono...

The ellipses in the above list are used to note the continuation of the text and not the actual punctuation mark patterns found in the dataset.

Table 18

*The Top 20 Punctuation-Exclusive Tokens*

<table>
<thead>
<tr>
<th>Punctuation-Exclusive Token</th>
<th>Usenet</th>
<th>Gutenberg</th>
<th>Relative % Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>59,251</td>
<td>2,764</td>
<td>95.34%</td>
</tr>
<tr>
<td>--</td>
<td>17,682</td>
<td>6,440</td>
<td>63.58%</td>
</tr>
<tr>
<td>.</td>
<td>14,967</td>
<td>60,108</td>
<td>75.10%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12,896</td>
<td>9,241</td>
</tr>
<tr>
<td>=</td>
<td>10,253</td>
<td>573</td>
<td>94.41%</td>
</tr>
<tr>
<td>&amp;</td>
<td>9,420</td>
<td>3,267</td>
<td>65.32%</td>
</tr>
<tr>
<td>...</td>
<td>9,014</td>
<td>3,196</td>
<td>64.54%</td>
</tr>
<tr>
<td>:</td>
<td>8,738</td>
<td>45</td>
<td>99.49%</td>
</tr>
<tr>
<td>*</td>
<td>6,783</td>
<td>43,269</td>
<td>84.32%</td>
</tr>
<tr>
<td>,</td>
<td>6,134</td>
<td>22</td>
<td>99.64%</td>
</tr>
<tr>
<td>?</td>
<td>3,938</td>
<td>32</td>
<td>99.19%</td>
</tr>
<tr>
<td>+</td>
<td>3,902</td>
<td>288</td>
<td>92.62%</td>
</tr>
<tr>
<td>&quot;</td>
<td>3,885</td>
<td>5,955</td>
<td>34.76%</td>
</tr>
<tr>
<td>/</td>
<td>3,581</td>
<td>494</td>
<td>86.20%</td>
</tr>
<tr>
<td>:)</td>
<td>2,997</td>
<td>1</td>
<td>99.97%</td>
</tr>
<tr>
<td>(</td>
<td>2,595</td>
<td>29</td>
<td>98.88%</td>
</tr>
<tr>
<td>)</td>
<td>2,204</td>
<td>76</td>
<td>96.55%</td>
</tr>
<tr>
<td>}</td>
<td>441</td>
<td>679</td>
<td>35.05%</td>
</tr>
<tr>
<td>***</td>
<td>285</td>
<td>1,776</td>
<td>83.95%</td>
</tr>
<tr>
<td>----</td>
<td>108</td>
<td>586</td>
<td>81.57%</td>
</tr>
</tbody>
</table>

The above examples were drawn from the Gutenberg dataset. Sometimes hyphens might be used in place of dashes as they look similar. One of the reasons might be due to the fact that users might have difficulties in placing true typographical dashes (e.g., en dash and em dash) because they take more effort to search and use them based on a conventional keyboard.
In relative terms, a considerably smaller number of hyphen-exclusive tokens was found in the Gutenberg dataset. The frequency occurrence of hyphen-exclusive tokens was 59,251 for the Usenet dataset, while the frequency occurrence of hyphen-exclusive tokens was 2,764 for the Gutenberg dataset. The relative percentage difference between the two dataset was 95.34%.

In contrast to the Usenet dataset, the most frequently used punctuation-exclusive token in the Gutenberg dataset was the period. The total occurrences of period-exclusive tokens were 60,108 for the Gutenberg dataset, whereas the total occurrences of period-exclusive tokens were 14,967 for the Usenet dataset. The relative percentage difference was 75.10%. Once again, the result indicated that the period is a dominant punctuation mark that can be used in a variety of affixes. Overall, there were a number of punctuation-exclusive tokens for which the percentage of difference was more than 90%. These were the hyphen -, equal sign =, colon :, comma ,, question mark ?, plus sign +, a colon followed by a right parenthesis :) , right parenthesis ), and left parenthesis (.

Notice that an individual punctuation mark in a series is different from the role of punctuation pattern as a whole. Combinational uses of punctuation marks can be referred to as combinational punctuation marks. For example, a combinational punctuation mark :) as shown in the table is semantically different from the colon : and the right parenthesis ). Yet, semantically, the roles of combinational punctuation marks can be closely related to its member punctuation mark, especially when the every member punctuation mark is identical to each other.

For example, the role of double hyphen -- is similar to the role of single hyphen - as shown in Figure 10. In the case of line 6 and line 9, it was used as a single hyphen -. In other cases, the double hyphen -- were used as a dash. Nevertheless, inconsistent uses of hyphens, often as a double hyphen -- were apparent in the Usenet. There was a possibility that dashes
might have been converted to double hyphens at some point prior to downloading the Usenet archive from the Internet.

Occasionally, combinational punctuation marks produced entirely different meanings when combined. They were usually used in a relatively non-standard way. For example, in relative terms, a considerable number of colons followed by a right parenthesis : was found in the Usenet dataset. Often a colon followed by a right parenthesis :) is used to represent a “smiling face”. As a result, this punctuation-exclusive token had the highest relative percentage of differences. In the Gutenberg, on the other hand, only one such punctuation pattern was found. This caused a higher relative percentage differences. Intuitively such punctuation marks are not considered as a standard expression in a typical book. Obviously, it was also evidence that showed a sense of “casualness” on the Usenet dataset.

<table>
<thead>
<tr>
<th>LINE#</th>
<th>Lines Containing Em dash</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Haines City, Florida -- The family of Robert Donaldson has already</td>
</tr>
<tr>
<td>2</td>
<td>Hampton was a dangerous radical -- just before his death he had told the</td>
</tr>
<tr>
<td>3</td>
<td>Hamrlik -- Gorges</td>
</tr>
<tr>
<td>4</td>
<td>Hardly -- that doesn't describe &quot;every&quot; [Muslim] &quot;in a robe&quot;.</td>
</tr>
<tr>
<td>5</td>
<td>Hash/pound sign -- slanted uprights: // and horizontal cross lines: ==.</td>
</tr>
<tr>
<td>6</td>
<td>He also noted -- in his journal -- that he became a close friend of</td>
</tr>
<tr>
<td>7</td>
<td>He and VP were both in the Senate for years -- so he definitely was at least</td>
</tr>
<tr>
<td>8</td>
<td>He can't -- his club is integrated...</td>
</tr>
<tr>
<td>9</td>
<td>He could tell how much liquor was left in a bottle -- and thus, when it was</td>
</tr>
<tr>
<td>10</td>
<td>He had eight children -- five sons and three daughters -- and lived</td>
</tr>
<tr>
<td>11</td>
<td>He has retired twice now -- two years ago in a full-blown quarrel with the</td>
</tr>
<tr>
<td>12</td>
<td>He has the same solidness as a lot of the late GI molds -- the</td>
</tr>
<tr>
<td>13</td>
<td>He once said his philosophy regarding business -- one that valued</td>
</tr>
<tr>
<td>14</td>
<td>He pushed in as deeply as he could -- and the balloon burst. He felt</td>
</tr>
<tr>
<td>15</td>
<td>He remembers trying the Chamfered Luby -- and failing miserably.</td>
</tr>
<tr>
<td>16</td>
<td>He stated that is results were significant -- but did not</td>
</tr>
<tr>
<td>17</td>
<td>He was seen about a week and a half ago outside -- a rarity, BTW -- and he</td>
</tr>
</tbody>
</table>

**Figure 10.** Exemplary text lines containing double dashes (Usenet).

To obtain a visual-based perspective of punctuation-exclusive tokens, Table 18 was transformed into a graph in Figure 11. The X-axis indicates the punctuation-exclusive mark
token, while the y-axis shows the frequency count. As shown in this figure, the Usenet dataset, in general, had a much higher occurrence of punctuation-exclusive tokens. In the Gutenberg dataset, a smaller number of punctuation-exclusive tokens was found. As shown in Table 18, 10 out of 20 punctuation-exclusive tokens have a 90% or higher relative percentage difference.

By examining punctuation-exclusive punctuation marks a number of causes maybe identified. A substantial number of asterisk-exclusive tokens was discovered in the Gutenberg dataset as formatting punctuation marks. As a result, this contributed to a high asterisk-exclusive frequency count as shown in Figure 11. In a non-conventional sense, any punctuation mark could be used as a formatting indicator. To this end, the implication for a text retrieval system was such that punctuation mark usages need to be detected and the roles need to be disambiguated.

We propose that the key to identifying punctuation mark roles is to find the root causes of extreme discrepancies. The frequency result comparison suggested that there was a greater
chance that the punctuation mark in one of the dataset was source-specific if the relative percentage difference was substantially higher. Also, based on identifiable punctuation mark roles, often the causes of the discrepancy can be intuitively explained.

Figure 12 is a snapshot of less frequently used punctuation-exclusive tokens in the Gutenberg dataset. While Table 18 only shows the most frequently punctuation-exclusive token, the less frequently occurring punctuation-exclusive token counts should not be ignored from the punctuation mark analysis. In this figure, a frequency count is shown on the left side of the figure while the actual punctuation-exclusive tokens are shown on the right of the figure. A more general observation can be made regarding the punctuation-exclusive token. In comparison to the most frequently used exclusive-tokens, the lower ranked punctuation-exclusive tokens appeared to be longer in number of characters than top frequently used punctuation-exclusive tokens.

In general, calculating punctuation-exclusive tokens was another useful method of detecting abnormal punctuation patterns. Once some type of sorted list becomes available, finding a punctuation pattern visually can be useful as well. In sum, for each punctuation pattern that cannot be characterized easily, further investigation was necessary. Two notable punctuation marks were selected and discussed further in the subsequent section.
Figure 12. Low ranked punctuation-exclusive token in the Gutenberg dataset.

Period-Exclusive Token

The period was one of the punctuation-exclusive tokens that occurred substantially in both datasets. Figure 13 is a random sample of text lines that contain the period as a punctuation-exclusive token. A period-exclusive token could be formed by one of the following:

- Ellipses
- Lines to fill the space in a chapter of the book
In a typical sentence, a period is usually placed at a suffix position or at an infix position. In these cases, a period needs to be placed immediately before or immediately after an alphanumeric character. Accordingly, in a typical sentence, a period should not be placed alone.

The high frequency occurrence of the period was an indication that the punctuation mark usage was neither sub-lexical nor source-independent. This type of period usage was clearly considered as formatting punctuation marks.

The Gutenberg dataset contained a greater number of period-exclusive tokens than the Usenet dataset. In the Gutenberg dataset, the purpose was to fill in empty blank spaces. To do this, the periods were used as formatting punctuation marks. Since the Gutenberg dataset is mainly comprised of books and periodicals, there was a fair number of punctuation marks occurring in the “table of contents”.

Figure 13. Example of punctuation-exclusive token in the Gutenberg dataset.
By examining exclusive token usage, we were able to recognize varying degrees of punctuation mark roles in an underlying text. Also, in this case, it demonstrates how the spacing needs to be considered in determining the punctuation mark role. In a typical “table of contents”, blank spaces between chapter heading and page numbers often contained a series of periods. Although ellipsis normally consists of three periods without using blank spaces, because of blank spaces between each surrounding period, this caused a high frequency count of period-exclusive tokens for the Gutenberg dataset.

*Hyphen-Exclusive Token*

As shown in Table 18, the punctuation-exclusive token with the highest frequency count in the Usenet dataset was the hyphen. The result was interesting since the relative percentage difference between the two datasets was high in many instances. As Table 18 indicates, 59,251 occurrences of hyphen-exclusive token were found in the Usenet while only 2,764 occurrences of hyphen-exclusive tokens were found in the Gutenberg dataset. The relative percentage difference was 95.34%.

Let us consider the overall frequency count of hyphen-infix-punctuation marks again. The difference was notable since the Usenet dataset only contained 112,948 occurrences of hyphen-infix $-$ while the Gutenberg dataset contained 145,903 occurrences of hyphen-infix $-$ . The relative percentage difference was 95.34% for the hyphen-exclusive token whereas 22.59% was for the hyphen-infix-punctuation mark. The discrepancy between these two relative percentage differences, one as a hyphen-infix-punctuation mark and the other as a hyphen-exclusive token, was considerably higher.
In general, it was necessary to recognize specific roles of hyphens in the underlying datasets. In doing so, most of hyphens fell into one of the following categories:

- Compound words (e.g., health-care)
- Prefixed words (e.g., re-read)
- Dashes (e.g., en dash and em dash)
- Marking the end of line (e.g., soft hyphen)
- Spellings of individual letters (e.g., l-o-u-d-e-r!!),
- Range of values (e.g, 90-100)
- Dates (e.g., 1-1-2011)
- Computer-command line options (e.g. -v)
- Mathematical formula (e.g., y=a-b)

Most of the above hyphen categories were based on filtering tokens with infix-hyphens and by manually inspecting the text segments that contain various types of hyphens. A standard grammar book on punctuation marks could be used as a guide in developing these type of categories. These include Quirk et al. (1985); and Straus and Fogarty (2007).

The reason for the greater number of infix-exclusive tokens was the fact that the Usenet dataset contained varied forms of hyphens as mentioned earlier. Especially, in the Usenet dataset, hyphen categories (c), (d), (f), (g), (h), and (i) were clearly more predominant. These were evidently linked to the causes of the high number of hyphen-punctuation-exclusive tokens.

The fact that spacing issues could cause problems was apparently noticeable. For the above (h) type, -v is a Unix command line option. Here, -v needs to be used without any blank space before the letter v. Having a blank space between the hyphen and the letter would be - v. This type of command is not allowed since computer commands are highly sensitive to blank
spaces and punctuation marks. In general, adding an internal blank space in this might result one or more following problems:

- Stylistically unconventional
- Syntactically unacceptable
- Semantically ambiguous

In other cases, surrounding blank spaces could be used either way. Both leaving and not leaving blank spaces were acceptable since the practice of leaving extra blank space was just related to a stylistic difference. For example, the mathematical formula \( c = a - b \) could be also written as \( c = a \cdot b \). For another example, the value range 90 - 100 could be also written as 90-100.

Semantically, there is no major difference between these two expressions. This different spacing convention, however, can provide clues to interpreting the statistical results. Such a case also implied that multiple tokenization options were possible. Segmenting based on blank spaces would have caused the formation of tokens \( c, =, \cdot, a, \cdot \), and \( b \).

To a certain extent, punctuation-exclusive tokens showed various effects of tokenization. The reason is that determining a correct segmenting marker is not a trivial task since it requires some type of generalization. Generalizing from specific cases to a more general case in dealing with tokenization is a challenging task of its own. Such tokenization problems indicated that the punctuation mark functions might be highly domain-specific or genre-specific in some cases. Thus neighboring tokens elements may need to be examined in conjunction with the particular token to determine its context and type of punctuated item. For this reason, tokenization is a required procedure in figuring out the punctuation mark role. Moreover, in terms of text retrieval, blank spaces could cause variability of character strings, both in the content side and in the search side. As a consequence, string matching could be performed in multiple ways.
The results suggested that further in-depth analysis needs to be conducted based on the hyphen phenomenon. At minimum, the most common issue such as multi-part character strings needs to be carefully examined from a text retrieval perspective.

Variations of Affix-Based Token Type

Some tokens could be constructed based on various punctuation mark affixes. For example, first consider only hyphenated tokens as shown in Table 19. Based on the punctuation mark affix, four possible formations are possible: infix-hyphen, prefix-hyphen, suffix-hyphen, and exclusive-hyphen. When the Usenet dataset was compared to the Gutenberg dataset, prefix, suffix-punctuation marks, and punctuation-exclusive token were much higher. These differences were noteworthy since more than likely they were indications of domain-specific hyphens.

Table 19

<table>
<thead>
<tr>
<th></th>
<th>Infix</th>
<th>Prefix</th>
<th>Suffix</th>
<th>Punctuation-Exclusive Token</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(z-z)</td>
<td>(-z)</td>
<td>(z-)</td>
<td>(-)</td>
</tr>
<tr>
<td>Usenet</td>
<td>112,948</td>
<td>3570</td>
<td>11,990</td>
<td>59,251</td>
</tr>
<tr>
<td>Gutenberg</td>
<td>145,903</td>
<td>91</td>
<td>933</td>
<td>2,764</td>
</tr>
<tr>
<td>Relative % Difference</td>
<td>22.59%</td>
<td>97.45%</td>
<td>92.22%</td>
<td>95.34%</td>
</tr>
</tbody>
</table>

Paired-Punctuation Marks

Often punctuation marks were placed in pairs. For example, punctuation marks such as parentheses have the opening and closing marks. Thus, in terms of affixes, a number of suffix-prefix-punctuated tokens were selected and investigated to understand the punctuation pattern in
detail. The result is shown in Table 20. The grep commands that were used for generating each frequency count is also shown in this table.

In Table 20, the letter \( \mathbf{x} \) was used to replace a single alphanumeric in a token, while the symbol \( \square \) can replace any type of characters between the pair of \( \mathbf{x} \). Because each paired punctuation marks reach beyond a word boundary, the frequency count was performed against the un-tokenized datasets, \textit{news.temp} and \textit{g.temp}. There is a blank space after the initial opening quotation mark and before the closing quotation mark since we used the un-tokenized datasets to calculate the paired-punctuation marks.

The frequency count shown in Table 20 does not accurately reflect all paired-punctuation marks in the dataset. Along with the paired-punctuation marks, often another same type of punctuation mark was present in the same line. For example, calculating a paired period-punctuation marks using the pattern \( \mathbf{x} \square \mathbf{x} \) did not produce accurate results since the two period were not functionally used as a pair in some instances. Also, some punctuation marks such as quotation marks might expand beyond a single line. For example, an opening quotation mark might be present in one line, and the closing quotation mark might be present in another line.

However, we detected a number interesting patterns through the frequency count of paired-punctuation marks. For example, in the Usenet dataset, occurrences of paired asterisk \( \star \mathbf{x} \square \star \) were much higher compared to the Gutenberg dataset. Also, the occurrences of paired underscore \( _{\mathbf{x}} \) were much higher in the Gutenberg dataset compared to the Usenet dataset. The paired underscore \( _{\mathbf{x}} \) turned out be an specific pattern that was anomalously present in the Gutenberg dataset.

Notice that the above calculation is based on “without” post and pre blank space. The post blank space is a blank space immediate after the opening punctuation mark (e.g., blank
space after left parenthesis). The pre blank space is a blank space immediately prior to the closing punctuation mark (e.g., blank space before right parenthesis). Thus, Table 20 is not based on post and pre blank spaces. The reported was based on “without” post and pre (post-pre) blank spaces. To produce the representation of “with” post-pre blank space would require one blank space added after and before each symbol □. For example, calculating a frequency count of parenthesis “with” post-pre blank space can be represented as ( □x ), while “without” post-pre blank space can be represented as (x□x). We also have to consider the fact that sometimes a paired-punctuation mark might not be evenly spaced out due to a writer’s own inconsistent behavior. For example, we could have (□x), which shows that pre blank space on the right side but no post blank space on the left side. Another possibility is that we could have (x□x), which shows that post blank space on the left side but no pre blank space on the right side.

Table 20

**Selected Paired-Punctuation Marks Without Post-Pre Blank Spaces**

<table>
<thead>
<tr>
<th>Paired-Punctuation Marks</th>
<th>Usenet</th>
<th>Gutenberg</th>
<th>Relative % Difference</th>
<th>Grep Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;x□x&quot;</td>
<td>62,358</td>
<td>18,411</td>
<td>70.48%</td>
<td>grep &quot;<em>[a-zA-Z0-9].</em>[a-zA-Z0-9]&quot; /&quot; filename</td>
</tr>
<tr>
<td><em>x□x</em></td>
<td>14,481</td>
<td>73</td>
<td>99.50%</td>
<td>grep &quot;!<em>[a-zA-Z0-9].</em>[a-zA-Z0-9]!* &quot; filename</td>
</tr>
<tr>
<td>(x□x)</td>
<td>14,331</td>
<td>9,521</td>
<td>33.56%</td>
<td>grep &quot;([^a-zA-Z0-9].*[a-zA-Z0-9]) &quot; filename</td>
</tr>
<tr>
<td>\x□x</td>
<td>12,399</td>
<td>4,467</td>
<td>63.97%</td>
<td>grep &quot;[^a-zA-Z0-9].[a-zA-Z0-9]&quot; &quot; filename</td>
</tr>
<tr>
<td>[x□x]</td>
<td>3,817</td>
<td>1,496</td>
<td>60.81%</td>
<td>grep &quot;[[a-zA-Z0-9].*[a-zA-Z0-9][ ]] &quot; filename</td>
</tr>
<tr>
<td><em>x□x</em></td>
<td>3,384</td>
<td>30,923</td>
<td>89.06%</td>
<td>grep &quot;[a-zA-Z0-9]*[a-zA-Z0-9]_ &quot; filename</td>
</tr>
<tr>
<td>/x□x/</td>
<td>1,033</td>
<td>10</td>
<td>99.03%</td>
<td>grep &quot;/[a-zA-Z0-9]*[a-zA-Z0-9]/] &quot; filename</td>
</tr>
<tr>
<td>.x□x.</td>
<td>401</td>
<td>35</td>
<td>91.27%</td>
<td>grep &quot;.[a-zA-Z0-9].*[a-zA-Z0-9]. &quot; filename</td>
</tr>
<tr>
<td>-x□x-</td>
<td>349</td>
<td>0</td>
<td>100.00%</td>
<td>grep &quot;-[a-zA-Z0-9].*[a-zA-Z0-9]- &quot; filename</td>
</tr>
<tr>
<td>...x□x...</td>
<td>29</td>
<td>0</td>
<td>100.00%</td>
<td>grep &quot;\[a-zA-Z0-9].*[a-zA-Z0-9]\ &quot; filename</td>
</tr>
<tr>
<td>%x□x%</td>
<td>21</td>
<td>0</td>
<td>100.00%</td>
<td>grep &quot;%[a-zA-Z0-9].*[a-zA-Z0-9]% &quot; filename</td>
</tr>
<tr>
<td>#x□x#</td>
<td>7</td>
<td>20</td>
<td>65.00%</td>
<td>grep &quot;#[a-zA-Z0-9].*[a-zA-Z0-9]# &quot; filename</td>
</tr>
</tbody>
</table>

*Note:* the symbol □ indicates any repeatable character such as a combinational punctuation pattern or alphanumeric characters.
There was a clear limitation to the above method of calculating the paired-punctuation marks. If we are considering a character string *Evan, R., 2013* that is surrounded by parentheses, the frequency count should examine all of the combinations including some internal punctuation marks between blank space based tokens as shown in Figure 14. Having a pre or post blank space only on one side is inconsistent and stylistically unconventional, but an inconsistency exists frequently when the Usenet dataset was inspected. The inconsistent use of blank spaces was probably created due to the writer’s casualness in the Usenet. Furthermore, this example depicts how the spacing can become a more complicated issue in conducting a frequency count. Some type of normalization of blank space is clearly required for conducting frequency counts of paired-punctuation marks, although most of these issues could be resolved once we identify meaningful tokens.

![Diagram](image)

*Figure 14. Variation of paired-punctuation marks based on post-pre blank spaces.*

In addition, the length of character strings or sentences varied. Accurately performing a frequency count “with” pre-post blank space was not a trivial task for a punctuation mark such as double quotes due to difficulties in matching the pairs. For example, in some cases, if there are three double quotes internally with blank spaces, then the calculation will be off since we do not
easily know where the correct pairing of double quotes. It could be a simple mistake in the text or there could be an orphan quote which could stand for an inch (e.g., 5” instead of 5 inch).

Overall, a paired-punctuation mark frequency count “with” post-pre blank space was no less than a paired-punctuation mark count using “without” pro-pre blank space for certain punctuation marks as quotation mark.

Table 20 shows that the double quotation mark “” (ASCII code decimal #34) was the most frequently used paired-punctuation mark. The parentheses () was the second highest paired-punctuation mark in the Usenet dataset. The asterisk * was the third highest paired-punctuation mark pair. The brackets [] were commonly used in both datasets as a paired-punctuation mark. In contrast, a relatively low number of asterisks * was found as a paired-punctuation pattern in the Gutenberg dataset. This clearly indicates a paired-asterisk * is used much more as a modern, casual expression. Also, some paired- asterisks are likely to be associated with mathematics since the asterisk denotes the notion of “multiplication”. Both datasets showed less common instances of the number sign # and percentage sign % as paired-punctuation marks.

Based on the relative percentage differences, some of the paired-punctuation mark usages were clearly different from one another. For example, 1,033 occurrences of paired-slashes were found in the Usenet, whereas only 10 occurrences of paired-slashes were found in the Gutenberg dataset. The paired-slashes were represented as /x□x/. The paired-period without internal blank space was represented as ∙x□x. There were 401 instances of the paired-period ∙x□x in the Usenet dataset, but only 35 instances of the paired-period ∙x□x were found in the Gutenberg dataset.
Overall, when the frequency count of lexical-level punctuation pattern was extremely low, drawing a definitive conclusion regarding the punctuation pattern usage was difficult. The reason was that the text lines that contained the punctuation pattern that was necessary to examine the contextual usage could not be retrieved. In general, however, variants of punctuation marks usage such as a period were much higher in the Usenet dataset than the Gutenberg dataset. Overall, these types of punctuation pattern usage reflected a non-standard, domain-specific related language found in the Usenet dataset.

In a conventional sense, leaving a blank space between some punctuation marks such as a dash or a parenthesis would be stylistically acceptable. An exception to this generalization is a specific stylistic guideline. A specific stylistic guideline can always dictate, override, or constrain the writing aspects especially concerning the particular use of punctuation marks.

According to Jones (1997), punctuation marks that have delimiting functions are known to be used in pairs. Furthermore, they belong to the category of source-independent punctuation marks. Yet, we noticed that paired-punctuation marks can occur with a variety type of punctuation marks other than ones Jones mentioned. For example, a paired-punctuation mark that had a high frequency count was the underscore _. Compared to the Usenet dataset, in the Gutenberg dataset a greater number of paired-punctuation marks was found. In the Gutenberg dataset, the underscore was primarily used as a formatting punctuation mark as they were used to fill up blank spaces.

A pertinent question that can be asked was whether the punctuation mark such as a paired-underscore can have any type of functional role during the preparation of a text retrieval system. The frequency strategy that was used showed that it was not difficult to detect these abnormalities to a certain extent. If the goal is to utilize these punctuation marks to a certain
extent, removing these punctuation marks requires a more judicious consideration of their
punctuation mark roles. In addition to blank spaces, perhaps a paired-punctuation mark such as
the underscore could be used as a tokenization point. Overall, the frequency count of a paired-
punctuation mark demonstrated another effective means to examine the characteristics of
punctuation marks in the dataset.

Additional Types of Punctuation

*Additional types of punctuation* is the third and last punctuation frequency analysis
procedure of this study. So far, the analysis focused on affix-based types. Often there were
some notable punctuation pattern differences between the two sets. It was possible to classify
additional punctuation mark types and punctuation pattern types based on the frequency counts.
At this point, it became clear that additional categories had to be considered in constructing
punctuation normalization rules. These results and issues are discussed in the following section.

*Basic Punctuation Patterns*

One of the reasons for applying punctuation frequency analysis procedures was to
identify and examine punctuation patterns. In our discussion so far, the notion of punctuation
pattern was used loosely in many instances. Most often the term punctuation pattern refers to a
*structural punctuation pattern*. This means a recurring use of punctuation marks based on a
particular choice of segmentation unit (e.g. punctuated character string, punctuation marks in a
sentence, etc.). Additional terms can be used to categorize the punctuation pattern for analysis
purposes. We categorized a punctuation pattern into two types:

- Lexical-level punctuation pattern
• Sentence-level punctuation pattern

A *lexical-level punctuation pattern* is a type of punctuation pattern that is recognizable as a result of a punctuated character string used in a text. Once again, a punctuated character string is a character string that contains at least one punctuation mark. For example, the use of an apostrophe ′, which often indicates a possessive or contraction, is a lexical-level punctuation pattern. A punctuation pattern that appears as an apostrophe-s ′s is a lexical-level punctuation pattern since the apostrophe is associated with a word. For another example, abbreviated character strings are often associated with lexical-level punctuation patterns since periods often appear after a single letter (e.g., x. x. x.). The letter x can be used to replace a single alphanumeric character.

A *sentence-level punctuation pattern* is a type of punctuation pattern that is recognizable as a result of using punctuation marks in a text segment. Suppose we want to analyze a pattern based on a sentence level. The letter e can be substituted for any alphabetical characters that appear in a segmented text. In this case, if we have a sentence such as *Mr. Smith, this is my dog.*, then the structural view of the punctuation pattern could be represented as *e.e.e.* In this way, one may calculate how many text segments have this particular pattern in the text.

From a perspective of normalizing punctuation marks, different interpretations of punctuation patterns are possible. Since various punctuation mark characteristics were discovered, the results were used in the latter part of analytical procedure that deals with qualitatively observing the punctuation normalization issues.

*Domain-Specific Punctuation Pattern*

By using a frequency count strategy, some punctuation patterns could be identified with
specific domain. A domain is a sphere of knowledge, influence (Domain, 2013). Just as the character strings used can differ depending on the context, some punctuation patterns depend upon the domain and have particular characteristics. For example, `cat -v` is a UNIX command with `-v` option. In this command, the `-v` option contains a hyphen before the letter `v`. A punctuation pattern, having a hyphen-prefix, can be commonly found in computer programming books. Unless noted otherwise, the hyphen is referred to as ASCII code decimal #45 in this study.

Identifying domain-specific punctuation patterns would be useful in predicting specific punctuation mark roles. If the content contains a hyphen-prefix then we need to determine whether the hyphen is perhaps a “dash” or a computer command option such as `-v`. If it is a dash, the dash could perhaps be normalized by simply removing it. However, if it is a computer command prefix, then we need to keep the hyphen intact with the letter `v`. Since domain differences were reflected in the form of punctuation patterns, categorizing punctuation patterns according specific domain types seemed useful for punctuation normalization. In the type of domain we just discussed, it is more appropriate to develop a punctuation normalization rule that is more sensitive to treating the hyphen followed by a single character.

For another example from the datasets, we need a more accurate method to disambiguate character strings such as `b/c` and `i/o`. If these character strings are used in the mathematics domain, then the slash most likely represents a division. Thus, a useful strategy is to understand the particular domain characteristics in relation to the underlying punctuation patterns. Essentially, a feasible approach may be the following. If a corpus contains the domain-specific type of a term or punctuation pattern, then a disambiguating process would be required to determine the context of the character string. Furthermore, based on an identification of domain-
specific punctuation patterns, contextual usage can be determined by examining neighboring character strings.

This type of disambiguation routine might have to rely on a specially created lexicon that includes exceptional punctuation patterns and their variations. In turn, a matching function would be processed according to the disambiguated punctuation pattern. Overall, determining the domain-specific punctuation requires examining punctuation patterns, overall frequency occurrences, and relative percentage differences.

In sum, it was important to not only identify punctuation patterns but also generalize these patterns to an extent that they can be useful in punctuation normalization procedures. This generalization is possible with the comparative method that was suggested in this chapter. Once we understand the effects of normalization, then we can support effective punctuation sensitive search queries.

**Combinational Punctuation Marks**

A combinational punctuation pattern has a specific role in the text. For example, a long repeated use of hyphen may indicate a breaking line. The semantics of the particular punctuation mark must be evidently linked to the length of punctuation marks and unique combinational of punctuation marks. A short repeated sequence of same punctuation mark (e.g., three hyphens) may be used to indicate a pause and emphasize the content.

A series of three consecutive punctuation marks or more can be defined as *triple punctuation marks*. Triple punctuation marks with highest frequency counts are shown in Table 21. In the Gutenberg dataset, 33,535 instances of `period-hyphen-hyphen` were found whereas in Usenet dataset, 117 instances of `period-hyphen-hyphen` were found. Because of double
hyphens, the frequency count of *period-hyphen-hyphen* is substantially higher in the Gutenberg dataset compared to the Usenet dataset.

Table 21

*The Most Common Tripple Punctuation Marks*

<table>
<thead>
<tr>
<th>Punctuation Pattern</th>
<th>Usenet</th>
<th>Gutenberg</th>
</tr>
</thead>
<tbody>
<tr>
<td>.--</td>
<td>117</td>
<td>33,535</td>
</tr>
<tr>
<td>--&quot;</td>
<td>145</td>
<td>11,352</td>
</tr>
<tr>
<td>…</td>
<td>71,273</td>
<td>12,073</td>
</tr>
</tbody>
</table>

Some of the combinational punctuation marks shown in this table are not typical punctuation patterns that would be found in a sentence. Removing these punctuation patterns seemed desirable since they appeared to be extraneous an first examination. In the above case, the use of ellipses was predominant especially in the Usenet dataset. The period-hyphen-hyphen could just be considered as formatting punctuation marks. In general, the longer the punctuation mark series, the greater chance that they are superfluous from the view of text retrieval. Therefore, in normalizing punctuation marks, the length of combinational punctuation marks is one of the useful elements that could be considered.

The frequency count strategy is effective in detecting nonstandard combinational punctuation marks. Combinational punctuation marks may have special domain-specific roles, and accordingly the frequency count could indicate a greater relative percentage difference when two datasets are compared. In general, by employing a frequency count before normalizing such punctuation marks patterns, some of these elements can be reviewed by humans. This process, however, can be difficult since some of the punctuation pattern with lower frequency count might be too numerous to review individually. In any case, without careful examination, the
effect of removing such punctuation marks could be less than optimal because of domain-specific roles they play.

*The Use of Punctuation Marks with Numbers*

Often some of the unique patterns of punctuation marks are associated with numbers. In general, punctuation marks provide a special meaning to the token. In this research, we refer the tokens that contain numbers and punctuation marks as *punctuated numeric tokens*. Table 22 is a summarized version of the type of numeric tokens which were found in both datasets.

**Table 22**

*Examples of Numeric Token*

<table>
<thead>
<tr>
<th>Numeric Type</th>
<th>Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>03/17/1968 or 03-16-1968</td>
</tr>
<tr>
<td>Currency</td>
<td>$1</td>
</tr>
<tr>
<td>Decimal Points</td>
<td>3.14</td>
</tr>
<tr>
<td>Percentage</td>
<td>99%</td>
</tr>
<tr>
<td>General Number</td>
<td>3,100</td>
</tr>
<tr>
<td>Math (Fraction or Division)</td>
<td>½</td>
</tr>
<tr>
<td>Math (Multiplication)</td>
<td>5*2</td>
</tr>
<tr>
<td>Math (Addition)</td>
<td>5+2</td>
</tr>
<tr>
<td>Math (Subtraction)</td>
<td>5-2</td>
</tr>
<tr>
<td>Math (Equal)</td>
<td>2+2=4</td>
</tr>
<tr>
<td>Time</td>
<td>4:00</td>
</tr>
<tr>
<td>Number on Keypad</td>
<td>#9</td>
</tr>
</tbody>
</table>
There were some punctuation marks such as the dollar sign that often is associated with numbers. Since it provides a unique meaning to numbers, deleting such a punctuation mark would cause retrievability problems for a user. As pointed out in Chapter 2, according to Jones (1997), the two broad categories of punctuation marks are source-specific and source-independent. The more source-specific punctuation marks were used with numbers except generally known source-independent punctuation marks such as a comma. For example, the dollar sign and percent signs are regarded as source-specific punctuation marks according to Jones.

In terms of our affix-based punctuation mark category, commas used in numbers are considered as infix-commas, without any blank spaces before and after the character. Because of its affix-based positional characteristics, normalizing a comma that is used in a number requires a different scheme than normalizing commas in typical sentences. As a result, one of the requirements for an effective normalization policy would be to disambiguate such source-independent punctuation marks from sub-lexical punctuation marks. Source-independent punctuation marks are considered inter-lexical punctuation marks and not sub-lexical punctuation according Jones’ (1997) category. See Chapter 2 for the discussion on Jones’ punctuation mark categories.

Table 23 shows various frequency counts related to numeric punctuation marks. In comparison to non-numeric tokens, the number of numeric tokens is relatively low. Overall, in either datasets less than 1% of tokens contained numbers. Although the percentage was relatively low, correctly indexing numeric tokens means that proper punctuation normalization is required even with numbers. In essence, an attempt to normalize punctuation marks meant that frequency counts of specific patterns associated numbers should be examined.
Table 23

<table>
<thead>
<tr>
<th>Basic Statistics of Numeric Punctuation Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric Types</td>
</tr>
<tr>
<td>Punctuated-numeric tokens</td>
</tr>
<tr>
<td>Non-punctuated-numeric tokens</td>
</tr>
<tr>
<td>Numeric infix numeric punctuation marks</td>
</tr>
<tr>
<td>Numeric prefix-punctuation marks</td>
</tr>
<tr>
<td>Suffix-punctuation marks</td>
</tr>
</tbody>
</table>

After browsing the patterns of the period, it was apparent that the datasets contained a vast amount of abbreviations and non-standard words. Thus, it prompted further analysis on various patterns of the periods. Periods were the most predominantly used punctuation marks for abbreviating character strings. The frequency counts of individual punctuation marks and their patterns indicate that the period was heavily used in both datasets.

Punctuation Marks in Abbreviation and Non-Standard Words

Since the language used in the Usenet was informal, this dataset contained non-standard expressions. Compared with the Gutenberg dataset, the Usenet dataset contained a considerable number of abbreviated character strings and non-standard words. The types of abbreviations that are found in the Usenet dataset varied greatly.

Often abbreviated character strings contain a period, but periods are often not used in a predictable pattern. Thus, a typical dictionary approach to identifying abbreviated character strings will work for commonly used abbreviation terms. For example, common abbreviated words such as Dr., Ph.D., etc. and Mr. are typically found in a dictionary. An abbreviated
character string such as \textit{B.C.} can be regarded as an ordinary abbreviated word, and it can be possibly listed in a general dictionary. They often refer to common well-known entities or notions such as \textit{British Columbia, Before Christ,} and \textit{Boston College.} It could be associated with a wider range of multi-part words that start with the letter \textit{B} for the first word and the letter \textit{C} for the subsequent word. Procedures to disambiguate the context of such character strings are beneficial, although such an attempt could be computationally difficult because of the inherent ambiguity associated with the abbreviated character strings.

Table 24 shows the top 20 tokens with \textit{x.x.} punctuation pattern with capitalization ignored. We only considered the cases where only two alphabetical characters have an infix-suffix-period pattern. As was discussed earlier, many variations involving periods were possible, including infix blank spaces. When the terms were calculated based on their appearance in the datasets, common abbreviated terms that can be found in a typical dictionary appeared.

\begin{table}
\centering
\caption{The Top 20 Tokens with \textit{x.x.} Punctuation Pattern}
\begin{tabular}{llll}
\hline
 & Usenet Dataset & & Gutenberg Dataset \\
Frequency Count & Token & Frequency Count & Token \\
\hline
5705 & U.S. & 405 & A.D. \\
806 & i.e. & 288 & B.C. \\
509 & e.g. & 276 & U.S. \\
460 & p.m. & 140 & M.D. \\
272 & U.N. & 127 & M.A. \\
210 & D.C. & 121 & N.Y. \\
187 & P.S. & 109 & D.D. \\
172 & U.K. & 99 & A.M. \\
172 & A.D. & 92 & P.M. \\
123 & L.A. & 83 & i.e. \\
\hline
\end{tabular}
\end{table}
In considering the type of punctuation marks that are involved with abbreviations, it was evident from the frequency count that one of the most common conventions is to use a period right after a single alphabet letter to abbreviate the character string (e.g. B.C.). Another common way to abbreviate character strings was to use the first letter of a character string, followed by other selected letters. The selected letters are mostly consonants from the character string and the abbreviated character string would normally end with a period (e.g., St. and recvd.). Based on browsing the punctuated string pattern, a period that appears after three alphabetical characters or more in lower case letters was more likely a full-stop. Nevertheless, the frequency count result suggests that the length of the individual characters in a token was a useful feature in determining some of the abbreviated character strings.

As mentioned earlier, there are other punctuation marks such as a slash which can be used for abbreviation as well. We should note that other punctuation marks such as slash that involve abbreviations could result in a slightly different punctuation pattern. As pointed out earlier, slashes were used as infixes as in case of I/O and w/o.

Utilizing possible variations of abbreviated character strings could enhance our ability to determine the types of punctuation normalization operations to apply. For example, the variation of the term B.C. could be BC, British Columbia, Boston College. For another example, the variation of PO BOX existed as P.O. BOX, POB, Post Office Box, etc. Again, we ignored capitalization differences for simplicity’s sake. Some of observations can be made regarding these types of variations:

- Some abbreviated character strings do not contain punctuation marks
- Abbreviated character strings sometimes are comprised of multiple tokens
- A punctuation mark such as a period that follows each abbreviation tend to collectively function within the boundary of lexical item
• A period is the most common punctuation mark in abbreviated character strings

The above type of approach can be regarded as a dictionary-based approach since the normalization considers all possible forms of semantically equivalent character strings. In most cases, simply associating all of these types of variations to each other is quite limited because of the non-standard nature of our language. For example, the character string gov’t is a contraction and not a standard abbreviated character string as was we pointed out in Chapter 2. It is difficult to exactly pinpoint a non-standard abbreviated word, but we could simply define a non-standard word as a word that is not found in a basic dictionary. For example, in the Usenet dataset, there was an abbreviated character string M.Kh. This type of abbreviation could be viewed as a less commonly used abbreviation and might not be listed in a typical dictionary. Thus keeping up with every possible character string variations in a general dictionary lookup table might be impractical and imprecise due to homonymy of abbreviated character strings.

In terms of examining abbreviated character strings and contractions, one of the main issues was how to distinguish the role of punctuation marks such as a period and apostrophe. At the same time, a dictionary-based list of known abbreviated character strings could aid in disambiguating the role of punctuation marks to a certain extent. However, to distinguish punctuation mark roles from abbreviated character strings to non-abbreviated character strings, all the elements of surrounding text along with abbreviated term variations must be considered. For this reason, it is desirable to distinguish abbreviated character strings from the rest of the non-abbreviated character strings. Regardless, these types of problems are lexical related issues mainly involving apostrophes, periods, and slashes.

Punctuation Mark Types Based on Referencing Area

We were able to find out that each punctuation mark role can be observed in conjunction
to its distinctive locality in the text. That is, the boundary area can be visually identified by noticing the role of punctuation marks. We call such a boundary area as referencing area. Based upon this punctuation mark locality characteristic, we argue that punctuation marks could be categorized broadly for the purpose of normalization. This notion can be better explained with an example. Figure 15 is an exemplary sentence with punctuation marks (period and comma). Here, we have three types of following punctuation marks:

- Punctuation mark #1 (periods)
- Punctuation mark #2 (comma)
- Punctuation #3 (full-stop)

In Figure 15, each punctuation mark has a distinctive role in a sentence, but for each punctuation mark, the referencing boundary area is different from one to another. The following observations could be made:

- Punctuation marks #1 are only local to the character string B.C.
- Punctuation mark #2 is applicable to two clauses: “This is an ultimate guide to Vancouver Island B.C.” and “the best vacation destination”
- Punctuation mark #3 is applicable to the entire sentence: “This is an ultimate guide to Vancouver Island B.C., the best vacation destination.”

![Figure 15. An exemplary sentence with punctuation marks.](image)

As shown in this example, we observe that each punctuation mark’s referencing area can differ depending on the individual punctuation mark role. The most elementary level of referencing area is confined to just one character. For example, the character normalization of
the letter é as in the character string café is confined to just one character. The affected area could be two characters at most due to the character normalization. In terms of a token, most sub-lexical punctuation marks and inter-lexical punctuation marks are usually confined to one to three tokens. Here, we are only roughly estimating the number of tokens. Yet, technically, if a blank space tokenization is used, the number of tokens could be even longer if we consider an expression such as * end of line * as one lexical unit.

Nevertheless, in the above example, punctuation mark #1 is a period that is confined to just one character string B.C. In case of source-independent punctuation marks (e.g., punctuation mark #2 and #3), the affecting linguistic unit can be greater than just one word.

Considering the referencing areas of individual punctuation marks, a punctuation mark such as a period used with abbreviation is always dependent on a lexical item, whereas a punctuation mark such as a full-stop is not bounded by a lexical item. In this study, punctuation marks could be roughly divided into two major categories: lexical independent (LI) punctuation marks lexical oriented (LO) punctuation marks.

They are important notions in terms of our ability to address and generalize different types of punctuation normalization effects. For now, we need to determine what constitutes LI and LO punctuation marks. In comparison to Jones’ (1997) punctuation categories as pointed out in Chapter 2, LI punctuation marks refer to the following categories: stress markers, delimiting, or and separating. In general, LI punctuation marks may include the following punctuation marks:

- Exclamation mark
- Question mark
- Ellipsis
Semicolon
Dash
Comma
Parenthetical (braces, brackets, parenthesis)
Colon
Quotes
Full-stop

We have to stress the fact that the above categories are general since the punctuation marks could be categorized as LO punctuation marks in certain circumstances. Unlike LI punctuation marks, LO punctuation marks refer to punctuation marks that have functional roles that are associated with lexical items. Usually, the boundaries LO punctuation marks are confined to two to three words. In case of a hyphen (ASCII code decimal #45), the role of punctuation marks could be associated with two words since the hyphen often neighbors two free morphemes. In comparison to Jones’ (1997) category of punctuation marks, it is closer to source-specific punctuation marks and sub-lexical punctuation marks.

The following list is a general guideline and the list is not absolute since a rich set of punctuation marks are available. These are:

Dot
Hyphen
Apostrophe
Decimal point
Asterisk (multiplication)
Carrot (power)
Our categorization punctuation marks, LI and LO, had to be different from Jones’ (1997) and Nunberg’s (1990) punctuation mark categories due to the fact that we are mainly interested in efficiently addressing the effects of punctuation normalization. Not only are the categories of punctuation marks more simplified in a sense, but the categories of punctuation marks are slightly different. We theorize that even punctuation marks that are associated with delimiting and separating roles can have a distinctively defined role at a lexical level.

For example, a comma in most cases is used as LI punctuation mark. Yet, a conventional rule also states that a comma needs to be placed if an individual writes a person's last name before the first name (e.g., *Washington, George*). Since the referencing area is fixed and consistently refers to the lexical item, the last name and first name, in this sense, such use of a comma can be identified as a LO punctuation marks. Thus the notion can be further examined by observing the punctuation marks’ referencing areas as described earlier.
In general, each punctuation mark has certain observable characteristics as far as referencing areas are concerned. More importantly, LI and LO punctuation marks have different referencing characteristics that could be considered in distinguishing the punctuation mark types. The referencing area of a particular LI punctuation mark is not consistently fixed to one particular linguistic unit. That is, the referencing area of LI punctuation marks can vary widely depending on the particular use in a sentence. For example, a quotation mark can be used to quote only one character or an entire paragraph. Another example would be to consider the comma uses using following two sentences:

1. Consequently, we were able to obtain a passport for her travel.

2. This is an ultimate guide to Vancouver Island B.C., the best vacation destination.

In the sentence #1, the comma is referencing the word “Consequently”. However, in the sentence #2, the comma is referencing the entire clause “This is an ultimate guide to Vancouver Island B.C.”. Unlike LI punctuation marks, the referencing area of an LO punctuation mark is fixed to the areas immediately surrounding the lexical items. This means the referencing areas are usually fixed within the boundary of the neighboring character strings.

Also, note that the role of punctuation marks is the major factor in deciding which category a punctuation mark should belong to. For example, if a period is used as an end of a sentence (full-stop) then it is an LI punctuation mark and if it is used within a lexical item, then it is an LO punctuation mark. Since an LI punctuation mark can be used in a context of an LO punctuation mark, inspecting the surrounding text and computationally developing heuristic based rules to distinguish functional roles is important. In this respect, all of the aspects of punctuation mark types and frequency count strategies as were discussed aid in developing such heuristic based rules.
As shown previously, a punctuation mark within the scope of abbreviated character string would be an LO punctuation mark. On the other hand, the ending period would be considered as an LI punctuation mark. A comma used within a number would be an LO punctuation mark and but a comma placed after a word such as punctuation #2 is an LI punctuation mark. Almost all LI punctuation marks can be also used as LO punctuation marks. The reason is that a punctuation mark can be used to create and add a unique semantic sense, or meaning, to a word. Also, as pointed out earlier, there are always special rules. For example, a comma is typically regarded as LI punctuation marks, but when used in the context of a number it can be regarded as an LO punctuation mark (e.g., comma in 100,000).

Furthermore, any punctuation mark can be altered to have a specific meaning in a domain-specific context that a user or system designer might be aware of. For another example, a punctuation mark such as an exclamation mark is always used at a suffix position in a natural language. For this reason, an exclamation is considered as an LI punctuation mark.

On the other hand, in a computer programming language, even an exclamation mark could be modified to possess a unique semantic effect. The exclamation mark is sometimes used as the logical operator *not*. It can be automatically categorized as a LO punctuation mark for this reason. In certain types of genres, punctuation marks can easily belong to multiple categories. Nevertheless, in such cases the system has to rely on heuristics to determine the punctuation mark roles before deciding to normalize such punctuation marks.

Chapter Summary

This chapter provided an analysis of punctuation focusing on two datasets: Usenet and Gutenberg. Based on the punctuation frequency analysis, this chapter demonstrated that
comparing various types of punctuation marks can provide preliminary insights to understanding the issues related to punctuation normalization. By and large, the frequency count strategy was an effective means to expose potential issues that are associated with punctuation normalization.

One of the central approaches taken in categorizing the punctuation marks was to make inferences based on the frequency counts of affix-based token types (prefix, suffix, and infix) and non-affix-based token type (punctuation-exclusive). For example, based on the datasets we examined, a single-letter-infix-slash often appeared as abbreviated character string. In characterizing punctuation mark usages in the datasets, frequency count results related to punctuation marks were reported in this chapter. Often the frequency count strategy helped to determine additional punctuation mark types. The punctuation pattern frequency counts showed that a wide range of punctuation mark types and punctuation patterns exist in each dataset. A myriad of interesting punctuation mark related patterns were discovered through the frequency count approach.

In particular, identification of the affix-based types and frequency count based results were effective means to gain an overall sense of punctuation mark roles. There were several noticeable thematic patterns of analysis that emerged from examining the frequency counts that helped to identify additional punctuation mark and punctuation pattern types: domain specific punctuation pattern, a combinational punctuation mark pattern, punctuation marks in numbers, and abbreviation and non-standard words. In particular, based on punctuation mark referencing areas, all of the discussed punctuation marks were categorized into two major types: LO punctuation marks and LI punctuation marks. These types of punctuation marks and punctuation patterns were used to analyze punctuation normalization phenomena in the subsequent procedures.
CHAPTER 5

THE NORMALIZATION STRATEGY ANALYSIS

This chapter discusses the results of the punctuation normalization analysis. Based on the knowledge accumulated from the previous chapters, this analysis focused on punctuation normalization issues from a holistic perspective. Also, by analytically observing punctuation normalization phenomena, the procedures were used as a means to discover the implications of punctuation normalization. They comprise the following analytical procedures:

- Punctuation normalization types
- Search query types and effects of punctuation normalization
- The normalization demonstration

These procedures were extended from the first part of the punctuation frequency analysis procedures that were presented in the previous chapter. Yet, it was not appropriate to include these procedures as the part of punctuation frequency analysis. Unlike the previous punctuation frequency analysis procedures, these procedures were designed to deal with the issues related to punctuation normalization more directly. At the end of this chapter, the implications of all of these analytical procedures are presented in a form of a discussion.

Punctuation Normalization Types

*Punctuation normalization types* is the first analytical procedure. In this procedure, attempts were made to identify different types of punctuation normalization. In some instances, the types of punctuation normalization were identified by observing the normalizations that could be applied to the text in the datasets. In other cases, characterizing the procedures
that were needed to normalize punctuated terms themselves led to an identification of punctuation normalization strategy.

Punctuation marks were identified as either Lexical Oriented (LO) or Lexical Independent (LI), and the punctuation normalization problems could be approached using these two major punctuation mark types. However, before discussing the punctuation mark normalization types, it is necessary to review the concept of punctuation normalization operations. Punctuation normalization operations and blank space normalization operations are identified as necessary procedures in the framework of punctuation normalization types.

Punctuation Normalization Operations

Punctuation normalization operations refer to the normalization operations more specifically pertaining to punctuation marks. Primitive types of punctuation normalization operations can include:

- Add a punctuation mark
- Remove a punctuation mark
- Replace a punctuation mark

Similarly, two basic blank space normalization operations are:

- Add a blank space
- Remove a blank space

Operations to normalize punctuation marks are tightly integrated with operations to normalize blank spaces. Based on the basic normalization operations of punctuation marks and blank spaces, more combinational operations could be formulated. Operations that involved both punctuation marks and a blank space could include the following:

- Remove-contract - delete a punctuation mark and delete a blank space
• Add-expand - add a blank space and add a punctuation mark

Thus, both punctuation marks and blank spaces could be normalized simultaneously in some normalization operations.

Table 25 shows examples of punctuation normalization operations. As shown in this table, there are punctuation mark affixes: prefixes, suffixes, and infixes. On the right side of the table, the character strings before and after the punctuation normalization operation are shown. The column that is labeled as Token Form after Normalization refers to the changes to the existing token forms after punctuation normalization. Thus, the token form shows a result of changes to the blank spaces and the number of tokens. As before, let us once again assume a blank space based tokenization method in identifying the number of tokens.

It is useful to observe the token form after punctuation normalization since the number of tokens sometimes can be changed. As a result of punctuation normalization, sometimes tokenization points, the points that are used to break a stream of text are also changed. The ability to accurately match user’s search query terms to the terms in the content may depend on how a term is tokenized and indexed. For this reason, we need to observe these types of variations carefully. Based on blank space tokenization, after applying a punctuation normalization procedure, three token forms are possible:

1. Unchanged
2. Single token to multi-token
3. Multi-token to single-token

In the first case, if an exclamation mark ! is removed from the token Yahoo!, then the resulting token would be Yahoo. Since the token Yahoo! is a single token, the punctuation mark has not changed the token form. In the second case, if the hyphen is removed from the multi-part
word *health-care*, based on a blank space tokenization, the resulting tokens would be *health* and *care*. Since two separate tokens are formed, we refer this phenomenon as *single token to multi-token*. For the third case, consider the character string *health care*. If we remove a blank space and concatenate rest of the tokens then it will end up with just a single token. We refer to this phenomenon as *multi-token to single-token*.

Table 25

*Common Punctuation Mark Operations*

<table>
<thead>
<tr>
<th>Punctuation Affixes</th>
<th>Primitive Operations</th>
<th>Blank Space Normalization</th>
<th>Token Form After Normalization</th>
<th>Before Normalization</th>
<th>After Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix</td>
<td>• remove prefix ◊</td>
<td>N/A</td>
<td>unchanged</td>
<td>◊ z</td>
<td>z</td>
</tr>
<tr>
<td></td>
<td>• remove</td>
<td>remove</td>
<td>multi-token to single token</td>
<td>◊ z</td>
<td>◊ z</td>
</tr>
<tr>
<td></td>
<td>• add blank space</td>
<td>add</td>
<td>single token to multi-token</td>
<td>◊ z</td>
<td>◊ z</td>
</tr>
<tr>
<td>Suffix</td>
<td>• remove suffix ◊</td>
<td>N/A</td>
<td>unchanged</td>
<td>z ◊</td>
<td>z</td>
</tr>
<tr>
<td></td>
<td>• add suffix ◊</td>
<td>N/A</td>
<td>unchanged</td>
<td>Z</td>
<td>z ◊</td>
</tr>
<tr>
<td></td>
<td>• add-expand suffix ◊</td>
<td>add</td>
<td>two tokens</td>
<td>Z</td>
<td>z ◊</td>
</tr>
<tr>
<td>Infix</td>
<td>• remove infix ◊</td>
<td>unchanged</td>
<td>single token to multi-token</td>
<td>z ◊ z</td>
<td>z z</td>
</tr>
<tr>
<td></td>
<td>• remove-contract infix ◊</td>
<td>removed</td>
<td>unchanged</td>
<td>z ◊ z</td>
<td>z z (or z)</td>
</tr>
</tbody>
</table>

*Note:* ◊ stands for repeatable punctuation mark(s)

This example demonstrates how an operation to normalize blank spaces can be integrated into a punctuation normalization process. Normalizing blank spaces to a standard form is referred to as *blank space normalization*. Blank space normalization can be simply viewed as a process of normalizing blank spaces to reduce blank space inconsistencies. For example, we
have to consider both character strings such as \( c = a + b \) and \( c = a + b \); not normalizing the inconsistent spacing may cause tokenization issues, and furthermore, it may cause indexing and retrievability problems.

As shown in Table 25, with respect to blank spaces, the token form needs to be taken into account for the cases when the *multi-token to single token* and *single token to multi-token* occur. Using the \( z \) notation, the normalized form \( z z \) could represent two tokens that are separated by a blank space. Also, using the \( z \) notation, the normalized form \( z z \) could represent a resulting single token. For example, *healthcare* is \( z z \). The left side \( z \) represents the word *health* while the right side \( z \) represents *care*. This token form could be expressed as \( z \) since they are all alphanumeric characters. The diamond symbol \( \diamond \) stands for one or more punctuation marks. Notice that in these two types of cases, the number of tokens could change because of the blank space whereas in other cases the number of tokens is unaffected.

As shown in this table, various operations might be required to carry out punctuation normalization. From an operational standpoint, often the decision in regards to suffix and prefix-punctuation marks was whether to remove or leave punctuation marks intact. Typically, these decisions are dependent upon a number of factors such as punctuated character string, punctuation mark types, search query types, etc.

For infix-punctuation marks \( z \diamond z \), the normalization operation was slightly different due to the fact that a blank space could be formed after removing the punctuation mark. In effect, multiple tokens could be formed as a result. For example, a hyphenated token \( z-z \) could be transformed to \( z z \). An alternative option was to remove infix-punctuation marks and then concatenate the remaining tokens together to form a single token. In this manner, the final form of normalized tokens could be represented as \( z z \). Thus, normalizing a hyphenated character...
string by using the remove-contract infix operation would transform *health-care* to *healthcare*. One of the effects of such a remove-contract operation is that a token can become concatenated without containing any hyphen.

For prefix-punctuation marks and suffix-punctuation marks, which are represented as ◊z and z◊ respectively, the number of tokens is not altered as a result of punctuation normalization. However, adding blank spaces between the alphanumeric characters and the punctuation marks would alter the number of tokens.

In general, there were some distinctive characteristics that could be observed with affix-based punctuation mark types. These include the following:

- LI punctuation marks were more closely associated with suffix-punctuation marks and prefix-punctuation marks
- LO punctuation marks were more closely associated with exclusive punctuation marks
- Most infix-punctuation marks could be regarded as LO punctuation marks

Also, as suggested earlier, an interesting finding from observing the above punctuation mark operations was the effect of punctuation normalization with respect to blank spaces. That is, blank spaces could be characterized as a non-punctuation element that often requires some type of normalization. As in the case of $\bar{c} = a + \bar{b}$, blank space normalization is particularly required for tokens that contain blank spaces.

Further, punctuation normalization differs depending on the particular type of punctuation mark. Since punctuation marks could be differentiated depending on their roles and their referencing areas, punctuation normalization could be categorized into the following:

- LO punctuation normalization
- LI punctuation normalization
**Lexical Oriented (LO) Punctuation Normalization**

In applying LO normalization, the affected area is limited to lexical items. That is, LO punctuation normalization usually does not affect more than two or three tokens. For example, for infix-hyphens and infix-apostrophes, typical normalization procedures are applied at a lexical level. Because of this, all character strings that are associated with infix-punctuation marks are usually subject to LO punctuation normalization. Almost all infix-punctuation marks are LO punctuation marks since LI punctuation marks are normally placed in prefix or suffix positions. For example, all hyphenated character strings that contain apostrophes are subject to LO punctuation normalization because of their infix positions.

The effects of normalizing LO punctuation marks, however, can be attributed to a particular form of the character string. We have shown how one form of a character string could be semantically related to another form of the character string where the only difference is the punctuation mark. For example, the character strings *B.C.* and *BC* could be semantically equivalent to each other. Other forms of string variations should be considered if an aggressive approximate searching query needs to be facilitated. Possible equivalent character strings to *B.C.* could include variations such as *British Columbia, Before Christ,* and *Boston College.* In determining the contextual usage of the abbreviated character strings, LO punctuation normalization could be coupled with a Named Entity Recognition type of algorithm.

Table 26 is an example of LO normalization that shows possible normalized punctuated character strings. Some of the affix-based punctuation mark types are in the *Punctuation Mark Types* column. Based on observing the un-normalized and normalized character strings, it is possible to formulate punctuation normalization operations in such a
way that punctuation normalization operations will produce corresponding normalized character strings.

Table 26

**LO Normalization Example**

<table>
<thead>
<tr>
<th>Un-Normalized String</th>
<th>Punctuation Mark Types</th>
<th>Type of String Form Before Normalization</th>
<th>Possible Normalization Operations</th>
<th>Possible Normalized Character Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>b/c</td>
<td>infix-slash;</td>
<td>non-standard abbreviation</td>
<td>none</td>
<td>b/c</td>
</tr>
<tr>
<td></td>
<td>single-letter-word</td>
<td></td>
<td>dictionary lookup</td>
<td>because</td>
</tr>
<tr>
<td>B.C.</td>
<td>infix-period;</td>
<td>abbreviation</td>
<td>remove suffix-period &amp; remove-contract infix-period</td>
<td>BC</td>
</tr>
<tr>
<td></td>
<td>suffix-period</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>suffix-plus</td>
<td>typical word in a dictionary</td>
<td>none</td>
<td>C++</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dictionary lookup</td>
<td>Cplusplus</td>
</tr>
<tr>
<td>l-o-u-d-e-r!!</td>
<td>infix-hyphen;</td>
<td>non-standard</td>
<td>remove-contract infix-hyphen &amp; remove-suffix exclamations</td>
<td>louder</td>
</tr>
<tr>
<td></td>
<td>suffix-exclamation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90-100</td>
<td>infix-hyphen</td>
<td>number (range)</td>
<td>add internal blank spaces (before &amp; after)</td>
<td>90 - 100</td>
</tr>
<tr>
<td>cat -v</td>
<td>prefix-hyphen</td>
<td>computer command</td>
<td>none</td>
<td>cat -v</td>
</tr>
<tr>
<td>c = a - b</td>
<td>punctuation-exclusive</td>
<td>mathematical equation</td>
<td>remove-contract blank space</td>
<td>c=a-b</td>
</tr>
<tr>
<td>health-care</td>
<td>infix-hyphen</td>
<td>multi-word</td>
<td>remove hyphen;</td>
<td>health care</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>none</td>
<td>health-care</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>remove-contract</td>
<td>healthcare</td>
</tr>
<tr>
<td>re-read</td>
<td>infix-hyphen</td>
<td>prefix “re”</td>
<td>remove-contract</td>
<td>re-read</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>unchanged;</td>
<td>Shi’aite</td>
</tr>
<tr>
<td>Shi’ite</td>
<td>infix-apostrophe</td>
<td>person/people</td>
<td>remove-contract apostrophe</td>
<td>Shi’ite</td>
</tr>
</tbody>
</table>

Comparing an un-normalized punctuated character string with normalized punctuated character string is often helpful in this regard. Similarly, we need to consider possible string
variations first. For example, for the first character string \textit{b/c}, the possible normalized character strings are \textit{b/c} and \textit{because}. Note that these are not only variations of character strings that could be produced.

In this example, several types of normalization operations can be formulated:

- Remove [options]
- Add [options]
- Remove contract [options]

The valid options could be affix-based punctuation marks or a particular type of blank space (e.g., internal blank space before and after a punctuation mark). There is no exact formula to create this type of list as we have only used a limited number of character strings. Also, as discussed later, some of the normalization procedures depend on the type of search queries. The point of this example is to illustrate various punctuation normalizations that need to be applied in producing normalized character strings.

\textit{LO Normalization and Orthographic Comparison of Strings}

A step before normalizing LO punctuation marks is to perform an orthographic comparison of character strings. The two most basic criteria of orthographic features that can be selected are punctuation marks and alphanumeric characters. Although alphanumeric characters of character strings are a factor, we focused on the punctuation marks part as a basis for its potential application. We call this distance value as a \textit{Punctuated String Distance (PSD)} in this study. A PSD can be defined as a score of how a non-punctuated character string and a punctuated character string are similar. The basic objective for defining a PSD value is to incorporate it into a punctuation normalization procedure.
Figure 16 shows pertinent components related to an orthographic comparison procedure.

Some of the procedures shown in this figure are fairly common to a typical text retrieval system.

![Figure 16. LO normalization and orthographic comparison environment.](image)

However, some of the procedures are unique. Punctuation normalization, which could be viewed as a one of the type of text normalization as pointed out in Chapter 1, is placed in this figure. It is divided into LI and LO punctuation normalization. Collectively, they can be considered as text normalization. The calculation of PSD can be further incorporated into an LO punctuation normalization process. The LI and LO punctuation normalization might have to be processed iteratively in some cases since some character strings might be surrounded by LI punctuation marks such as a comma. For instance the token ‘B.C.’, which contains a pair of single quotation marks, needs both LI and LO punctuation normalization procedures to normalize a number of
punctuation marks: the single quotes, comma, and the periods. The normalization procedures may have to be processed iteratively until all punctuation marks are correctly processed for the normalization purposes. Moreover, the normalization procedure might need some type of precedence rule in terms of which normalization occur first. As in the case of character string ‘B.C.’, the LI punctuation marks, typically, appear more outside than the LO punctuation marks.

As is discussed later, disambiguation of a punctuation mark role is important in applying correct punctuation normalization procedures. Thus, LI and LO punctuation normalization often need to be examined collectively. As a result, building an index would depend on the character strings produced by LI and LO punctuation normalization procedures.

In Figure 16, the orthographic comparison procedure, shown in the shaded circle, is an optional component that could be coupled with the LO punctuation normalization procedure. For this orthographic comparison procedure, the PSD can be used. Measuring the PSD is discussed later. Once a PSD value is obtained, an LO punctuation normalization procedure can take the quantified value into consideration using punctuation mark variations between two character strings.

During the process of building indexes, conditional rules can be used check and see if the character strings meet certain orthographic distance criteria. The figure also shows indexing and matching procedures as a critical components surrounding the LO punctuation normalization process. As pointed out in Chapter 2, various indexing schemes exist. One of the options is to allow tree-like structure (e.g., suffix-tree) that would not require indexing individual words. Yet, word-based indexing, or lexical-item based indexing, was assumed in this study.

As shown in the figure, distance measurement can be applied to both un-normalized character strings and normalized character strings. Notice that an un-normalized punctuated
character string and a normalized punctuated character string could be differentiated by a number of punctuation marks and spellings of character string. Since the spellings only consider the standard alphabets, we could optionally expand the notion to include alphanumeric characters. This means orthographic measurement can be devised by using a number of orthographic features such as:

- Punctuation mark differences
- Alphanumeric differences of character strings

However, for simplicity’s sake, in applying orthographic measurement for our case, we will only examine character strings that have equal set of alphanumeric characters. Thus, misspellings of alphanumeric characters are not considered (e.g, helth-care instead of health-care). This problem is often referred to as a string similarity problem. More specifically, it is considered as an approximate string matching problem (Navarro, 2001). To be semantically equivalent, it is not necessary to have an equal set of alphanumeric characters. For example, the character strings B.C. and BC have an equal set of alphanumeric characters. We only focused on investigating the distance involving punctuation marks. Yet, the applicability of considering both punctuation mark differences and different spellings into a single distance measurement needs to be investigated.

Despite the numerous available string similarity metrics, we decided to investigate the potential application of Levenshtein distance (1966) and the Stoilos distances (Stoilos et al, 2005). First, the Levenshtein algorithm is popular, and Levenshstein has many applications as Wichmann and Grant (2012) pointed out. The idea behind the Levenshtein distance is that the distance can be calculated by counting the number of steps to transform from one character string to another. Each operation – insertion, removal or substitution – is considered as a step.
That is, Levenshtein distance is the least-sum of the costs that are required to transform one character string into another. Here, we only suggest the possibility of applying in the context of measuring the PSD.

For example, let us calculate the Levenshtein distance between the two terms \( B.C \) and \( BC \). The minimum editing steps that are required to change one character string into the other are the following:

- \( B.C \rightarrow BC \) (remove infix-period) +1 edit
- \( BC \rightarrow BC \) (remove suffix-period) +1 edit

By adding up all these values, 2 can be obtained. This value is the Levenshtein distance between the two terms \( B.C \) and \( BC \). Note that this example does not sufficiently illustrate the Levenshtein distance algorithms as it does not show rigorous algorithmic details. The most convenient ways to implement Levenshtein algorithms is to use what is known as dynamic programming. An extended description of the algorithm and an application example of the Levenshtein distance algorithm can be found in Haldar and Mukhopadhyay (2011).

To apply PSD correctly, we need to examine how punctuated character strings and non-punctuated character strings are semantically equivalent, semantically close, or semantically different. First, let us consider the character strings that contain only single punctuation mark:

- \( \text{gov't} \) versus \( \text{govt} \)
- \( \text{don't} \) versus \( \text{dont} \)
- \( 15 \text{ in.} \) versus \( 15 \text{ in} \)
- \( \text{Yahoo!} \) versus \( \text{Yahoo} \)

For each of the above paired-character strings, the Levenshtein’s distance is 1. Although the above listed normalized character strings, which are shown on the right side, are not in a
grammatically correct form, the character strings maintain a close or semantic sense in relation to the character strings that are shown on the left side.

The Levenshtein distance itself can be normalized. By normalizing the Levenshtein distance, it incorporates the length of the character string, and, at the same time, is more convenient to compare the distance value with other pairs of character strings. Yujian and Bo (1995) described their version of the normalized Levenshtein distance metric; the normalized distance formula is:

\[
\text{Norm. Lev}(S_1, S_2) = \frac{\text{Lev}(S_1, S_2)}{\text{Max}(\text{length}(S_1), \text{length}(S_2))}
\]

\text{Lev} is the Levenshtein distance value. Unlike the Levenshtein distance value, the normalized Levenshtein distance as shown as \text{Norm. Lev}(S_1, S_2) has a value from 0 to 1. The maximum value is 1, which indicates that there is no match between two character strings of characters \( S_1 \) and \( S_2 \). However, if we assume an equal set of letters, the value of 1 is not obtained. The distance value closer to 0 indicates that the distance is extremely close.

The second distance formula is the Stoilos distance measurement (Stoilos et al., 2005). The Stoilos commonality measurement was used to measure the PSD distance since it seemed useful in incorporating a number of matching substrings into formulating the string distances. The Stoilos commonality measurement was provided as the following:

\[
\text{Comm}(S_1, S_2) = \frac{2 \times \sum \text{length}(\text{Max_component_sub}_i)_i}{\text{length}(S_1) + \text{length}(S_2)}
\]

\text{Comm}(S_1, S_2) is the Stoilos commonality measurement value of string \( S_1 \) and \( S_2 \).

\text{Max_component_sub}_i is the individual components of matching substring whereas \( i \) represents the index value of each string set. Let us consider two character strings: \text{gov’t} and \text{govt}. We have \text{Max_component_sub}_1 as 3 because of subcomponent \text{gov}. We have

173
Max\textsubscript{components} as 1 because of the sub-component $t$. The length of $gov't$, length($gov't$), is 5; the length of $gov't$, length($govt$), is 4. Therefore, Comm($gov't$ and $govt$) is $8/9$, which is 0.89. Stoilos commonality measurement can have a value ranging from 0 to 1. However, in contrast to the normalized Levenshtein distance, the higher value indicates the condition that the two character strings are orthographically closer, whereas the lower value indicates that the two character strings are more orthographically distant.

Table 27 is an example that shows the PSD using the normalized Levenshtein distance and Stoilos commonality measurement. Based on the PSD values, each pair of the character strings shown in this table is semantically close. It can be observed that as the length of the character string increases, the value of the Stoilos distance increases slightly. On the other hand, the value of the normalized Levenshtein distance decreased somewhat proportionally. From this example, there seem to be three different types of correlations:

- Between the length of the character strings and the Stoilos distance,
- Between the length of the character strings and the normalized Levenshtein distance
- Between Stoilos distance and the normalized Levenshtein distance

The key question is: *What do the distance values indicate and how can we interpret the distance value?* In some cases, punctuation marks in a punctuated character string could have more vital roles in maintaining the original meaning of the character strings. In such cases, when such a punctuation mark is removed, the meaning of the changed character string and the original character string are completely different. The changed character string can depict completely different sense from the original sense of the character string. For example, consider the following pairs:

- $C++$ versus $C$
- $A-B$ versus $AB$
- \( a=b+c \) versus \( abc \)

Table 27

*The Punctuated String Distance (PSD) Example*

<table>
<thead>
<tr>
<th>Un-Normalized Character String (U)</th>
<th>Normalized Character String (N)</th>
<th>Levenshtein Distance</th>
<th>Max (U,N)</th>
<th>Normalized Levenshtein Distance</th>
<th>Stoilos Distance (Commonality)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gov’t</code></td>
<td><code>govt</code></td>
<td>1</td>
<td>5</td>
<td>0.2</td>
<td>0.89</td>
</tr>
<tr>
<td><code>don’t</code></td>
<td><code>dont</code></td>
<td>1</td>
<td>5</td>
<td>0.2</td>
<td>0.89</td>
</tr>
<tr>
<td><code>15 in.</code></td>
<td><code>15 in</code></td>
<td>1</td>
<td>6</td>
<td>0.17</td>
<td>0.91</td>
</tr>
<tr>
<td><code>Yahoo!</code></td>
<td><code>Yahoo</code></td>
<td>1</td>
<td>6</td>
<td>0.17</td>
<td>0.91</td>
</tr>
</tbody>
</table>

The left side shows the un-normalized character strings, while the right side shows the normalized character strings. Notice that the length of character strings is relatively short in this example. According to Sproat et al. (2001), and as mentioned in Chapter 2, all the above punctuated character strings are non-standard.

In contrast, let us consider another example with following pairs of character strings:

- *thousand ++* versus *thousand*
- *health-care* versus *health care*
- *Shi’ite* versus *Shiite*

As can be seen in this example, all of pairs of character strings indicate a sense of semantic “closeness”. Marton et al. (2009) pointed out that two character strings are considered to be semantically close if there is a lexical semantic relation between them. As they noted: “relation may be a classical relation such as hypernymy, troponymy, meronymy, and antonymy, or it may be what have been called an ad-hoc non-classical relation.” (Marton et al., 2009, p.776). Their definition of the semantic “closeness” seems like a reasonable one to adopt for this study.
In comparing these examples, as the length of character string increases, the punctuated character strings seem to become closer to the context of the non-punctuated character string. This observation is simply based on these instances.

Table 28 shows the PSD comparison of short character strings and long character strings. In this table, punctuation normalization was applied to both short character strings and long character strings.

Table 28

*The PSD Comparison Example*

<table>
<thead>
<tr>
<th>Short Character String</th>
<th>Un-Normalized String (U)</th>
<th>Normalized String (N)</th>
<th>Levenshtein Distance</th>
<th>Max (U,N)</th>
<th>Normalized Levenshtein Distance</th>
<th>Stoilos Distance (Commonality)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C++</strong></td>
<td><strong>C</strong></td>
<td>2</td>
<td>3</td>
<td>0.67</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td><strong>A-B</strong></td>
<td><strong>AB</strong></td>
<td>1</td>
<td>3</td>
<td>0.33</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td><strong>a=b+c</strong></td>
<td><strong>abc</strong></td>
<td>2</td>
<td>5</td>
<td>0.4</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>average:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.67</td>
<td>3.67</td>
<td>0.47</td>
<td>0.68</td>
<td></td>
</tr>
<tr>
<td>Long Character String</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>thousand++</strong></td>
<td><strong>thousand</strong></td>
<td>2</td>
<td>10</td>
<td>0.2</td>
<td>0.89</td>
<td></td>
</tr>
<tr>
<td><strong>health-care</strong></td>
<td><strong>healthcare</strong></td>
<td>1</td>
<td>11</td>
<td>0.09</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td><strong>Shi’ite</strong></td>
<td><strong>Shiite</strong></td>
<td>1</td>
<td>7</td>
<td>0.14</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>average:</td>
<td></td>
<td>1.34</td>
<td>9.34</td>
<td>0.14</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>relative % difference between two groups</td>
<td>20%</td>
<td>60.71%</td>
<td>69.29%</td>
<td>25.72%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The normalized Levenshtein distance value is much higher for the first group (short character string) than for the second group (long character string). In the opposite manner, the Stoilos measurement showed that the value was much higher for the second group (long character string)
than for the first group (short character string). In terms of the average Levenshtein distance, there was not a substantial relative percentage difference between the two groups. However, in terms of the normalized Levenshtein distance, the relative percentage difference between the two groups indicated a considerable amount of difference between the two groups. The relative percentage difference was 69.29% for the normalized Levenshtein distance and 25.72% for the Stoilos distance.

Based on the Stoilos distance and the normalized Levenshtein distance, it is useful to rank the items for the purpose of observing their patterns. Based on a lowest to highest value ordering, the ranked list based on the normalized Levenshtein distance is the following:

1. **health-care** versus **healthcare**
2. **Shi’ite** versus **Shiite**
3. **thousand++** versus **thousand**
4. **A-B** versus **AB**
5. **a=b+c** versus **abc**
6. **C++** versus **C**

Similarly, we could rank the pairs of character strings according to the Stoilos commonality distance values. In contrast to above distance values, the highest value indicates that the pairs are semantically close to each other. Based on a highest to lowest value ordering, the ranked list is the following:

1. **health-care** versus **healthcare**
2. **Shi’ite** versus **Shiite**
3. **thousand++** versus **thousand**
4. **A-B** versus **AB**
5. \( a=b+c \) versus \( abc \)

6. \( C++ \) versus \( C \)

Comparing the two ranked list, no difference between the two ranked orders can be found. Yet the ranked order could be slightly different depending on which of the two methods was used. Regardless which method was used, the longer character strings ranked higher than the shorter character strings. However, we cannot be certain if this pattern can hold since we have generalized the pattern based on just few example cases.

Semantically, the following observations can be made. If the PSD value is very low, in the case of the normalized Levenshtein method, the semantic effects of punctuation normalization seems to be minimal. On the other hand, if the normalized Levenshtein value is very high, then the semantic effects of punctuation normalization seem to be substantial. However, these observations need to be investigated by conducting additional experiments involving a much larger number of character strings. More calculations should be done that are similar to Table 28. Moalla et al. (2011) investigated the applicability of Levenshtein and Stoilos distances for spell-checking applications. They considered using a threshold value (e.g., \( Lev < 0.2 \) AND \( Stoilos > 0.8 \)) for correcting misspelled words. Although the specific application was different, such a threshold value could be useful in building an index or establishing matching rules in a text retrieval system component.

So far we have explored the notion of the PSD by adopting the existing methods that measured orthographic distances. In general, orthographic distance value might indicate the likelihood of character strings being related to each other. If such measurements are properly designed and applied, it could be incorporated into a punctuation normalization process.
However, the applicability of PSD measurement still needs to be investigated further to produce a meaningful pattern. In doing so, we need to investigate the problems using various types of tokens and different length of tokens. It should be stressed that in spite of the usefulness of the PSD, applying the distance measurement to understand the semantic effects of punctuation normalization is still limited. Instead, the PSD should be considered as a combinable attribute to punctuation normalization as we have mentioned earlier.

In a larger sense, although the notion of the PSD was mainly discussed, the more important point of this discussion was the general applicability of orthography as a whole. The PSD metric is only one type of measurement that seems to be useful for measuring punctuation mark differences. We could consider measuring the distance values involving alphanumeric-based character strings in conjunction with punctuation marks. The application of such a metric, however, needs further investigation.

Once the terms are normalized, matching can take place based on four types of combinations:

- Non-punctuated character strings in the search query to punctuated character strings in the content
- Non-punctuated character strings in the search query to non-punctuated character strings in the content
- Punctuated character strings in the search query to non-punctuated character strings in the content
- Punctuated character strings in the search query to punctuated character strings in the content

_Lexical Independent (LI) Normalization_

While LO normalization applies to LO punctuation marks, LI normalization only applies to LI punctuation marks. In Chapter 4, we discussed how we can distinguish LO punctuation
marks from LI punctuation marks. Since LI punctuation marks have a separating role and a 

delimiting role, the LI punctuation mark role is not bounded by any lexical items.

Table 29 is an example showing results of LI normalization. A sentence was selected 

from the Gutenberg dataset to describe the difference between LO normalization and LI 

normalization. The example only deals with small set of punctuation marks. As shown in the 
table, LI punctuation normalization was applied by simply removing the punctuation marks. The 
results of LI punctuation normalization differ depending on the two types of tokenization:

- Blank space based tokenization
- Lexical item based tokenization

After applying LI punctuation normalization operations, there is a need to keep track of 
punctuation normalization positions. By doing this, adverse effects of punctuation normalization 
could be avoided. These positions are indicated by the double vertical line ||. Such positions 
could be physically tagged by a retrieval system as a part of applying LI punctuation 

normalization. In the table, the complete views of all the available tokenization options are not 

shown. To simplify this example, the two tokenization options are the blank space based 
tokenization and the word-based tokenization. In this example, in option #1, the string 14 in. 
is separated into two tokens 14 and in., whereas in option #2, the string 14 in. is not separated into 
two tokens. In option #2, we assume that it is a text retrieval system’s responsibility to detect 
and tokenize the terms into proper lexical item. Let us assume that such tokenization could be 
aranged since the details of the tokenization process are not provided here. The effects of LI 
punctuation normalization is also discussed later in this chapter.
Table 29

*An Example of LI Punctuation Normalization Results*

<table>
<thead>
<tr>
<th>Un-normalized Sentence</th>
<th>She is completely belted with 14 in. armor, with a 15 in. backing, and has the central battery armored with plates of 91/2 in. in thickness.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option #1 Blank space based Tokenization with LI Normalization</td>
<td>She is completely belted with 14 in. armor with a 15 in. backing and has the central battery armored with plates of 91/2 in. in thickness.</td>
</tr>
<tr>
<td>Option #2 Lexical Item Based Tokenization with LI Normalization</td>
<td>She is completely belted with 14 in. armor with a 15 in. backing and has the central battery armored with plates of 91/2 in. in thickness.</td>
</tr>
<tr>
<td>Normalized Punctuation Marks</td>
<td><strong>Remove comma-suffix after “armor”</strong>&lt;br&gt;<strong>Remove comma-suffix after “backing”</strong>&lt;br&gt;<strong>Remove period-suffix after “thickness”</strong></td>
</tr>
</tbody>
</table>

*Note: || indicates LI normalization points where as | indicates tokenization points.*

Once again, the role of each punctuation mark in this sentence is worth noting. In this example, commas and periods are considered LI punctuation marks. However, both periods and commas can be LO punctuation marks when they are used with numbers (e.g., decimal point and a comma in 1,000). In essence, disambiguation of punctuation marks is a pre-requisite before applying punctuation normalization procedures.

**Search Query Types and the Effects on Punctuation Normalization**

*Search query types and effects on punctuation normalization* is the second analytical procedure that was incorporated in this study. In this procedure, attempts were made to identify search query types along with their effects on punctuation normalization. We noticed that the effects of punctuation normalization could be described in terms of different types of search queries. First, for simplicity’s sake, let us only consider short search queries (2 to 3 words). To
investigate punctuation normalization problems, the user’s search queries were categorized into two types:

- Punctuated queries versus non-punctuated queries
- Exact search queries versus approximate search queries

A punctuated query is a query that contains at least one punctuated token. In contrast, as the name suggests, a non-punctuated query is a query that does not contain any punctuated token. It was possible to further categorize a punctuated query into LI punctuated query and LO punctuated query. A punctuated query can contain both LI punctuation marks and LO punctuation marks.

In identifying the types of search queries, the notion of exact search query and approximate search query had to be examined. Given a variety of user’s search queries, by conceptualizing and comparing these two search types, we can analyze the search results even further for the purpose of analyzing punctuation normalization effects. Furthermore, we can identify the type of scenario in which the user can maximize the particular search feature. Let us assume that an exact search query requires a search function that is exact, while an approximate search query is possible with an approximate search function.

The notion of “exactness” could be subjective and imprecise due to the consideration of punctuation marks in the text. The question that could be raised was: Does “exact” mean only a continuous stream of alphanumeric characters, or does “exact” actually mean consideration of all strings including punctuation marks as they appear? Since this requires a decision, in this study, let us assume that an exact search query requires an exact sequence of all of characters: alphanumeric characters and punctuation marks.
Also, there was a question of “exactness” in terms of matching different units of search query. A typical search query could be comprised of one or more units:

- Sub-words (e.g., affixes or free-morphemes in a hyphenated-lexical item)
- Blank space based tokens
- Lexical items
- Phrases (e.g., more than two blank space based tokens)

The question that can be asked is: *Does matching only subpart of the above text unit comprise an exact search query?* In this study, to be considered as an exact search query, the entire unit of the exact search query specified by a user needs to be exactly matched as well. If a user’s search query is specified at a phrase length level, then only matching the phrase is considered as an exact match. On the other hand, if the user’s exact search query has only one token (e.g., *B.C.*), then the matching needs to match the particular token to be considered as an exact search query.

Another issue that needs to be determined is capitalization. The next question that can be asked is: *Does case insensitive search still comprise an exact search query?* Although it could be defined in either way as suggested earlier, we chose to disregard case sensitivity in this study. Since we assume an exact search is not case sensitive, an exact search query will produce case insensitive results. In practice, however, case sensitive searches should be optionally provided by a text retrieval system.

So far we defined an exact query as a query that is intended to match terms as exactly as they appear. All punctuation marks would be matched as they appear under this definition. As we stated, the only caveat to this definition is that the capitalization of search query is not treated differently. In general, exact searches offer a user with optional searching capabilities. This
alternative searching method can be sought by a user when a default search fails to meet the user’s expectation.

Unlike exact search queries, approximate search queries are intended to approximately match the character strings in the content side to the character strings in the search query. For example, if a user enters a search query as $BC$, we may match both $BC$ and $B.C.$ by assuming an approximate searching method. In addition, $B.C.$ could be semantically matched with other forms of lexical items such as *British Columbia* and *Boston College*. However, for the purpose of a discussion, let us set aside the issue of matching character strings that are semantically equivalent but contains different spelling.

For exact search queries, the most pertinent question is: *What are the effects of matching punctuation marks for exact search queries?* For the exact search queries, the punctuation normalization requirement is straightforward, especially for the definitions we have just provided. It merely has to support the matching exactly as it appears. Thus, punctuation marks need not be normalized. For an exact search query type, possible normalized character strings are shown in expanded list of character strings in some cases since we would want to able to generate more matches than the conservative approximate searching

As shown in Table 30, the search query types can be punctuated or non-punctuated. To support case insensitive searches, we have “folded” text cases to be all lower cases. By applying case-folding, terms may appear as case insensitive during the search and retrieval. Regardless, there is an obvious need for a punctuated search query in some cases. For character strings such as $C++$, it is difficult to imagine a situation where a user might want to type *plusplus* instead of the plus signs $++$ as a potential search query term. There seems to be no other reasonable way to specify the query other than by using the punctuated character string $C++$.  

184
Table 30

*Exact Search Queries and Normalized Character Strings*

<table>
<thead>
<tr>
<th>Search Query Type</th>
<th>Search Query Example</th>
<th>Normalized Character Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punctuated queries</td>
<td>cat–v</td>
<td>cat–v</td>
</tr>
<tr>
<td>in exact searching</td>
<td>shi’ite</td>
<td>shi’ite</td>
</tr>
<tr>
<td></td>
<td>ne’ve</td>
<td>ne’ve</td>
</tr>
<tr>
<td></td>
<td>b.c.</td>
<td>b.c.</td>
</tr>
<tr>
<td></td>
<td>c++</td>
<td>c++</td>
</tr>
<tr>
<td></td>
<td>thousand++</td>
<td>thousand++</td>
</tr>
<tr>
<td>Non-punctuated queries</td>
<td>cat command</td>
<td>cat command</td>
</tr>
<tr>
<td>in exact searching</td>
<td>shiite</td>
<td>shiite</td>
</tr>
<tr>
<td></td>
<td>never</td>
<td>Never</td>
</tr>
<tr>
<td></td>
<td>british columbia</td>
<td>british Columbia</td>
</tr>
<tr>
<td></td>
<td>thousand</td>
<td>Thousand</td>
</tr>
</tbody>
</table>

On the other hand, the options for approximate search can vary widely. For an approximate search, we do not have to match the search queries as they appear. Also, there is an issue of what constitutes an approximate search. The notion of an approximate search query implies that what constitutes an approximate search query has to be pre-determined by a designer and carried out by a system. A designer has to define the notion and provide the details with respect to determining the type of matching process it needs to execute the search requests. We could assume an approximate search is a default search and consider an exact search query is an optional search feature that needs to be manually selected by the user in a text retrieval system.

In general, the notion of an approximate search query can be expanded even more to include conservative approximate searching and aggressive approximate searching. To facilitate these types of searches, punctuation normalization has to be applied according to required matching functions based on these two broad categories. In conservative approximate searching, searching is closer to the notion of an exact search that we just discussed. Under this type of scenario, the punctuation normalization operations can be viewed as a less aggressive. It needs
to avoid making various attempts to normalize punctuation marks that might produce non-
relevant search results. Generally, it should carry out fewer attempts to find other character
strings that might match the user’s search term semantically. It is a stricter type of search that
avoids matching various string patterns as well.

The idea behind this method is to match only possible known matching patterns under a
stricter set of specifications. Because semantically equivalent lexical terms are not pursued, the
system only needs to match both the lexical terms in the search side and in the content side in
close string distance. This means the PSD value should be low as in the case of normalized
Levenshtein distance value.

On the other hand, in an aggressive approximate searching situation, searching can be
more relaxed so that more related terms are produced by a pattern matching technique. More
possible matched character strings are pursued under this approach. Even spelling of character
strings might be different, but they are semantically equivalent. As we discussed in the previous
chapters, gains by means of punctuation normalization are made at the expense of retrieving
more irrelevant results. Once again, distinguishing approximate searches in terms of
conservative or aggressive is useful, but a system designer needs to define what constitutes a
conservative search and what constitutes an aggressive search method. Once the proper
decisions are made, character strings should be normalized, configured, and indexed according to
these types.

Based on the notion of conservative or aggressive search query types, Table 31 shows a
few examples of possible LO normalized character strings. Again, there is no precise way to
generate the possible matching character strings. This is our own version of hypothetical
normalized character strings that illustrate the notion of approximate search query types.
Table 31

*Approximate Search Queries and LO Normalized Strings*

<table>
<thead>
<tr>
<th>Search Query Type</th>
<th>Search Query Example</th>
<th>Conservatively Normalized Strings</th>
<th>Aggressively Normalized Character Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Punctuated</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>queries</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>in approximate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>searching</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cat -v</td>
<td>cat -v</td>
<td>cat-v, cat v, cat +v, cat</td>
<td>command, cat commands</td>
</tr>
<tr>
<td>shi’ite</td>
<td>shi’ite, shiite</td>
<td>shi’ite, shiite, shiites, shi’ite,</td>
<td>shiitah</td>
</tr>
<tr>
<td>ne’ve</td>
<td>ne’ve, neve</td>
<td>ne’ve, neve, never</td>
<td></td>
</tr>
<tr>
<td>b.c.</td>
<td>b.c., bc</td>
<td>b.c, b.c. -bc- b/c, british,</td>
<td>columbia, boston college, before christ</td>
</tr>
<tr>
<td>c++</td>
<td>c++</td>
<td>c++, cplusplus</td>
<td></td>
</tr>
<tr>
<td>thousand++</td>
<td>thousand++, thousand</td>
<td>thousand++, thousand, 1000,</td>
<td>thousandplusplus</td>
</tr>
<tr>
<td>cat minus v</td>
<td>cat minus v option</td>
<td>cat minus v option, cat -v, cat</td>
<td>options, cat command, cat -v options</td>
</tr>
<tr>
<td>option</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>never</td>
<td>never</td>
<td>never, ne’ve</td>
<td></td>
</tr>
<tr>
<td>shiite</td>
<td>shiite, shi’ite</td>
<td>shi’ite, shiite, shiites, shi’ite,</td>
<td>shiitah</td>
</tr>
<tr>
<td>cplusplus</td>
<td>cplusplus, c++</td>
<td>cplusplus, c++</td>
<td></td>
</tr>
<tr>
<td>british columbia</td>
<td>british columbia</td>
<td>bc, b.c., british columbia,</td>
<td>(british columbia), b/c</td>
</tr>
<tr>
<td>bc</td>
<td>b.c., bc</td>
<td>b.c., b.c., bc, (bc), british</td>
<td>columbia, before christ, boston college</td>
</tr>
<tr>
<td>thousand</td>
<td>thousand, thousand++</td>
<td>1000, thousand, thousand++,</td>
<td>thousand++, -thousand-, <em>thousand</em></td>
</tr>
</tbody>
</table>

Non-punctuated queries in approximate searching
In comparing the set of normalized character strings as shown in this table, we can obtain a sense of what the conservative punctuation normalization and aggressive punctuation normalization might be.

Whether the specific character string is aggressively normalized or conservatively normalized is subjective to a degree. Similar to the previous table, punctuated search queries and non-punctuation search queries are shown in this table. Notice that the aggressively normalized character strings would have longer list of character strings, as in some cases we would want to be able to generate more matches than the conservative approximate searching method.

As pointed out earlier, the decisions regarding what constitutes a conservative approximate search query and an aggressive approximate searching query must be made. Facilitating these two distinctive search query types is optional. Such defined terms are only useful for understanding the issues and comparing their features to each other. This study suggests that the approximate search query could be categorized into these types.

As shown in Table 31, in some cases the use of a punctuated term in a search query is as vital as a good discriminating character string. Sometimes punctuation marks are desirable for punctuated character string such as `cat -v` and `C++`. This type of use is particularly suitable for an exact search query. These types of LO punctuation marks are highly effective discriminators in this searching situation. We noticed that in this situation, not using any punctuated query could produce unpredictable results. Generally, it would be extremely difficult to search instances of these punctuated character strings without using any type of punctuated search terms.

While both `cat -v` and `C++` are punctuated character strings, a distinction can be made between these two types. The distinction is necessary since it can help us to formulate a specific approach needed to normalize these types of punctuated character strings. `C++` is a well-known
word that can be found in a lexicon, whereas *cat -v* is a domain-specific character string that is not expected to be found in a lexicon.

Normalizing LO punctuated character strings such as *C++* and *cat -v* could be approached in two ways. The first approach is to use a customized look-up dictionary. Using a dictionary-based approach can be implemented by using what we can refer to as *exceptional routines*. Exceptional routines can skip these types of character strings from being normalized, or exceptional routines can be used to carry out more targeted punctuation normalization operations only for specific terms such as *C++* and *cat -v*. The disadvantage for a less well-known expression such as *cat -v* is that it would take more efforts to identify this type of string. In such cases, we would have to rely on methods such as conducting frequency counts to identify these types of domain-specific character strings. Maintaining the exceptional routines and the customized look-up table will ensure that special character strings are not treated as typical words in the text. However, this approach has its disadvantages since it is difficult to detect and maintain a huge amount of character strings.

The second approach is to examine the lexical-level punctuation patterns that exist in the content side and carefully craft the punctuation normalization rules accordingly. For example, when the `++` punctuation pattern is used in a punctuated character string, it is considered as LO punctuation marks. Let us consider `++` in the character string *C++* in more detail. These are known facts about these combinational punctuation marks:

- `++` at the suffix position
- `++` could be a unique discriminator since `++` is a combinational punctuation mark
- `++` are LO punctuation marks
Instead of using a dictionary approach, we could use a more straightforward approach by considering all of these facts. Also, we could perform frequency counts based on the suffix ++ pattern. However, unlike the example involving the character string C++, in some cases it only adds additional senses to the existing character string. For example, the character string thousand++ is not substantially different from the meaning of the character string thousand. There was an instance of the punctuated character string thousand++ in the Usenet dataset. The ++ was used to indicate that the quantity is much larger or more than thousand, and this is an unconventional invented character string. The question is whether we want to allow matching thousand ++ with thousand for an approximate search case.

In the previous case, C and C++ could be considered as semantically close. These two character strings refer to distinctively different, yet related, computer languages. For a typical search term, we probably do not want to substitute one term for another since they have distinctively different meanings. In comparison, the sense of distinction is somewhat less strong for the pair of character strings thousand++ and thousand. Thus, if the search query term contains the character string thousand, it might be acceptable to match with the character string thousand++ as well. In this type of case, a punctuated character string that ends with ++ should not be simply normalized in most cases. This type of punctuation pattern seems to offer an essential discriminating feature because of its unique punctuation pattern.

One the other hand, we want to normalize the punctuation marks for some of these types of character strings so that only alphanumeric portions will be properly matched. Perhaps allowing an approximate match approach for some cases such as these, indexing the character string two separate ways (e.g., thousand and thousand++), which was discussed earlier, might be an option. In addition, employing an orthographic comparison approach in conjunction with
alphanumeric characters might be useful. Matching such a type of punctuated character strings with non-punctuated character strings also depends on particular search features (e.g., conservative or aggressive).

In other cases, the operations that are required to normalize LO punctuation marks and support the matching of strings may seem fairly straightforward. This is especially true for cases where the LO punctuation marks do not add strong discriminatory sense to the character string. For example, if character strings such as Shi’ite appear in the content side, this term is a punctuated query that could be normalized as the non-punctuated term Shiite. In this case, the semantic effect of the apostrophe seems so subtle that Shi’ite and Shiite are substantially equivalent and merely signifies stylistic differences. This is also true for most hyphenated lexical items (e.g., health-care vs. healthcare). In such a case, users should be able to retrieve all of the known variations of the character string, if proper normalization procedures were applied in both the content side and the search side.

We also have to consider the user’s behavioral characteristics in using search queries. In some situations, users may only consider punctuated character strings as an alternative search attempt. Users might neglect to specify punctuation marks, although users might expect a retrieval system to return some type of relevant search results. For example, a user might simply type cant instead of can’t as it is faster and more convenient to type without using an apostrophe. We do not know to what extent the users avoid using punctuation marks for the efficiency of time and effort. However, although the application was different, as pointed out in Chapter 2, the efficiency of time and effort was observed as a reason for not using punctuation marks during text messaging. Similarly, a user’s behavioral characteristics might be linked to formulating search queries.
To support a conservative approximate matching and exact search queries, exact search queries, however, requires much simpler punctuation normalization operation is required for LI punctuation marks. Notice that LI punctuation marks need a different treatment from LO punctuation marks. LI should be used to disambiguate the role of LO punctuation marks, but this is a separate task. Yet, generally users have a little need to search contents directly by specifying LI punctuation marks. The same is applicable to the content side as well. For example, if the term \textit{B.C.}, with a comma, appears as a term in the content, then removing the ending comma is more appropriate. The reason is that the ending comma is an LI punctuation mark.

\textit{Lexical Oriented (LO) Punctuation Normalization Effects}

The analysis indicated various effects of LO punctuation normalization. The effects of LO punctuation normalization can vary significantly depending on the type of LO punctuation normalization. Some of the issues related to LO punctuation marks were described in the previous chapter. As pointed out in the previous chapter, infix-punctuation marks are LO punctuation marks. Additionally, for the purpose of an in-depth discussion, let us consider two types of the LO punctuation normalization effects:

- Effects of a search query involving infix-hyphens
- Effects of a search query involving abbreviations

\textit{The Effects of a Search Query Involving Infix-Hyphens}

We examined the effects of a search query involving infix-hyphens. Some of the character strings discussed in this section were already examined in the previous chapter.
However, in this section, different types of user’s search queries are being factored into analyzing punctuation normalization. There are three types of infix-hyphenated situations:

- Affix-based word
- Compound word
- Abbreviated or non-standard character string

In the first case, one way to retrieve character strings that are associated with a prefixed word, such as *re-read*, was the following:

- Retrieve character string that contain *reread*
- Retrieve character string that contain *re-read*

To find all search text segments that contain *reread* and *re-read*, the results could be merged together after running independent search queries. An alternative option is to normalize both *reread* and re-read into just one form. The latter option eliminates the overhead associated with the merging step. As a result, normalizing the character string is a much more efficient means to process such a search query.

To elaborate the normalization options, two types of punctuation normalizations for the word *re-read* are possible. These are:

- *re-read* to *reread*
- *reread* to *re-read*

Either of these options is suitable. By normalizing these words, text segments that contain both *reread* and *re-read* can be retrieved by a user. To simplify the matching process, the user’s search query needs to be normalized in the same manner.

In the second case, one way to retrieve short text segments that contain instances of a compound word such as *health-care* is to normalize the lexical item into one of the following:
Preferably, all of these character string variants should be normalized into just one of the above. By normalizing the punctuation marks equally on the search side and on the content side, any variants of the character strings will be matched with each other. First, we need to define the notion of a sub-word for this study. A sub-word in this study is any type of main word fragment that consist of contiguous string. Generally a sub-word is separated by one or more punctuation marks that is typically in alphanumeric sequence. For example, for the lexical item health-care, two character strings, health and care, are individually considered as sub-words as they are separated apart by a punctuation mark. Later in this chapter, indexing and retrieving sub-words are examined in conjunction with the punctuation marks.

The third case involves hyphens in abbreviated character strings or non-standard character strings. For an abbreviated character string that contains an infix-hyphen, let us consider the character string cu-ft which could reside in the content. For this character string, it is possible to search using character strings patterns such as:

- cuft
- cu-ft
- cu ft
- cubic feet
- cubic-feet

Similar to compound word normalization, the remove-contract operation is acceptable in this case. If a remove-contract operation is applied, the original character string cu-ft becomes cuft.
Since the character string was abbreviated, more possible string variations are linked to the character string \textit{cu-ft}.

In this case, since the normalized hyphen character string is \textit{cuft}, this character string still has the same meaning as \textit{cu-ft}. However, disregarding capitalization, the character string \textit{cuft} could be also \textit{Colegio Universitario Fermin Toro} or \textit{Children of the Universe Foundation} according to Thefreedictionary Website (cuft, 2013). For these types of character strings, the character string \textit{cu-ft}, instead of \textit{cuft}, might not be used as appropriate abbreviation for neither of these entities because of the infix-hyphen. While \textit{cuft} could be viewed semantically the same as \textit{cu-ft}, it could be also associated with other contextual uses.

In other cases of non-standard character string, abbreviated character string may create a character strings that has an entirely different meaning if a remove-contract operation is applied. Thus the semantic effects of the normalizing infix-hyphens could be more risky in this type of case. For instance, changing \textit{a-b} to \textit{ab}, the remove-contract operation might not be a suitable solution as it may destroy the original meaning of the non-standard word. Therefore, it is difficult to generalize and develop an effective LO punctuation normalization procedure for non-standard words.

In comparing all of the above cases, each type of character strings with infix-hyphens should have different LO punctuation normalization procedures. In the case of the word \textit{re-read}, we would not expect a user to specify \textit{re read} as a search query term since the word is a bound morpheme. In the case of the character string \textit{cu-ft}, \textit{cu} could be regarded as an abbreviated character string rather than an affix-based word. There is a difference between \textit{re} and \textit{cu} in this case. That is, since \textit{cu ft} is regarded as a searchable character string, it should be indexable as a
potentially matchable character string. For the compound word *health-care*, the character string *health care* are an acceptable alternative representation since it is a compound word.

All of the above variations should be either indexed as one single unit or supported as a phrase searching. Although partially indexing sub-words of infix-hyphenated character strings such as *re-read*, *health-care*, and *cu-ft* makes much more sense, it is questionable whether users need to be able to search *re* or *cu* as an individual character string. In this case, *re* is a simply an affix and *cu* is a less distinguishable abbreviated character string when used as an individual term.

In the case of a short non-standard hyphenated term, such as *a-b*, it is also highly questionable whether we want to index individual sub-words. The first choice is to index the elements as *a*, *b*, and *a-b*. The second choice is to just index *a-b*. Also, we have to consider the fact that blank space variations might exist for the character string *a-b* which increases the level of complexity in supporting this type of search.

*Matching Sub-Words and Neighboring Character String*

Earlier, we discussed that LO punctuation normalization can produce different token form (e.g., *z-z* to *z z*). Accordingly, we discussed the need for indexing and searching sub-words as a result of LO punctuation normalization. In turn, we further discussed the types of effects on supporting exact and approximate searches.

Before continuing with the notion of indexing sub-words, there is a caveat to the notion of indexing sub-words as mentioned in Chapter 2. That is, if a full-text is treated as a non-broken stream of characters without word boundaries, then the underlying structure to support sub-string matching is possible and word-based indexing is not required. Therefore, indexing
sub-words becomes somewhat a different issue since finding any part of sub-strings are naturally supported by a substring indexing method such as a *suffix-tree*.

In this study, we mainly focused our attention on analyzing the punctuation normalization issues based on blank space based and word based tokenization. To index every term in corpus, it would be more natural to proceed with the following order: tokenization, punctuation normalization, and indexing. However, the process of tokenization and punctuation normalization might require an iterative procedure in the sense that appropriate tokenization points must be determined in conjunction with punctuation marks. For example, a suffix comma usually indicates a definitive tokenization points. These types of issues are further discussed later in this chapter.

In terms of lexical-level patterns, if a user’s search query pattern is a hyphenated string pattern *z-z*, the matching pattern could be *z-z*, *z z*, or *zz* pattern. Table 32 summarizes the possible normalized forms. The possibilities that are shown in this table are not exhaustive since some exceptions can be expected. In addition, retrieving all of the relevant hyphenated character strings requires a careful examination of the individual type of hyphenated character strings. Such a process might involve looking up existing dictionary based terms. At the same time, it may involve algorithms to distinguish one form of a hyphenated character string to another. In the case of abbreviations and non-standard text, features such as the length of alphanumeric characters should be an important attribute in conjunction with terms in a dictionary.

In terms of indexing, different options are possible. These decisions are dependent on the individual model of text retrieval that a system designer decides to adopt. In the case of a compound word, indexing sub-words as bound-free morphemes is desirable since a user may
only search for a sub-word. For example, *heath* and *care* could be individually indexed. In the
case of the word *b/c*, both *b* and *c* could be considered as sub-words.

Table 32

**Infix-Hyphen Normalized Pattern**

<table>
<thead>
<tr>
<th>Types of Hyphenated Character String</th>
<th>Allowable Normalized Pattern</th>
<th>Avoidable Normalized Pattern</th>
<th>Sub-word Indexing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affix-Based Word</td>
<td>• z-z</td>
<td>z z</td>
<td>index bound-free morpheme; optional for affixes</td>
</tr>
<tr>
<td></td>
<td>• zz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compound Word</td>
<td>• z-z</td>
<td>none</td>
<td>index both bound-free morpheme</td>
</tr>
<tr>
<td></td>
<td>• z z</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• zz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abbreviation and Non-Standard Form of Character String</td>
<td>• z-z</td>
<td>depending on the type of abbreviation and non-standard character string</td>
<td>depending on the type of abbreviation and non-standard character string</td>
</tr>
<tr>
<td></td>
<td>• z z</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In some cases, indexing sub-words individually could be viewed as an important problem that needs to be processed for the efficiency of text retrieval. Sub-word indexing of only the non-standard parts of the character string might depend upon its applications. Let us consider the earlier example of Jiang and Zhai’s (2007) again. As mentioned in Chapter 2, the character string *(MIP)-1alpha* was used in the biomedical domain. For such a character string, searching a sub-word pattern for a character string such as *(MIP)-1alpha* could be useful. In this case, we could index the character string using two sub-words: *MIP* and *1alpha*. Indexing sub-words works for compound words since compound words contain two bound-free morphemes. On the other hand, for affix-based words, an affix could be disregarded and the remaining free morpheme could be indexed.

To observe the effects of punctuation normalization of infix punctuation marks, it is particularly important that we consider the phenomena with following factors:
• Types of searches (exact vs. Approximate)
• Types of indexing (e.g., sub-word vs. Compound-word)
• Types of character strings (e.g., affixes, compound words, non-standard)
• Types of punctuation marks

We also observe that sometimes sub-words and neighboring character strings had to be considered in examining the effects of normalizing infix-punctuation marks. Let us consider the following phrase: important health-care provider. To examine the effects holistically, a slight modification to z-notion was suitable. For an infix-hyphenated lexical item, the left side of the character string is represented as \( z \) and the right side character string is represented \( z! \). Thus, the lexical item health-care could be represented as \( z-z! \). In addition, the token on the left side of \( z \) is represented by \( z \) whereas the token on the right side of \( z! \) is represented as \( z! \). In sum, for the sub-word or the character string, we have the following representation:

- **important**: \( z \)
- **health**: \( z \)
- **care**: \( z! \)
- **provider**: \( z+ \)

Collectively, using the modified z-notion, the phrase important health-care provider can be simply represented as \( z \cdot z-z! \cdot z+ \). Notice that there are blank spaces between each element that should not be ignored in this phrase representation. Let us examine a case where an exact search method is being used to retrieve short text segments. Table 33 shows an exact search query scenario with various search query terms. Here, one method indexes sub-words while another method does not index sub-words. In comparison, sub-word indexing retrieves every type of
search term combinations. On the other hand, if a method without sub-word indexing is used, some terms in search queries would not be match to the lexical items in the content side.

Table 33

*Exact Search Query Involving a Compound Word (health-care)*

<table>
<thead>
<tr>
<th>search query</th>
<th>z notation</th>
<th>Matched String (Sub-Word Indexing)</th>
<th>Matched String (Without Sub-Word Indexing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>health</td>
<td>z</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>care</td>
<td>z</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>important</td>
<td>z</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>provider</td>
<td>z</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>health-care</td>
<td>z-z</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>important health</td>
<td>z-z+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>care provider</td>
<td>z-z</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>important health-care provider</td>
<td>z-z-z-z-z+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

*Note:* The plus sign (+) indicates a matched condition, whereas minus sign (-) indicates unmatched condition.

There is a clear difference how these two methods treat sub-words from matching the terms. While sub-word indexing is suitable for approximate search, in this type of situation, it may not be a suitable option for exact search queries. By indexing the sub-words, the neighboring character strings were retrieved as a phrase. Although this example may not seem to be a problem, in some situations, particular problems involving non-standard character strings, could have a noticeable effect.

For example, Table 34 shows an exact search query scenario involving a non-standard word. In this table, search terms such as *ne* and *er* would be matched using the sub-word indexing scheme. The character string *ne* and *er* are short abbreviations which could stand for a number of lexical items. For example, the character string *ne* could stand for *Nebraska* and the
word er could stand for emergency room. In both cases, the context of these lexical items is much different. In this sense, unlike in the case of Table 33 which involves a compound word, the adverse effect of sub-word indexing involving non-standard character string could be more apparent.

Table 34

**Exact Search Query Involving a Non-Standard Word (ne'er)**

<table>
<thead>
<tr>
<th>search query</th>
<th>z notation</th>
<th>Matched String (Sub-Word Indexing)</th>
<th>Matched String (Without Sub-Word Indexing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ne</td>
<td>z</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>er</td>
<td>z</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>youth</td>
<td>z</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>left</td>
<td>z</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ne'er</td>
<td>z'z</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>youth ne</td>
<td>z, z'z</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>er left</td>
<td>z, z'z</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>youth ne'er left</td>
<td>z, z'z, z'z</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

*Note: The plus sign (+) indicates a matched condition, whereas minus sign (-) indicates unmatched condition.*

**The Effects of a Search Query Involving Abbreviations**

As discussed in the previous chapter, abbreviations, which often contain periods, can appear frequently in a corpus. To explore the punctuation normalization problem associated with abbreviated character strings, variants of abbreviated character strings had to be investigated. A relevant issue was the ways in which users might express the particular abbreviated character string. In examining the effects of punctuation normalization, first we need to consider what type of character strings in a punctuated query users might use. Then, we could develop a punctuation normalization scheme accordingly.
Consider the abbreviated character string \textit{B.C.}. As shown in Table 35, by running the exact search query, various segments that contain the term in the Usenet dataset could be examined. Let us assume a situation where the ability determines the context of the character string \textit{B.C.} from the search query alone is quite limited. For instance, a user might have initiated only this character string as a search query.

By using the exact search query method, we also retrieved few lexical items that contained the string \textit{BC}. When the string \textit{BC} was used as a search term, more search results were produced than when using \textit{B.C.} as a search term. It is difficult to determine whether not having periods in abbreviated character strings produces more results in general.

Table 35

<table>
<thead>
<tr>
<th>Term</th>
<th>Character String and Neighboring Strings</th>
</tr>
</thead>
</table>
| \textit{B.C.} | • 3300 \textit{B.C.}  
• the \textit{B.C.} Court |
| \textit{BC} | • 2370 \textit{BC Nov 20}  
• Social Club in Van \textit{BC}  
• \textit{BC} = \{ (bi*ej, (BiCj)) \}  
• CASE NO. \textit{BC} 245271  
• deborah \textit{BC} |

More importantly, there were contextual differences between the two terms. In some instances, \textit{B.C.} and \textit{BC} were used in the same context. The string \textit{B.C.} was used as in the case of 3300 \textit{B.C.} and \textit{BC} was used in the case of 2370 \textit{BC Nov 20}. In both cases, the meaning of the character strings is \textit{Before Christ}. However, there were additional instances in which these two abbreviated form of character strings could be equivalent. For example, in the case of \textit{BC} = \{ (bi*ej, (BiCj)) \}, the string \textit{BC} was used in the context of a mathematical formula. In \textit{CASE NO.}
BC 245271, BC was used in the context of referencing indexes (e.g., BC1, BC2, BC3, etc.). In these types of instances, B.C. could not be substituted for BC.

Assuming that we have limited means of knowing the context of the character string, if a user decides to use the string BC then the notion of the character string could be associated with any of the following:

- All instances of the character string that is semantically equivalent to the abbreviated string B.C.
- A mathematical formula
- A referencing index, such as a product or part number

Some other effects of normalizing punctuated abbreviated character strings are noticeable from this example. In this example, without considering any contextual usage, only a string based matching method was used to retrieve text segments. The result indicates that resolving such lexical ambiguity problem requires additional disambiguation procedures. Thus, the problems involving lexical ambiguity can be noticeable in Table 35. Two terms with identical spellings can be associated with the classic homonymy and polysemy problem (Krovetz, 1997). However, because we are dealing with punctuated character strings, the definitions need to be established more clearly. Nevertheless, more than one meaning is associated with the abbreviated string B.C. As shown in this table, B.C. was used to indicate Before Christ, and to indicate a street name.

This case effectively demonstrates that punctuation marks provide additional contextual meaning or additional senses to a character string. However, ambiguity of the character strings can be high if the contextual usage is not clearly identifiable. Theoretically, since this term is in the form of an abbreviated string, the abbreviated strings B.C. and BC could stand for extremely
large numbers of character strings. The domain knowledge is critical as it could narrow the possibilities.

Similarly, \( b/c \) is not a typical dictionary-based word as it can be used loosely to represent various lexical items. This character string could stand for \textit{because} or \textit{b divided by c}, etc. Using an exact search method, the string matching of the string \( b/c \) is the primary concern. Identifying the contextual role of the string \( b/c \) is not required as the user has the responsibility to correctly utilize the search feature. The character strings require additional clues from the neighboring character strings or punctuation pattern to determine the correct context usage as previously discussed. Even so, from the text retrieval point of view, it is unclear whether it is useful to retrieve all variants of the abbreviated character strings at this point. In a larger sense, judicious decisions are required to normalize punctuation marks in abbreviated character string.

\textit{The Semantic Effect Categorization}

The semantic effects of LO punctuation normalization can be observed at a lexical-level. In most cases, since punctuation marks clarify or add distinctive context to the text, by applying the LO punctuation normalization procedures, it can easily produce additional false meaning of the character string. Most of the times, observing semantic effects can be done by comparing an un-normalized term that corresponds to a normalized term.

We can derive the semantic effects of punctuation normalization by first establishing these criteria:

- Two character strings are semantically equivalent
- Two character strings could be semantically equivalent depending on the contextual usage
- Two character strings are totally unrelated
It is useful to conceptualize the effects of LO punctuation normalization by examining the conditions between an un-normalized character string with the normalized character string. Figure 17 is a conceptual configuration of LO normalization effects. This figure depicts the semantic relationship between an un-normalized character string and a normalized character string. Each circle represents a semantic space of the un-normalized character string or the normalized character string. These diagrams are abstract representations of the contextual usages of character strings. The shaded area represents semantic space where these two types of character strings intersect, implying that two terms can have the same meaning, or the same sense. Three configurations shown in this table can be used to represent the degree of similarity between un-normalized character strings and normalized character strings.

![Figure 17](image)

*Figure 17. Conceptual configuration of LO normalization effects.*

The condition when two character strings – un-normalized character strings with the normalized character strings – are semantically equivalent is shown as box (a). The two circles are perfectly aligned to each other. This condition represents the ideal state of punctuation normalization. There is no variation of semantic variations of character strings as the side effects of LO punctuation normalization, if any, are minimal.
The condition when two character strings could be semantically equivalent but differ in the contextual usage is shown as box (b). In most cases, box (b) should represent a “semantic closeness” condition since some sense or meaning of the two character strings overlap with each other. The overlapping area of two circles shown in box (b) can differ depending on a particular situation. That is, the two circles might move more closely together, or they could move away from each other slightly. Since the overlapping area could vary depending on the nature of the character string, the effects are difficult to predict beforehand. Moreover, depending on the user’s intended context of the character string, the normalized character string could fall either into the shaded area or the non-shaded area.

The circles in box (c), on the other hand, do not overlap at all. This is the condition where the two character strings are totally not related and there is no overlapping shaded area as shown. The search results would produce mostly un-related items for the user. This condition should be avoided as it would be difficult to meet the end user’s need.

Table 36 shows some of the effects of LO normalization. Sometimes these categories are less obvious due to the imprecise nature of interpretation. Sometimes punctuation normalization produces less conventional form of lexical items. For example, *non combatants* appears to be less conventional as *non* is an affix and not a free-morpheme. However, collectively the character string provides a sufficient sense as *non-combatants*. Since all of these are exemplary cases, further examination of the context would be useful in most cases in assessing a semantic equivalence. As in this case, the sense of non-standard character strings that contain LO punctuation marks is more difficult to generalize without further investigation.
Table 36

*An Example of LO Normalization Effects*

<table>
<thead>
<tr>
<th>Un-normalized Character String</th>
<th>Normalized Character String</th>
<th>Unconventional Lexical Item Form?</th>
</tr>
</thead>
<tbody>
<tr>
<td>white-space</td>
<td>blank space</td>
<td>yes</td>
</tr>
<tr>
<td>shi’ite</td>
<td>shiite</td>
<td></td>
</tr>
<tr>
<td>non-combatants</td>
<td>non combatants</td>
<td></td>
</tr>
<tr>
<td>Equivalent Depending on the Contextual Usage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>No</td>
<td>yes</td>
</tr>
<tr>
<td>B.C.</td>
<td>B C</td>
<td></td>
</tr>
<tr>
<td>9-1-1</td>
<td>911</td>
<td></td>
</tr>
<tr>
<td>B.C.</td>
<td>British Columbia</td>
<td></td>
</tr>
<tr>
<td>#9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>King, James re-read</td>
<td>King James re read</td>
<td>yes</td>
</tr>
<tr>
<td>Totally Not Related</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cat -v</td>
<td>cat v</td>
<td></td>
</tr>
<tr>
<td>b-c</td>
<td>bc</td>
<td></td>
</tr>
</tbody>
</table>

*Lexical Independent (LI) Punctuation Normalization Effects*

The effects of LI punctuation normalization are more apparent and straightforward. The most distinctive effects of LI punctuation marks are the association of neighboring character strings as a single phrase. Earlier we pointed out that LI punctuation marks could be normalized by removing them as long as the punctuation normalization was tagged by a system. Essentially by marking these points, when a phrase search comes along, we are able to skip the neighboring two character strings from matching the search. Table 37 is an example of LI punctuation normalization effects.
Table 37

An Example of LI Normalization Effects

<table>
<thead>
<tr>
<th>Without LI Normalization (Un-normalized) Text:</th>
<th>Option #1 Text After Simply Removing Punctuation Marks</th>
<th>Option #2 Text After LI Normalization by Tagging the Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnny Justice, King of this living Universe, wants to have a word with every American in America</td>
<td>Johnny Justice King of this living Universe wants to have a word with every American in America</td>
<td>Johnny Justice</td>
</tr>
</tbody>
</table>

Exact Search Query “Justice King”
- + -

Exact Search Query “Johnny Justice”
- + +

Exact Search Query “living Universe”
- + +

Note: The plus sign (+) indicates a matched condition, whereas minus sign (-) indicates an unmatched condition.

According to Option #2 in this table, if we apply punctuation normalization by removing commas, then we end up disregarding LI punctuation marks entirely. This means the character strings Justice King are matched and returned to the user. However, in a strict phrase search, this type of matched results should not be performed since the user who is searching for the phrase might be interested in only Justice King as it exactly appears and not Justice, King. Thus, this effect is particularly notable with phrase type of search queries. Therefore, Option #2 approach could be used to avoid two adjacent character strings from matching as an exact single phrase.
The Punctuation Normalization Demonstration

*The punctuation normalization demonstration* is the third and last analytical procedure of this study. In this section, we report brief results of our punctuation normalization demonstration. The normalization demonstrations were designed initially, but this was re-considered due to the fact that sufficient punctuation normalization analysis had already been performed qualitatively. Providing a general framework to address these issues was more important for this study since many concepts had to be defined and categorized according to their punctuation normalization characteristics.

The demonstration that could be conducted could vary widely. Since observations were entirely based on the texts from the corpus, only one demonstration, in particular, was selected for the purpose of gaining further insights into the punctuation normalization effects in certain areas of this research. As mentioned earlier, there are more difficult issues involved with LO punctuation normalization since the punctuation mark usage varies significantly. First, selecting a punctuated character string was necessary. Although a search term could have been randomly selected, we decided to examine certain punctuated character string that contained two alphabetical characters followed by a period. The top 10 character strings that met this criterion in the Usenet dataset were the following:

- me.
- Mr.
- up.
- do.
- on.
- is.
Because these were the most frequently used character strings, they can be easily found in a typical dictionary. Although it was not verified, more than likely, some of the unabbreviated character strings in the above list might contain full-stops, or end of sentence periods, rather than suffix periods. To investigate more interesting cases, we attempted to find slightly less frequently used character strings from the Usenet dataset. Randomly selected character strings with two alphabetical characters followed by a period were the following:

- **ID.**
- **Ah.**
- **AA.**

Viewing the stand-alone punctuated character strings brought about uncertainty regarding the context of the character strings. Therefore, we used these character strings to retrieve short segments from the two datasets to view the context of the character strings. Most likely, this indicated some possibilities of how the character string was used.

Table 38 shows a few lines of the text that contain the character strings with suffix-period. When these types of character strings were used to match character strings, there were notable similarities between these types of abbreviated character strings that contained a suffix-period. As expected, all of the periods after two letters indicated the end of each sentence, marking a full-stop. For the instances that involved **ID.**,**Ah.**,**AA.**, after inspecting the text, the periods were categorized as LI punctuation marks. In contrast, in the previous case, the suffix-period of a character string such as **Mr.** was not the full-stop.
### Table 38

*Examples of Text Segments with Two-Letter-followed by Period*

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Punctuated String</th>
<th>Example of Short Text Segments</th>
<th>Comments</th>
<th>Possible Character String Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usenet</td>
<td><em>ID.</em></td>
<td>...<em>pay to get a new ID.</em> ...<em>with your verifiable ID.</em> ...<em>Idaho Falls, ID.</em> ...<em>with the Product ID.</em> ...<em>close to my real ID.</em></td>
<td>• ID as an identification • ID as Idaho</td>
<td>Abbreviation</td>
</tr>
<tr>
<td>Ah</td>
<td><em>Ah. Oh, darn.</em></td>
<td><em>Ah. But how can you be... Ah. Generally &quot;affordable housing&quot;... Ah. I misspoke, you're right... Ah. Hines, the wayward...</em></td>
<td>• Express emotion of relief or regret.</td>
<td>Normal Word</td>
</tr>
<tr>
<td>AA</td>
<td><em>..for my involvement in AA.</em> ..sponsor and AA. ..but I do recognize they in AA. ..realized the &quot;New Age&quot; into AA. ..don't got no business in AA.*</td>
<td>• Alcoholic Anonymous</td>
<td>Abbreviation</td>
<td></td>
</tr>
<tr>
<td>Gutenberg</td>
<td><em>ID.</em></td>
<td><em>n/a</em></td>
<td>• no record found</td>
<td></td>
</tr>
<tr>
<td>Ah</td>
<td><em>KING: Ah.</em></td>
<td><em>Ah. God in heaven, is...</em></td>
<td>• Express emotion of relief or regret.</td>
<td>Normal Word</td>
</tr>
<tr>
<td>AA</td>
<td><em>NOTE AA. Has <em>Macbeth</em>. sack and ale call themselves AA. Style AA. For Kodak Ball.</em></td>
<td>• Indexing Letters • Person</td>
<td>Non-Standard</td>
<td></td>
</tr>
</tbody>
</table>

*Note:* The ellipses indicate the continuation and do not indicate the actual returned ellipses in the content.

These tokens further demonstrate that it is difficult to determine whether the ending period contains a full-stop or not based on only viewing the character string. Additional clues by examining surrounding words and pattern are needed in this case where the period is found with words that are short (e.g., 1 to 2). To disambiguate the role of periods, methods such as the ones suggested by Grefenstette & Tapanainen (1994) could be adopted. In any case, once the period's
role is disambiguated, a proper selection of punctuation normalization can be applied. Broadly speaking, we need to select LO punctuation normalization, LI punctuation normalization, or some combinations of both.

Although most of the character strings appeared in both datasets, occasionally, the sense of the each character string that were used was slightly different. In the Usenet dataset, AA was used as an abbreviation for the lexical item *Alcoholics Anonymous*. However, in the Gutenberg dataset, no instances of the lexical item *Alcoholics Anonymous* were found. As shown in the table, the token *AA* was used to represent some type of indexing reference number or a person. In the Gutenberg dataset, there were instances of texts that contained *ID*. Furthermore, to find out the meaning of the character string in the Gutenberg dataset, the term *ID* without the ending period was searched. The retrieved results of the term *ID* showed that this character string could stand for identification or a person’s name.

Since two abbreviated character strings were identified from this example, a useful demonstration to conduct is abbreviated character strings with an infix-period and a suffix-period. Since the character strings *ID* and *AA* were identified as proper abbreviated character strings, we searched these character strings against the Usenet dataset and the Gutenberg dataset. However, we used infix-period and suffix-period this time. Table 39 shows examples of abbreviated character strings with infix-suffix-period.

The results showed that there was not a significant contextual usage difference between the two datasets except for the string *A.A.* As shown in this table, this character string was used as an abbreviation for a person’s name. No record was found of it as an abbreviated character string for *Alcoholics Anonymous*. The reason for the missing record is probably due to the fact that the Gutenberg dataset consisted mostly of older books. The results suggest that a more
targeted approach is desirable for certain types of collection. However, let us keep in mind that this is only one type of instance and involved a specific type of abbreviated string patterns.

Table 39

*Examples of Two Letter Abbreviated Words with Periods*

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Punctuated character string</th>
<th>Example of Short Text Segments</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usenet</td>
<td><em>I.D.</em></td>
<td>this <em>I.D.</em> Kit will give authorities license and secondary <em>I.D.</em></td>
<td>• ID as an identification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Branson, Missouri <em>I.D.</em> U.S.A. - Child <em>I.D.</em> program an</td>
<td>• ID as Idaho</td>
</tr>
<tr>
<td></td>
<td><em>A.A.</em></td>
<td><em>A.A.</em> membership is a desire <em>A.A.</em> membership is a desire <em>A.A.</em> membership is a desire</td>
<td>• Alcoholic Anonymous</td>
</tr>
<tr>
<td></td>
<td></td>
<td>An <em>A.A.</em> group ought never endorse</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>A.A.</em> Milne.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>I.D.</em></td>
<td>The receptionist took his <em>I.D.</em></td>
<td>• ID as an identification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kennon's <em>I.D.</em> and a small yellow plastic</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>A.A.</em></td>
<td>Portland, Maine; the Rev. <em>A.A.</em> ... thanks to Mr. <em>A.A.</em> Robinson ..</td>
<td>• Person</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mr. <em>A.A.</em> Boyden, Mr. Bruce...</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>In 1860, Dr. <em>A.A.</em> Liebeault ...</td>
<td></td>
</tr>
</tbody>
</table>

The more relevant issue is the determination of a situation where one might want to associate the abbreviated character string to its corresponding non-abbreviated character string. Obviously, numerous instances of the non-abbreviated character strings *identification* and *Alcoholics Anonymous* could be found in the datasets. As we have shown, in a corpus, there could be many instances where non-abbreviated character strings and abbreviated character strings that share the same meaning. Figure 18 is a Venn diagram representation of the character strings: *A.A.*, *AA*, and *I.D.* and *ID*. The left side box shows typically what happens when LO punctuation normalization is applied. As discussed earlier, the normalized character string and
un-normalized character string would match depending on the contextual usage. On the right side, the senses of the character strings are the same for both \textit{I.D.} and \textit{ID}. Even though such diagrammatically represented conditions exist, unfortunately it is difficult to generalize the effects of lexical-level punctuation pattern for other character strings other than \textit{A.A.} and \textit{I.D.}.

To sum up, similar to the case of the string \textit{B.C.}, simply applying punctuation normalization operations to the string such as \textit{B.C.} could produce unintended semantic effects. To probe into the semantic effects of LO punctuation marks, we would have to consider different senses of the individual LO lexical term itself. However, this representation is only based on specific instances involving the Gutenberg dataset.

\textbf{Figure 18.} Conceptual punctuation normalization effects: \textit{A.A.} and \textit{I.D.}

One last consideration is the blank space issue. This adds a slight complexity to the Venn diagram representation, but, in principle, examining the punctuation normalization effects should work the same way as before. As mentioned previously, blank spaces are an integral part of punctuation normalization procedures. We also need to consider the blank space variations for the above character strings. By applying add-infix blank space and expanding, we could form
the character strings as $A.A.$ and $I.D.$.

Using the Gutenberg dataset, we retrieved the text lines containing the above string variations: $A.A.$, $I.D.$, $A.A.$ and $I.D.$. Table 40 shows various retrieved text lines from the Gutenberg dataset. As shown, in both datasets, there were no instances of text lines that contained the string $I.D.$. However, when $I.D.$ was used, different results were found. In the Gutenberg dataset, the character string matched the text line that contained $BY\ DAV\ ID\ SAM\ WEL\ L$. If the aggressive searching method could be used, it might be reasonable to match $I.D.$ with $BY\ DAV\ ID\ SAM\ WEL\ L$. Although this is a non-punctuated search query case, this particular search implies that various search type scenarios need to be considered, including such a case involving abnormal blank spaces.

If the context of the abbreviation could be determined by a system, this type of string matching could be avoided. In the Usenet dataset, however, most uses of the string $I.D.$ was associated with entity or personal names. However, using the term as “identification” was still evident (e.g., Medical Alert I.D Bracelet). We also searched for the string $A.A.$, but there were only one instance where the text line contained the string. The returned result was the following: $were\ marked\ A\ A\ A\ A$. This line contained the period at the end. In this case, four A’s $A\ A\ A\ A$ should have been recognized as a single character string by a tokenizer that attempts to find proper tokens.
Table 40

*Lines Containing Abbreviated Words with Infix Blank Space*

<table>
<thead>
<tr>
<th>Punctuated Strings</th>
<th>Gutenberg</th>
<th>Usenet</th>
</tr>
</thead>
<tbody>
<tr>
<td>A A</td>
<td>were marked A A A.</td>
<td>Share Business ExchangeTwitterFacebook</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Size A A A</td>
</tr>
<tr>
<td>7.</td>
<td>H A A G U N</td>
<td>A A A A ATYPE(one_type) :: ot(:)</td>
</tr>
<tr>
<td>A A A A Aot = one_func(a, b, c, update(tt)) ! -- A????</td>
<td>A TYPE(two_type) :: tt</td>
<td></td>
</tr>
<tr>
<td>A A A A A</td>
<td>A A A A -</td>
<td></td>
</tr>
<tr>
<td>A A A A TYPE(two_type) :: tt</td>
<td>A A END FUNCTION one_func</td>
<td></td>
</tr>
<tr>
<td>A A END TYPE one_type</td>
<td>A A END TYPE two_type</td>
<td></td>
</tr>
<tr>
<td>A A FUNCTION one_func(a, b, c, callback)</td>
<td>A A TYPE one_type</td>
<td></td>
</tr>
<tr>
<td>A A TYPE two_type</td>
<td>A A USE one</td>
<td></td>
</tr>
<tr>
<td>A A</td>
<td>D G A A B B A A</td>
<td></td>
</tr>
<tr>
<td>7. 0 0 H A A G U N</td>
<td>British A A Driving Directions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Career As A A Graphic Designer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Find A A Person</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Writing A A Essay</td>
<td></td>
</tr>
</tbody>
</table>

A. A. *(see figure 16)*

<table>
<thead>
<tr>
<th>I D</th>
<th>(see figure 16)</th>
<th>them from A. A. Attanasio's &quot;In Other Worlds&quot; part of the Radix that cause more than half our ills. Dr. A. A. Brill, an American</th>
</tr>
</thead>
<tbody>
<tr>
<td>D I D Motorcycle Wheels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gold Diamond I D Bracelet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hill's I D Pet Food</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hills I D Canine Pet Food</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hills I D Pet Food</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Juvenile Diabete I D Bracelets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K I D Productions Mini Truck</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical Alert I D Bracelet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical Alert I D S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Photo I D Software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Science Diet I D Canine Recall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Science Diet I D Recall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Science Diet I D Reviews</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I. D. *(none)*

<table>
<thead>
<tr>
<th>I. D.</th>
<th>(none)</th>
<th>(none)</th>
</tr>
</thead>
</table>

There were numerous instances of A. A. Figure 19 shows 24 text lines that contained the string A. A. Most of them were used in the context of personal name references. Similarly, there were 2 similar occurrences of text that contained the A. A. in the Usenet dataset that were
used in the context of personal names. As suggested in the previous chapter, as the length of the alphanumeric portion of string increases the punctuation marks in the string may have less semantic effects on the string.

Another important implication of this example is that all 4 variations of the character strings – \( A.A \), \( AA \), \( A \ A \), and \( A \ A \) – need to be examined collectively because of blank spaces in some of these character strings. In these types of cases, it is difficult to simply come up with blank space normalization rules that do not alter character string senses. Yet, in the case of \( A.A \) and \( A \ . \ A \) , unlike \( AA \) and \( A \ A \) , the character strings appear to have equivalent senses. Thus, the effects of blank space normalization need to be examined together with these types of cases.

Index

<table>
<thead>
<tr>
<th>Index</th>
<th>Text Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>annotated by E. H. and E. W. Blashfield and A. A. Hopkins. Passing</td>
</tr>
<tr>
<td>2.</td>
<td>Ill., The Berlin Machine Works, Beloit, Wis., A. A. Loetscher,</td>
</tr>
<tr>
<td>3.</td>
<td>Mrs. Padraic Colum, Mr. A. A. Boyden, Mr. Ellery Sedgwick, Mr. Henry A.</td>
</tr>
<tr>
<td>4.</td>
<td>REV. C. A. A. TAYLOR, OCALA, FLA. 80 R. A. A. L.. A. A. Bungart, J. C. McDaniel</td>
</tr>
<tr>
<td>5.</td>
<td>In the spring of 1862, the Rev. A. A. Miner, D.D., was elected to</td>
</tr>
<tr>
<td>6.</td>
<td>TWX COL. A. A. NEUMAN TO WHITE HOUSE:</td>
</tr>
<tr>
<td>7.</td>
<td>TWX COL. A. A. NEUMAN TO WHITE HOUSE:</td>
</tr>
<tr>
<td>8.</td>
<td>Diseases of the fetlock, ankle, and foot. By A. A. Holcombe 395</td>
</tr>
<tr>
<td>9.</td>
<td>adjutant was Lieutenant A. A. Ross, who rose from the ranks and some</td>
</tr>
<tr>
<td>10.</td>
<td>Carrol D. Bush, A. A. Bungart, J. C. McDaniel</td>
</tr>
<tr>
<td>11.</td>
<td>MEMBERSHIP--Mrs. S. H. Graham, A. A. Bungart, Mrs. Herbert Negus,</td>
</tr>
<tr>
<td>12.</td>
<td>untiring hospitality. To Miss A. A. Pirie, who was with us for the</td>
</tr>
<tr>
<td>13.</td>
<td>A. M. mi. N Nejapa, Oaxaca, taken by A. A. Alcorn, on August 6, 1955,</td>
</tr>
<tr>
<td>14.</td>
<td>collected on June 19, 1954, by A. A. Alcorn, are from 1/2 mi. E Piaxtla</td>
</tr>
<tr>
<td>15.</td>
<td>CAMPING OUT._ By A. A. MACDONELL. [Double vol., 2_s._]</td>
</tr>
<tr>
<td>16.</td>
<td>They are a goody number, and include Mr. A. A. Arnold, Mr. Stephen T.</td>
</tr>
<tr>
<td>17.</td>
<td>(1950); J. R. and A. A. Alcorn (1954 and 1955); R. H. Baker and a party</td>
</tr>
<tr>
<td>18.</td>
<td>by J. R. Alcorn and A. A. Alcorn in 1955 and 1956. A few</td>
</tr>
<tr>
<td>19.</td>
<td>Miss A. A. Cowan, 30, Gauden Road, Clapham, London, S.W.</td>
</tr>
<tr>
<td>20.</td>
<td>of subsistence for the men of his detail-- The A. A. Q. M. will A. MILNE</td>
</tr>
<tr>
<td>21.</td>
<td>Miss A. A. Cowan, 30, Gauden Road, Clapham, London, S.W.</td>
</tr>
<tr>
<td>22.</td>
<td>November--having as his Chief of Staff and A. A. General, Captain</td>
</tr>
<tr>
<td>23.</td>
<td>William O. Kimball of Boston, Mrs. A. A. Lord of Newton, Mrs. Charles M.</td>
</tr>
<tr>
<td>24.</td>
<td>amongst them Dr. A. A. Riddel--my &quot;Archie&quot;--and my dearest friend Dr.</td>
</tr>
</tbody>
</table>

Figure 19. Text lines containing \( A.A \) from the Gutenberg dataset.
Punctuation Normalization Framework

Until now, the two types of punctuation normalization discussed were LI punctuation normalization and LO punctuation normalizations. In addition, there were distinctive operations that were associated with these two punctuation normalization types. These were blank space normalization and various types of punctuation normalization operations (e.g., remove prefix ◊). While each punctuation normalization problem was isolated from the other text normalization issues, the procedures can often be co-dependent upon each other. In the overall framework of punctuation normalization, one type of procedure has a direct effect on the other procedure. Among components of text retrieval environment, these types of punctuation normalization procedures shown in Figure 20.

In this figure, punctuation normalization is a common required procedure for both sides of the text retrieval environment. That is, there is the content side and the search side. LO punctuation normalization and LI punctuation normalization are directly associated with additional notions such as disambiguation, blank space normalization, and text normalization operations. These two types of normalization can be treated independently, or they can be integrated into a particular design.

As the figure suggests, sometimes disambiguating punctuation marks and punctuated character strings is essential for effective punctuation normalization. As indicated earlier, disambiguating the role of punctuation marks often arises. For example, a person’s name Jones, B. has a comma and a period. These punctuation marks cannot be simply removed when a user’s intention is to search for the name exactly as it appears above. It could be useful for a user to find a bibliographic reference that contains the above string pattern. LI punctuation marks do not apply semantic effects at a lexical level.
Figure 20. Punctuation normalization framework.

By the process of elimination, LI punctuation marks can aid in determining the role of LO punctuation marks; LO punctuation marks can aid in determining the role of LI punctuation marks. For example, by identifying the comma after the period, we know that the period is more likely used in the case of abbreviation than in ending a sentence. Text features such as capitalization, the length of the character string in LO, the order of punctuation marks that appear in character string and co-occurrences of punctuation marks within a text segment are all important attributes that can aid in disambiguating the role of punctuation marks. There have been attempts to disambiguate the role of punctuation marks, such as disambiguation of a period as suggested by Grefenstette and Tapanainen (1994) and also by Kiss and Strunk (2006). Such
efforts are needed to disambiguate more types of punctuation marks in a corpus, particularly for the purpose of distinguishing LI punctuation marks from LO punctuation marks.

Earlier we pointed out that blank space had an effect on matching character strings or phrases. Particularly, one phenomena associated with blank spaces was the fact that removing infix-punctuation marks produces blank spaces. Also, there were blank space behavior characteristics that involved abbreviated character strings that were demonstrated. We compared the effects of having a blank space in the middle of an abbreviated character string and not having one in the middle character string. The variability of the blank space character had to be taken into account in all aspects of processing punctuation marks. Identifying tokenization points can become a problematic issue since some units such as $A A A$ should have been treated as one unit when searching for $A A$. Also, we had to keep in mind that blank spaces were not always consistently used (e.g., $A.A.$ instead of $A . A$). The demonstration procedures performed showed that blank spaces and search query types were strongly linked to the problem of punctuation normalization.

We should stress the point that frequently matching a character string to another character string does not involve any punctuation marks at all (e.g., *healthcare* to *health care*). Issues related to punctuation marks could be viewed as somewhat an independent issue from this standpoint. Still, blank space normalization is special type of issues that could be examined along with hyphenated character strings. Also, since a compound word could be hyphenated, they may still relate to punctuation normalization issue.

One of the primary goals of this study was to investigate the meaningfulness of punctuation marks and punctuation patterns. Comprehensively, several concrete factors had to
be considered in assessing the punctuation marks, and a simplistic decision could not be reached for this reason.

Chapter Summary

This chapter has shown qualitatively analyzed results of punctuation normalization strategy. An analytical observational approach was undertaken to identify various types of punctuation normalization. These are LI normalization, LO normalization, blank space normalization, and normalization operations. LO punctuation normalization focused on normalizing LO punctuation marks whereas LI punctuation normalization focused on normalizing LI punctuation marks.

In addition to these two types of punctuation normalization is blank space normalization. Blank space normalization and tokenization play a critical role in punctuation normalization as well as disambiguating the role of punctuation marks and punctuated character strings. Blank space normalization was categorized into its operational type: add, delete, or replace.

Punctuation normalization decisions may depend upon the type of search queries. In particular, the effects of punctuation normalization were investigated based on different types of exact search queries and approximate search queries. This study suggests that search queries can be largely divided into the following: exact search query, approximate search query, punctuated search query, or non-punctuated query. Approximate search queries could be conservative or aggressive in this study, but the criteria had to be defined clearly by a designer.

Semantic relationships between character strings were visually represented using a Venn diagram. We also discussed how the disambiguation of punctuation marks can be a vital aspect of punctuation normalization. Normalization demonstrations were conducted to gain additional
insights into the punctuation normalization effects. The demonstrations indicated that characterizing the character string meaning was important for different forms of abbreviated character string. Finally, the discussion on the implications of the overall analysis of punctuation normalization was presented. We discussed how all of these elements can be taken into account in assessing the meaningfulness of punctuation marks and punctuation patterns.
CHAPTER 6
CONCLUSIONS

This research attempted to find implications of normalizing punctuation marks from a perspective of text retrieval. We conducted this study to see the feasibility of a punctuation-centric approach in analyzing issues related to normalizing punctuation marks. For this purpose, we relied on a punctuation frequency analysis to initially explore punctuation marks and understand issues surrounding punctuation normalization. Subsequently, the normalization strategy analysis was performed. We attempted to analyze the effects of punctuation normalization by assuming that a particular set of decisions are required for a text retrieval system. At the same time, we investigated various factors that need to be taken into consideration for punctuation normalization; we have demonstrated this by examining varying types of punctuation marks. The punctuation normalization framework presented in Chapter 5 provides an overview of the complex relationships of factors that need to be considered in normalizing punctuation marks.

Let us examine how we went about finding the detailed issues relevant to punctuation normalization and how we investigated the effects of punctuation normalization. Using two datasets, we first employed a frequency count strategy to characterize punctuation marks. This study mainly used punctuation frequency analysis procedures to analyze punctuation marks, punctuation patterns, and punctuated character strings in the datasets. By exploring punctuation marks, we came to a conclusion that dividing punctuation marks into two groups, namely Lexical Independent (LI) punctuation marks and Lexical Oriented (LO) punctuation marks, was useful. This punctuation mark categorization was important to understand and distinguish the types of punctuation marks from a text retrieval perspective. In terms of normalizing these broad types of
punctuation marks, an LI punctuation normalization approach was suitable for LI punctuation marks and an LO punctuation normalization approach was suitable for LO punctuation marks.

In the analysis of punctuation normalization procedures, blank spaces and search query types were examined in conjunction with punctuation normalization. In particular, by examining the semantic differences between abbreviated character strings and non-abbreviated character strings, we identified potential semantic effects that were associated with LO punctuation marks. Toward the end of this study, we discussed different types of semantic effects that punctuation normalization can have on LO punctuation marks.

In many respects, the role of an individual punctuation mark is a critical aspect of punctuation normalization. We have shown that depending on the role of punctuation marks, the effects of punctuation mark differed on the character string. We further discussed how disambiguation of the individual role of punctuation marks and associated punctuated character string was necessary in determining the type of punctuation normalization technique to apply. Some punctuation marks have had multiple roles in our language. This often causes problems by raising the complexity of the character string since a system needs to computationally determine the punctuation marks roles. Determination of punctuation mark roles meant that a proper category of punctuation marks (e.g., LI punctuation marks or LO punctuation marks) was necessary. Once the type of punctuation marks was determined, an appropriate approach could be undertaken to normalize punctuation marks.

Many types of character strings can be formed due to punctuation normalization. For example, hyphenated character strings could be formed as z-z, z–z. Along with this issue, blank spaces and tokenization played critical roles in punctuation normalization. As demonstrated in the study, the problems had to be explored in conjunction with different types of
assumptions related to search features (e.g., different types of approximate searching).

Somewhat unexpected results of the study were how complex the effects of punctuation normalization can be due to the sense, or meaning, of a character string. The semantic effects of normalizing LO punctuation marks were that the normalization procedure could change the senses that are associated with the original character strings.

For example, we showed how $BC$ and $B.C.$ were used in different contexts. It is clear that differences among punctuated character strings and non-punctuated character strings can potentially produce unexpected search results. Thus, determining the contextual usage is required in many types of situations. Because of undesirable semantic effects, substituting one form of a punctuated character string with another form of the character string had to be carefully examined. In particular, we have explored the semantic problems through the use of Venn diagrams focusing on the abbreviated and non-abbreviated character strings.

Attempts to solve this type of problem could be computationally challenging. A challenge to improve punctuation normalization is to figure out the different senses or meanings of punctuated character strings, in the context of improving a text retrieval system. In many instances, simplistic answers to punctuation normalization were not available. For LO punctuation marks, achieving optimal punctuation normalization is more difficult to achieve because of three issues:

- A singular approach to normalize punctuation is often inadequate because of varying types of punctuated character strings.
- Some LO punctuation marks are highly sensitive to normalization (e.g., the character string $C++$).
- The semantic effects of LO punctuation marks are difficult to generalize since in each case the specific senses that were involved could be different as a result of LO punctuation normalization.
Limitations of Study

There were a number of limitations of this study that needs to be mentioned. Some of the limitations were due to the fact that we had to constrain the analysis setting to control the study, while other limitations were associated with the findings of this study. As we pointed out, the complexity of punctuation normalization is enormous and it requires further investigation. More specifically, the following are noteworthy limitations of the study:

First, although punctuation marks were extensively analyzed, a more thorough investigation of punctuation patterns could be made. Some examples have been extensively used in this study, but we have not investigated LO punctuation patterns comprehensively. In the previous chapter, we pointed out that un-normalized and normalized punctuated character strings could be equivalent to each other depending on their contextual usage. Additional analysis that focused more on the effects of LO punctuation normalization could have discovered more sub-categories of “equivalent depending on the contextual usage” since the effects in this category were rather simplified.

Second, this research indicated that there are cases of domain- and genre-specific punctuation marks. However, we have only relied on these results based on the examination of two datasets. We have not sufficiently examined domain-specific punctuation patterns and character strings that involve domain-specific punctuation marks. Thus, identification of domain punctuation marks and with different types of domains can provide certain domain specific knowledge. Such domain-specific types of punctuation patterns were not fully explored.

Third, the framework of punctuation normalization in the previous chapter pointed out the need for disambiguation of the roles of punctuation marks. Yet we were unable to provide sufficient details on specific methods to disambiguate all types of punctuation mark roles. While
previous research has suggested the feasibility of disambiguating the role of punctuation marks, computational approach to determine whether the role of punctuation marks is LO or LI might be challenging.

Fourth, there could be limitations due to various assumptions that we have laid out in conducting this study. For instance, the search terms we considered only applied to short queries that involved just 2-3 words. The indexing we relied upon was word-based. As pointed out in this study, word-based indexing is not required for an internal data structure that is necessary to support string matching. Because our research was based on such restrictive assumptions, there could be other punctuation normalization approaches than we have considered. Although the overall framework still might be applicable and useful in terms of understanding the issues, under this scenario there could be other effects of punctuation normalization that could be different and might require elaborate tuning.

Lastly, our research did not provide concrete, procedural methods to improve the overall effectiveness of text retrieval systems. In general, the text contained a vast number of punctuation marks that needs to be normalized in one way or the other. In many instances, we have shown specific cases, where proper punctuation normalization can lead to producing optimal search results for a user. Yet, we have not shown concrete procedural steps in normalizing punctuation patterns. Moreover, admittedly, there are certain text elements that are inherent to our language that do not necessarily involve punctuation marks. For these reason, the effect this research has on the overall performance of text retrieval systems has not yet been proven. Instead, by focusing on punctuation marks, we attempted to shed a light on the text retrieval and text normalization related issues as a whole.
Broader Implications of the Study

In a broader sense, we provided some noteworthy findings that could point toward designing more effective text retrieval systems. This research has shown that a punctuation centric view was feasible to tackle the punctuation normalization problem. After conducting this study, we realized that the capability of punctuation normalization related decisions must be made at an extremely detailed level.

Yet, we believe that for effective normalization, determining the individual roles of punctuation marks in a corpus is essential. While punctuation marks are not particularly ambiguous to a human, they are often ambiguous from a computational perspective. Computationally, because one punctuation mark can be used for multiple types of situations, sometimes not even consistently, it requires substantial effort to resolve these issues in a corpus. This would help in the overall punctuation normalization process.

Nevertheless, systematically analyzing punctuation marks seemed necessary for all types of domains, even if we are not certain about the outcome of the text retrieval implications in general. Rather than recommending solutions at a detailed level, this study explored the issues so that practitioners of system design can utilize this research as a means to gain insights into the relevant factors and effects of punctuation normalization. In the form of sub-research questions and responses, more implications of punctuation normalization are presented.

Also in a broader sense, the bottleneck to resolving punctuation normalization problems is due to the complexity of our language and how it is used. In many ways, the issues related to punctuation normalization touched upon the core issue of language versus the ability to retrieve relevant texts efficiently. We have shown that punctuation marks often played a critical roles in reducing lexical ambiguity by clarifying the notion of character strings in various ways. At the
same time, because of variants of punctuation marks that require punctuation normalization most often for the purpose of retrieving the relevant text, the goal of retrieving only relevant punctuation marks are challenging. Moreover, a user’s perspective has significant impact on how that particular user attempts to find new information and that this should be taken into account in building a representation (O’Connor et al., 2009). In a sense, this study was an initial attempt to build such a representation that could satisfy user’s information need.

To produce effective search and retrieval results, we believe that an approach to punctuation normalization problem needs to be systematic at a micro-level. We indicated in our results that many punctuation patterns are likely to exist in a corpus. A simple approach to punctuation normalization without understanding the effects is not desirable since it would not improve the text retrieval process. Thus, many detailed decisions in handling punctuation normalization would require many relevant factors. In analyzing punctuation normalization problems, a systematic method needs to be developed that include the uses of frequency counts. We have shown that frequency counts are a viable tool in uncovering domain- and genre-specific characteristics of punctuated character strings and punctuation marks. Also, there needs to be a systematic method of evaluating scenarios with various types of search queries for the purpose of tackling the issues related to semantic effects of punctuation marks and its normalization.

Sub-Research Questions and Responses

This research had the overarching research question: *What are the implications of normalizing punctuation marks from the perspective of text retrieval?* In addition, the study had these research questions:
1. In what ways and to what extent are punctuation marks and punctuated character strings (i.e., character strings that contain punctuation marks) important in terms of text retrieval?

2. How does punctuation normalization affect the process of matching punctuated character strings?

3. What are some of the useful categories of punctuation marks, punctuation patterns, and punctuation normalization?

4. How can we measure similarity between terms in an orthographic sense so that it can be incorporated into normalizing punctuation marks?

5. In what ways can we use underlying punctuation patterns in a text for a text retrieval system?

Answering these sub-research questions provides a way of summarizing our research results efficiently. Before continuing, readers are reminded that the main research question focused on finding implications of the normalization of punctuation marks. The following summarizes findings from the research by answering each of these questions.

**First, in what ways and to what extent are punctuation marks and punctuated character strings important in terms of text retrieval?** From a text retrieval perspective, the degree of importance was dependent upon the roles and utilization of punctuation marks in text retrieval systems. The degree of punctuation mark importance varied depending upon the specific types of search and retrieval circumstances.

The degree of importance could be examined from the search side first. In some instances, punctuation marks provide users with an ability to search for precise information. It meant that users could be searching for segments of text that contain any type of punctuated character strings. For example, a search query might contain terms such as the following: X.22, C++, (MIP)-1alpha, and cat -v. The importance of punctuation marks could be relevant in these types of situations. The reason is due to the discriminating attribute that the punctuation marks have on the character string.
In general, there are situations where punctuation normalization is straightforward. Yet, the effects of LI punctuation normalization were that it could often retrieve two adjacent character strings that are separated by LI punctuation marks as a single continuous phrase. We have shown that LI punctuation marks that are present in this type of character strings were less meaningful in general except in these types of situations. However, LI punctuation marks as a whole were still essential because of the tokenization points. Also, disambiguation of LI punctuation marks meant that LO punctuation marks that are positioned close by could be disambiguated collectively.

Also, the longer the combinational punctuation marks, the less meaningful they became in the context of text retrieval. It is highly improbable that the user of text retrieval systems can specify a search query having an exact pattern of long combinational punctuation marks. We have shown infrequent uses of long combinational punctuation marks in Chapter 4 (See Figure 9 and Table 21). None of these punctuation marks appeared to be particularly common in the context of text retrieval. Even disregarding the semantics of a punctuated character string, to be useful as a search query, the pattern must be natural and easy enough for the user to know exactly how to specify a search query based on the assumption that such a search query will be effectively processed by a text retrieval system.

In assessing the importance of punctuation marks, the potential utilization and the adverse effects of punctuation marks can be viewed as important criteria. Based on the roles of punctuation marks, we have seen that punctuation marks can be used to disambiguate the character strings in a text. For instance, we pointed out that the periods used after capitalized letter might be an indication of an abbreviated character string.
When the search results of B.C. and BC were compared, at least for the case that we examined, there was a greater sense of abbreviation with the character string B.C. than with the character string BC. The character string BC could have been used in other context (e.g., math equations) rather than as an abbreviated character string. The specific role of punctuation marks in this case clarifies that this is an abbreviated character string. This attribute, in turn, can be viewed as an indication of importance in the context of clarifying the particular form of the abbreviated character string. We have also demonstrated some of the adverse effects of punctuation marks using the Venn diagram. From the perspective of improving text retrieval systems, procedures that could create adverse effects of punctuation normalization need to be avoided.

Second, how does punctuation normalization affect the process of matching punctuated character strings? Answering this question required an examination of the goals of text retrieval systems. We can assume that the responsibility of a text retrieval system was to deliver the results of search requests in an effective manner without overloading a user with irrelevant information. In this sense, the system needs to carefully sort out the types of punctuation marks and punctuated character strings for the purpose of normalizing punctuation marks. Simply eliminating punctuation marks could bring unwanted irrelevant information to the user. In some situations we examined, simply normalizing punctuation marks could produce adverse effects by finding too many irrelevant matches.

Punctuated character strings comprise two types of punctuation marks: Lexical Independent (LI) punctuation marks and Lexical Oriented (LO) punctuation marks. Punctuation normalization techniques differed depending on these two punctuation mark types. Also, depending on whether the punctuated character string contained LI punctuation marks or LO
punctuation marks, the effects of these types of punctuation normalization differed. LO
punctuation marks are punctuation marks at the lexical-level. As a result, the variant form of
punctuated character strings and semantically similar character strings had to be considered in
the matching process.

On the other hand, LI punctuation marks were not associated with the variants of
punctuated character strings. In general, the system could normalize LI punctuation marks by
simply removing them. However, we pointed out that the LI punctuation mark locality should be
tracked, perhaps by tagging the locations. In this manner, when exact phrase searches are made,
these two neighboring character strings would not be part of the search results. Otherwise, two
adjacent character strings that are separated by a LI punctuation mark could form a single phrase
when the LI punctuation mark is removed. Moreover, the punctuation normalization issues also
had to be examined according to additional types of searches, namely exact versus approximate
searches.

Third, what are some of the useful categories of punctuation marks, punctuation patterns,
punctuation normalization, and the effects of punctuation normalization? Various categorization
schemes were provided in this study. Based on the frequency count approach, this study
identified various types of punctuation marks and related concepts, different types of punctuation
normalization, and the effects of punctuation normalization.

Table 41 is a summary of the categorization of punctuation marks related concepts that
were discussed in this study.
Table 41

*The Categorization of Punctuation Mark Related Concepts*

<table>
<thead>
<tr>
<th>Types</th>
<th>Types/Categorization</th>
<th>Key Criteria or Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punctuation Marks and Punctuation Pattern</td>
<td>• Jones punctuation mark (source independent versus source-specific punctuation marks)</td>
<td>• linguistic role</td>
</tr>
<tr>
<td></td>
<td>• paired-punctuation marks</td>
<td>• punctuation marks occurring as a pair</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• delimiting punctuation mark roles</td>
</tr>
<tr>
<td></td>
<td>• affixes (prefix, suffix, and infix, punctuation mark exclusive)</td>
<td>• locality of punctuation marks</td>
</tr>
<tr>
<td></td>
<td>• LI punctuation marks</td>
<td>• referencing area</td>
</tr>
<tr>
<td></td>
<td>• LO punctuation marks</td>
<td>• role of punctuation marks (sentence level versus lexical level)</td>
</tr>
<tr>
<td></td>
<td>• sentential punctuation pattern</td>
<td>• sentence based punctuation</td>
</tr>
<tr>
<td></td>
<td>• combinational punctuation mark</td>
<td>• multiple combinations of punctuation marks</td>
</tr>
<tr>
<td></td>
<td>• lexical-level punctuation pattern</td>
<td>• pattern of LO punctuated character string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• length of punctuated character string</td>
</tr>
<tr>
<td>Punctuation Normalization</td>
<td>• LI punctuation normalization</td>
<td>• LI &amp; LO punctuation marks</td>
</tr>
<tr>
<td></td>
<td>• LO punctuation normalization</td>
<td>• internal blank spaces</td>
</tr>
<tr>
<td>Effects of Punctuation Normalization</td>
<td>• blank space normalization</td>
<td>• exact search/approximate search</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• semantic effects</td>
</tr>
<tr>
<td></td>
<td>• LI punctuation normalization effects</td>
<td>• blank spaces</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• exact search/approximate search</td>
</tr>
<tr>
<td></td>
<td>• LO punctuation normalization effects</td>
<td>• semantic effects</td>
</tr>
</tbody>
</table>

*Fourth, how can we measure the similarity between terms in an orthographic sense?*

In this study, we noticed that an orthographic measurement could provide a quantifiable value in determining similarity between two terms. In considering a particular formula, we suggested measuring orthographic distances based on spelling and punctuation mark differences. In
calculating the Punctuated String Distance (PSD), we assumed that a set of alphanumeric characters were identical. Thus, the PSD metric was used to measure the orthographic distance of character string having different sets of punctuation marks.

By observing the characteristic of string strength and the senses of lexical items, we pointed out that the PSD could be an additional attribute to be utilized during the process of punctuation normalization and building indexes. LO punctuated character strings can be often associated with different types of character string variants. The orthographic distance would be different except for the character strings that have exact spelling, including the punctuation marks in this study’s context.

This study indicated that PSD could be measured by adopting an algorithm such as Levenshtein distance and Stoilos distance measurements. In particular, we showed how these measurements could indicate the sense of “closeness” to each other if they meet a certain threshold value. However, as pointed out in Chapter 4, this measurement alone could be limited in showing the effects of LO punctuation normalization.

Lastly, in what ways can we use underlying punctuation patterns that exist in a corpus? This question was partly answered by the previous question responses. However, this question was about the punctuation patterns rather than the type of punctuation marks. This study categorized punctuation patterns as lexical-level punctuation pattern and structural-level punctuation pattern. Broadly speaking, we have categorized punctuation marks as LI and LO. Nevertheless, since there was no previous study completed that pointed to the extent of punctuation marks in search queries, this study assumed that specifying a punctuated search query that could utilize any type of underlying punctuation pattern is not expected from an average user.
On the other hand, from a system’s point of view, useful ways to utilize the underlying punctuation pattern to disambiguate the role of punctuation marks and the type of lexical word (e.g., abbreviation) was more critical. The disambiguation of character strings could be accomplished by inspecting different textual features: the type of punctuation mark and punctuation pattern, capitalization, neighboring character strings, etc. For example, a period after a single letter is often an abbreviated character string.

To this end, heuristic based rules for LO punctuation normalization procedures could be crafted specific to a corpus. Developing such rules was not undertaken. However, we have experienced with our datasets that such tasks require extensive examination of punctuation patterns in character strings. By far, the most challenging task seem to be punctuation normalization that maintains the original meaning of character string while taking into account all the string variations that the user might use in the search query.

Contributions

Numerous contributions of this study can be mentioned. Most of the contributions were based on the fact that punctuation normalization issues were explored from a text retrieval perspective. We summarize our contributions according to the following key areas:

- **Characterization of punctuation marks and punctuation patterns**: By exploring punctuation marks from a text retrieval perspective, we increased general understandings of punctuation mark related issues. In particular, various roles of punctuation marks and punctuation patterns were explored. We demonstrated how punctuation marks and punctuation patterns can be utilized in terms of developing punctuation normalization procedures.
• **Categorization of punctuation marks and punctuation patterns**: Several types of punctuation marks and punctuation patterns were developed in this study. They were primarily used as a means to investigate this study's larger research question. Overall, we demonstrated how this categorization could aid in understanding the effects of punctuation normalization. This categorization should be useful for future studies that are related to text retrieval.

• **Methodological approach**: This study demonstrated that frequency count is a useful strategy in identifying underlying punctuation mark characteristics. In our case, frequency counts were an effective tool to characterize domain- and genre specific punctuation marks and punctuated character strings. Numerous important punctuation mark characteristics were obtained through conducting various frequency counts. In this study, many terms, or concepts, have been defined through the study. In turn, these concepts can be used as a basis for investigating any type of punctuation mark related studies.

• **Punctuation normalization framework**: In the previous chapter, we provided a punctuation normalization framework. We categorized punctuation normalization problems into two types, namely LI punctuation normalization and LO punctuation normalization. We also suggested how disambiguation problems can be dependent on each type of text normalization procedures and often require conceptualization in terms of its punctuation normalization categories. We also suggested the possibility of exploring orthographic distance, particularly the PSD metric for the purpose of incorporating it into punctuation normalization procedures. Collectively, the framework suggests how punctuation normalization problem should be approached.
Recommendations for Future Studies

As in many research studies, some additional research topics were identified as a result of conducting this study. Overall, this research has laid a foundation for a systematic analysis of punctuation normalization problem. Many detailed issues including the need for algorithms to disambiguate punctuation mark roles have not been resolved. Research in some of these areas might further support this study’s findings. Topics for future suggested research areas include the following:

- Additional corpus-based investigation on punctuation patterns
- Domain-specific and genre-specific punctuation mark studies
- Disambiguation of the role of punctuation mark and punctuated character string
- Investigation of the applicability of the psd measurement
- Punctuation normalization issues other than the english language
- Comparison of punctuated character strings and non-punctuated character strings based on character string senses

First, more corpus-based investigations of punctuation patterns are desirable. Because insufficient amounts of statistics were available in the literature, it was difficult to differentiate domain-specific punctuation patterns from more general punctuation patterns. More frequency analyses will provide a better understanding of the characteristics related to punctuation patterns. Since there was little previous research that could be used as a guideline in exploring the frequency strategy to investigate punctuation pattern, identifying many types of punctuated character strings were limited due to time constraints. Future studies can advance this area by utilizing the punctuation mark categories developed in this study. As a result, investigating more punctuation patterns should point to devising algorithms that are needed to disambiguate punctuation marks and punctuated character strings.
Second, more research is needed in examining domain- and genre-specific punctuation marks. In this study, we concluded that punctuation mark usage is different depending on the particular domain or corpus. For example, research could be done using a corpus that might represent academic written materials. By analyzing punctuation patterns in such a domain, more generalizable domain-specific punctuation patterns could be discovered for text retrieval. In conjunction to punctuation marks from natural languages, the underlying punctuation patterns from content areas such as bibliographic references and mathematical equations could be explored.

Third, future research can examine the issue of disambiguation problems by further examining punctuation marks and punctuated character strings. Although the LI and LO punctuation marks were established in this study, in terms of the effects of LO punctuation normalization, sufficient investigations have not been carried out. Thus, any future studies can simply extend this research by examining LI and LO punctuation normalization. In particular, this study suggested that generally LO punctuation marks are dependent on punctuated character strings at a lexical level or inter-lexical level. We pointed out that disambiguating the roles of LI and LO punctuation marks can help identify the type of character strings such as abbreviated character strings and non-standard words. Such issues could be examined in terms of developing algorithms.

Fourth, studies focusing on the applications of the PSD measurement are clearly desirable. We investigated the notion of the PSD measurement in our study. Yet, as mentioned in the previous chapter, we have not fully investigated the applicability of the PSD measurement for text retrieval purposes. By investigating character strings in a corpus, we could find the specific patterns that are associated with PSD measurements. Also, we considered only
matching sets of alphabetical characters in our study. By mixing the problem with non-matching alphabetical characters, we could find more intriguing patterns. For instance, character strings that are mis-spelled but have same type of pattern of punctuation marks could be examined using the same approach.

Fifth, in this study punctuation normalization was focused on the English language only. While punctuation marks are highly language dependent (Santos, 1998), there are some notable similarities among Indo-European languages as pointed out by Parkes (1993). If we take the study further, punctuation marks could be examined from an evolutionary point of view (Holden, 2008). It would be interesting to setup a corpora to compare punctuation marks in different languages as (Tavecchio, 2010) attempted. From this study’s point of view, we would be more interested in punctuation normalization issues in an attempt to develop an accurately responsive text retrieval system.

Lastly, research efforts need to focus more on the comparison of punctuated character strings and non-punctuated character strings based on string senses or meaning. This study indicated that punctuation marks add different sense to a character string, although un-normalized and normalized character strings can be semantically equivalent depending on the contextual usage. We demonstrated how different punctuated character strings are associated or change the meaning of a character string. The PSD measurement was important in this type of investigation. The Venn diagram approach to model different semantic relationships between normalized character strings and un-normalized character strings could also be extended. In sum, issues related to the senses of the individual types of punctuated character strings could provide additional understanding to the semantic effects of LO punctuation normalization.
APPENDIX A

ASCII CODES
The punctuation marks for this study are shown in the following ASCII table. The only exception to this table is the *space* (decimal value #32), which can be considered as a special class of punctuation mark. For this study, the blank space corresponds to space (decimal number #32). This item is highlighted in Table A.1. See (Final Text of DIS 8859, 1998) for a complete listing of ASCII and 8-bit ASCII characters. A number of ASCII character variants exist. ISO-8859-1 was intended for Western European languages, and it is also commonly known as ASCII Latin-1. The decimal value #0 to #126 belong to ASCII code 7 bit. The decimal value #128 and higher belong to only 8 bit ASCII code. The term “extended” ASCII table refers to the 8 bit ASCII table.

The ASCII codes from decimal #128 to #159 are reserved for control purposes (Korpela, 2012b). The character set that is referred to as Windows character set, more officially known as Code Page 1252 Windows Latin-1, utilizes some of these character numbers. As shown in this table, ASCII characters #128 to #159 from the Code Page Windows 1252 Latin-1 were included as punctuation marks for this table. See (Code Page 1252 Windows, 2013) for a complete listing of the Code Page Windows 1252 Latin-1.

Table A.1

<table>
<thead>
<tr>
<th>DEC</th>
<th>OCT</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>040</td>
<td></td>
<td>Space</td>
</tr>
<tr>
<td>33</td>
<td>041</td>
<td>!</td>
<td>Exclamation Mark</td>
</tr>
<tr>
<td>34</td>
<td>042</td>
<td>&quot;</td>
<td>Quotation Mark</td>
</tr>
<tr>
<td>35</td>
<td>043</td>
<td>#</td>
<td>Number Sign</td>
</tr>
<tr>
<td>36</td>
<td>044</td>
<td>$</td>
<td>Dollar Sign</td>
</tr>
<tr>
<td>37</td>
<td>045</td>
<td>%</td>
<td>Percentage Sign</td>
</tr>
<tr>
<td>38</td>
<td>046</td>
<td>&amp;</td>
<td>Ampersand</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>---</td>
<td>-----------</td>
</tr>
<tr>
<td>39</td>
<td>047</td>
<td>'</td>
<td>Apostrophe</td>
</tr>
<tr>
<td>40</td>
<td>050</td>
<td>(</td>
<td>Left Parenthesis</td>
</tr>
<tr>
<td>41</td>
<td>051</td>
<td>)</td>
<td>Right Parenthesis</td>
</tr>
<tr>
<td>42</td>
<td>052</td>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>43</td>
<td>053</td>
<td>+</td>
<td>Plus Sign</td>
</tr>
<tr>
<td>44</td>
<td>054</td>
<td>,</td>
<td>Comma</td>
</tr>
<tr>
<td>45</td>
<td>055</td>
<td>-</td>
<td>Hyphen-Minus</td>
</tr>
<tr>
<td>46</td>
<td>056</td>
<td>.</td>
<td>Full Stop</td>
</tr>
<tr>
<td>47</td>
<td>057</td>
<td>/</td>
<td>Solidus</td>
</tr>
<tr>
<td>58</td>
<td>072</td>
<td>:</td>
<td>Colon</td>
</tr>
<tr>
<td>59</td>
<td>073</td>
<td>;</td>
<td>Semicolon</td>
</tr>
<tr>
<td>60</td>
<td>074</td>
<td>&lt;</td>
<td>Less-Than Sign</td>
</tr>
<tr>
<td>61</td>
<td>075</td>
<td>=</td>
<td>Equals Sign</td>
</tr>
<tr>
<td>62</td>
<td>076</td>
<td>&gt;</td>
<td>Greater Than Sign</td>
</tr>
<tr>
<td>63</td>
<td>077</td>
<td>?</td>
<td>Question Mark</td>
</tr>
<tr>
<td>64</td>
<td>100</td>
<td>@</td>
<td>Commercial At</td>
</tr>
<tr>
<td>91</td>
<td>133</td>
<td>[</td>
<td>Left Square Bracket</td>
</tr>
<tr>
<td>92</td>
<td>134</td>
<td>\</td>
<td>Reverse Solidus</td>
</tr>
<tr>
<td>93</td>
<td>135</td>
<td>]</td>
<td>Right Square Bracket</td>
</tr>
<tr>
<td>94</td>
<td>136</td>
<td>^</td>
<td>Circumflex Accent</td>
</tr>
<tr>
<td>95</td>
<td>137</td>
<td>_</td>
<td>Low Line</td>
</tr>
<tr>
<td>96</td>
<td>140</td>
<td>`</td>
<td>Grave Accent</td>
</tr>
<tr>
<td>123</td>
<td>173</td>
<td>{</td>
<td>Left Curly Bracket</td>
</tr>
<tr>
<td>124</td>
<td>174</td>
<td></td>
<td>Vertical Line</td>
</tr>
</tbody>
</table>
Right Curly Bracket
Tilde
Euro Sign
Single Low-9 Quotation Mark
Latin Small Letter F With Hook
Double Low-9 Quotation Mark
Horizontal Ellipsis
Dagger
Double Dagger
Modifier Letter Circumflex Accent
Per Mille Sign
Latin Capital Letter S With Caron
Single Left-Pointing Angle Quotation
Latin Capital Ligature Oe
Latin Capital Letter Z With Caron
Left Single Quotation Mark
Right Single Quotation Mark
Left Double Quotation Mark
Right Double Quotation Mark
Bullet
En Dash
Em Dash
Small Tilde
Trade Mark Sign
Latin Small Letter S With Caron
<table>
<thead>
<tr>
<th>Unicode</th>
<th>Decimal</th>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>155</td>
<td>233</td>
<td>&gt;</td>
<td>Single Right-Pointing Angle Quotation Mark</td>
</tr>
<tr>
<td>156</td>
<td>234</td>
<td>œ</td>
<td>Latin Small Ligature Oe</td>
</tr>
<tr>
<td>158</td>
<td>236</td>
<td>ź</td>
<td>Latin Small Letter Z With Caron</td>
</tr>
<tr>
<td>159</td>
<td>237</td>
<td>Ŷ</td>
<td>Latin Capital Letter Y With Diacritic</td>
</tr>
<tr>
<td>161</td>
<td>241</td>
<td>i</td>
<td>Inverted Exclamation Mark</td>
</tr>
<tr>
<td>162</td>
<td>242</td>
<td>ĺ</td>
<td>Cent Sign</td>
</tr>
<tr>
<td>163</td>
<td>243</td>
<td>£</td>
<td>Pound Sign</td>
</tr>
<tr>
<td>164</td>
<td>244</td>
<td>☟</td>
<td>Currency Sign</td>
</tr>
<tr>
<td>165</td>
<td>245</td>
<td>¥</td>
<td>Yen Sign</td>
</tr>
<tr>
<td>166</td>
<td>246</td>
<td>‖</td>
<td>Broken Bar</td>
</tr>
<tr>
<td>167</td>
<td>247</td>
<td>§</td>
<td>Section Sign</td>
</tr>
<tr>
<td>168</td>
<td>250</td>
<td>¨</td>
<td>Dieresis</td>
</tr>
<tr>
<td>169</td>
<td>251</td>
<td>©</td>
<td>Copyright Sign</td>
</tr>
<tr>
<td>170</td>
<td>252</td>
<td>a</td>
<td>Feminine Ordinal Indicator</td>
</tr>
<tr>
<td>171</td>
<td>253</td>
<td>«</td>
<td>Left-Pointing Double Angle Quotation Mark</td>
</tr>
<tr>
<td>172</td>
<td>254</td>
<td>¬</td>
<td>Not Sign</td>
</tr>
<tr>
<td>173</td>
<td>255</td>
<td></td>
<td>Soft Hyphen</td>
</tr>
<tr>
<td>174</td>
<td>256</td>
<td>®</td>
<td>Registered Sign</td>
</tr>
<tr>
<td>175</td>
<td>257</td>
<td>¯</td>
<td>Macron</td>
</tr>
<tr>
<td>176</td>
<td>260</td>
<td>°</td>
<td>Degree Sign</td>
</tr>
<tr>
<td>177</td>
<td>261</td>
<td>±</td>
<td>Plus-Minus Sign</td>
</tr>
<tr>
<td>180</td>
<td>264</td>
<td>′</td>
<td>Acute Accent</td>
</tr>
<tr>
<td>181</td>
<td>265</td>
<td>µ</td>
<td>Micro Sign</td>
</tr>
<tr>
<td>182</td>
<td>266</td>
<td>¶</td>
<td>Pilcrow Sign</td>
</tr>
<tr>
<td>183</td>
<td>267</td>
<td>·</td>
<td>Middle Dot</td>
</tr>
<tr>
<td>Code</td>
<td>Numeric Value</td>
<td>Character</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>270</td>
<td>, Cedilla</td>
<td></td>
</tr>
<tr>
<td>185</td>
<td>271</td>
<td>' Superscript One</td>
<td></td>
</tr>
<tr>
<td>186</td>
<td>272</td>
<td>° Masculine Ordinal Indicator</td>
<td></td>
</tr>
<tr>
<td>187</td>
<td>273</td>
<td>» Right Pointing Double Angle Quotation Mark</td>
<td></td>
</tr>
<tr>
<td>191</td>
<td>277</td>
<td>¿ Inverted Question Mark</td>
<td></td>
</tr>
<tr>
<td>215</td>
<td>327</td>
<td>× Multiplication Sign</td>
<td></td>
</tr>
<tr>
<td>247</td>
<td>367</td>
<td>÷ Division Sign</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX B

USENET FILES FOR YEAR 2009
APPENDIX C

SUMMARY OF GREP COMMANDS
In this Appendix, the *grep* commands that are used in this study are described in further
detail. Grep searches the named input filenames for lines containing a match to the given
pattern. By default, grep prints the matching lines. The *grep* command can be used in the
following manner:

\[ \text{grep [options] PATTERN filename} \]

Several options are available with the *grep* command. The -c option was mainly used to perform
the frequency count. The PATTERN needs to contain details about that strings that need to be
matched.

In Chapter 3, the following *grep* commands were used:

- The command that was used to calculate the occurrences of punctuated token was

\[ \text{grep -c "^[^0-9a-zA-Z]*$" filename} \]

The above *grep* command was used to match only one of more non-alphanumeric characters
that end with non-alphanumeric characters. The filename is simply a name of the file that
contains a token list based on a blank space. The caret symbol \(^\) matches the expression at
the start of a line, while the dollar sign \(\) matches the expression at the end of a line. The
asterisk \(\ast\) matches zero or more occurrence of the previous character. Furthermore, a
bracket expression is a list of characters enclosed by the bracket \([]\). The caret \(^\) matches any
character \(\text{not}\) in the list.

- The command that was used to calculate the occurrences of terms that end with (‘s) was

\[ \text{grep -c "'s$"} \]

In the above *grep* command, the pattern inside quotation mark \(\"\) describes details about the
character strings that needs to be matched. The apostrophe-s with dollar sign \(\$\) specifies
that the lines ends with apostrophe-s need to be matched. The -c option was used to count the lines that containing the pattern.

- The command that was used to calculate the occurrences of hyphen-infix punctuated tokens with two prefix characters was

```bash
grep '^[a-zA-Z0-9][a-zA-Z0-9][a-zA-Z0-9]-[a-zA-Z0-9]' filename
```

In the above `grep` command, the pattern `^[a-zA-Z0-9][a-zA-Z0-9]-[a-zA-Z0-9]` states that character strings that begins with two alphanumeric characters need to be matched. Then the hyphens with any alphanumeric characters are being matched.

- The command that was used to calculate the occurrences of all tokens that contain a hyphen was

```bash
grep -c $'\055' filename
```

In the above `grep` command, the pattern inside the apostrophes specifies the octal value 055. Using this command, any token with the ASCII code and with octal value 055 was matched and the occurrences were counted.

- The commands that were used to calculate the occurrences of all tokens that contain a paired-punctuation marks were the following:

```bash
"\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]" grep "\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]" filename

*\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]* grep "*\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]*" filename

([a-zA-Z0-9].*[a-zA-Z0-9]) grep "([a-zA-Z0-9].*[a-zA-Z0-9])" filename

'\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]' grep "'[a-zA-Z0-9].*[a-zA-Z0-9]'" filename

\[\[a-zA-Z0-9].*[a-zA-Z0-9]\] grep "\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]" filename

\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]\_ grep "\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]\_" filename

/\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]/ grep "/\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]/" filename

.\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]. grep ".\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]." filename
"\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]" grep "\[\[a-zA-Z0-9].*[a-zA-Z0-9]\]" filename
```
The period matches any character and the asterisk * matches the preceding character. Collectively, they are used to match any characters between the two alphanumeric characters as shown in each command. As in case of previous `grep` commands, the pattern `[a-zA-Z0-9]` means an alphanumeric character. The back slash \ is an escape character. In doing so, the system knows that the preceding character is not part of a `grep` command character.

- The command that was used to calculate the occurrences of all tokens with single-letter-infix-slash (e.g., w/o).

  ```
  grep " ^-[a-zA-Z0-9].*-[a-zA-Z0-9]$ " filename
  ```

  The above `grep` command specifies to match one alphabetical character, followed by a slash, and followed by another alphabetical character. The carat indicates the beginning of the character and the dollar sign $ indicates the end of the line. Using this command, only tokens that contain single-letter-infix-slash are matched and the occurrences were counted.
APPENDIX D

INFIX-PUNCTUATION PATTERN (GUTENBERG TO USENET)
<table>
<thead>
<tr>
<th>Infix-punctuation Pattern</th>
<th>Gutenberg</th>
<th>Usenet</th>
<th>Relative % Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>z-z</td>
<td>145903</td>
<td>112948</td>
<td>22.59%</td>
</tr>
<tr>
<td>z'z.</td>
<td>3245</td>
<td>4558</td>
<td>28.81%</td>
</tr>
<tr>
<td>z..z.</td>
<td>2077</td>
<td>1454</td>
<td>30.00%</td>
</tr>
<tr>
<td>z-z-z-z</td>
<td>5288</td>
<td>7827</td>
<td>32.44%</td>
</tr>
<tr>
<td>z'z</td>
<td>273112</td>
<td>445472</td>
<td>38.69%</td>
</tr>
<tr>
<td>z.z.z.</td>
<td>2885</td>
<td>1658</td>
<td>42.53%</td>
</tr>
<tr>
<td>z.z</td>
<td>6837</td>
<td>13032</td>
<td>47.54%</td>
</tr>
<tr>
<td>z-z.</td>
<td>17735</td>
<td>7949</td>
<td>55.18%</td>
</tr>
<tr>
<td>z'z.</td>
<td>6601</td>
<td>2950</td>
<td>55.31%</td>
</tr>
<tr>
<td>&quot;z'z z</td>
<td>24282</td>
<td>8828</td>
<td>63.64%</td>
</tr>
<tr>
<td>z.z.z.</td>
<td>1036</td>
<td>3283</td>
<td>68.44%</td>
</tr>
<tr>
<td>z.z</td>
<td>2728</td>
<td>10325</td>
<td>73.58%</td>
</tr>
<tr>
<td>z-z.</td>
<td>36160</td>
<td>8862</td>
<td>75.49%</td>
</tr>
<tr>
<td>z..z.</td>
<td>465</td>
<td>2233</td>
<td>79.18%</td>
</tr>
<tr>
<td>(z'z</td>
<td>340</td>
<td>2212</td>
<td>84.63%</td>
</tr>
<tr>
<td>z.z</td>
<td>4017</td>
<td>27614</td>
<td>85.45%</td>
</tr>
<tr>
<td>z.z.</td>
<td>302</td>
<td>2608</td>
<td>88.42%</td>
</tr>
<tr>
<td>$z.z.</td>
<td>270</td>
<td>2746</td>
<td>90.17%</td>
</tr>
<tr>
<td>$z.z. z</td>
<td>197</td>
<td>2040</td>
<td>90.34%</td>
</tr>
<tr>
<td>z(z)</td>
<td>170</td>
<td>2828</td>
<td>93.99%</td>
</tr>
<tr>
<td>z/z.</td>
<td>88</td>
<td>1495</td>
<td>94.11%</td>
</tr>
<tr>
<td>z/z.</td>
<td>73</td>
<td>1628</td>
<td>95.52%</td>
</tr>
<tr>
<td>z/z</td>
<td>821</td>
<td>21281</td>
<td>96.14%</td>
</tr>
<tr>
<td>z..z.</td>
<td>298</td>
<td>10273</td>
<td>97.10%</td>
</tr>
<tr>
<td>z-z-z</td>
<td>132745</td>
<td>3576</td>
<td>97.31%</td>
</tr>
<tr>
<td>z...z</td>
<td>47</td>
<td>2710</td>
<td>98.27%</td>
</tr>
<tr>
<td>z&amp;z</td>
<td>16</td>
<td>1306</td>
<td>98.77%</td>
</tr>
<tr>
<td>z/z/z</td>
<td>33</td>
<td>3547</td>
<td>99.07%</td>
</tr>
<tr>
<td>z...z.z</td>
<td>12</td>
<td>2340</td>
<td>99.49%</td>
</tr>
<tr>
<td>z...z</td>
<td>35</td>
<td>9058</td>
<td>99.61%</td>
</tr>
<tr>
<td>z(z)</td>
<td>4</td>
<td>1577</td>
<td>99.75%</td>
</tr>
<tr>
<td>z..z</td>
<td>2</td>
<td>1301</td>
<td>99.85%</td>
</tr>
</tbody>
</table>
APPENDIX E

TOP RANKED FREQUENCY USED INFIX-APOSTROPHE (GUTUENBERG)
APPENDIX F

TOP RANKED INFIX-APOSTROPHE S ('S) (GUTENBERG)
Crow's
brother's,
Aristotle's
agent's
"isn't
ye've
wolf's
Smollett's
sea's
Quixote's
Prophet's
Poe's
Ne'er
Mason's
Lamb's
Hugo's
Horace's
Harold's
Guy's
Fletcher's
CHRIST'S
chap's
Chancellor's
can't,"
CAN'T
Browne's
bride's
Beethoven's
Angel's
Andy's
Amy's
afternoon's
don't_
"she's
Wright's
Willie's
Taine's
steamer's
spider's
APPENDIX G

LESS FREQUENTLY USED INFIX-APOSTROPHE (GUTENBERG)
APPENDIX H

TOP RANKED INFIX-HYPHEN (GUTENBERG)
APPENDIX I

SUFFIX PUNCTUATION MARKS (GUTUENBERG)
257  
248  
245  )
234  ??
233  ??
231  __
224  .).
223  :"__
APPENDIX J

FIRST TWO LETTERS WITH INFIX-HYPHEN (GUTENBERG)
to-night

to-morrow.

br-read,
dr-rink!"
to-day,
re-exports;
re-imports.
so-called
to-day.

ex-Premier's
to-morrow
up-to-date
Li-a-bil-ity
so-styled
re-ception
to-day?
to-day,
to-day,
to-day.
to-day.
to-night,"
to-night

up-river

to-morrow.
to-night?

To-night,
to-night,
to-night

up-to-date

Li-a-bil-ity
so-styled
re-ception

to-day?
to-day,
to-day,
to-day,
to-day.
to-night,"
to-night

in-sensitiveness

re-edified,
to-day,"
to-day;
to-day.
re-echoing
to-day.
So-and-so,
Me-doras
re-appeared.
to-day,
in-dulgences,
to-day." he-doll,) un-frequent
re-echoing
to-morrow," up-to-date
to-morrow
to-night."
to-night

to-night.
to-morrow, 
to-night;
to-day?"
go-cart 
to-day, 
re-created
Bo-peep.]
Bo-peep
Bo-peep
Bo-peep
mo-ther
al-ways
mo-ther
fa-ther's
mo-ther
be-fore
in-stead;
al-ways
ar-bour
ar-bour
ar-bour, 
ar-bour 
be-fore.
ex-student
40-pagxan
to-day, 
to-day. 
ha-has,
APPENDIX K

FREQ. COUNTS OF SINGLE-LETTER-INFIX-SLASH (USENET)
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>168</td>
<td>I/O</td>
</tr>
<tr>
<td>91</td>
<td>w/o</td>
</tr>
<tr>
<td>57</td>
<td>b/c</td>
</tr>
<tr>
<td>41</td>
<td>N/A</td>
</tr>
<tr>
<td>37</td>
<td>A/C</td>
</tr>
<tr>
<td>31</td>
<td>i/o</td>
</tr>
<tr>
<td>25</td>
<td>a/c</td>
</tr>
<tr>
<td>21</td>
<td>O/S</td>
</tr>
<tr>
<td>16</td>
<td>s/w</td>
</tr>
<tr>
<td>13</td>
<td>A/B</td>
</tr>
<tr>
<td>13</td>
<td>A/V</td>
</tr>
<tr>
<td>13</td>
<td>y/o</td>
</tr>
<tr>
<td>12</td>
<td>S/N</td>
</tr>
<tr>
<td>11</td>
<td>c/o</td>
</tr>
<tr>
<td>10</td>
<td>m/s</td>
</tr>
<tr>
<td>10</td>
<td>o/p</td>
</tr>
<tr>
<td>10</td>
<td>v/s</td>
</tr>
<tr>
<td>9</td>
<td>d/l</td>
</tr>
<tr>
<td>8</td>
<td>M/C</td>
</tr>
<tr>
<td>8</td>
<td>m/n</td>
</tr>
<tr>
<td>7</td>
<td>a/b</td>
</tr>
<tr>
<td>7</td>
<td>b/w</td>
</tr>
<tr>
<td>7</td>
<td>G/K</td>
</tr>
<tr>
<td>7</td>
<td>L/D</td>
</tr>
<tr>
<td>7</td>
<td>s/n</td>
</tr>
<tr>
<td>7</td>
<td>S/S</td>
</tr>
<tr>
<td>6</td>
<td>B/W</td>
</tr>
<tr>
<td>6</td>
<td>e/w</td>
</tr>
<tr>
<td>5</td>
<td>C/I</td>
</tr>
<tr>
<td>5</td>
<td>D/A</td>
</tr>
<tr>
<td>5</td>
<td>g/f</td>
</tr>
<tr>
<td>5</td>
<td>n/a</td>
</tr>
<tr>
<td>5</td>
<td>o/s</td>
</tr>
<tr>
<td>5</td>
<td>S/R</td>
</tr>
<tr>
<td>5</td>
<td>v/o</td>
</tr>
<tr>
<td>4</td>
<td>b/d</td>
</tr>
<tr>
<td>4</td>
<td>C/S</td>
</tr>
<tr>
<td>4</td>
<td>G/B</td>
</tr>
<tr>
<td>4</td>
<td>G/H</td>
</tr>
<tr>
<td>4</td>
<td>H/K</td>
</tr>
<tr>
<td>4</td>
<td>N/R</td>
</tr>
<tr>
<td>4</td>
<td>N/S</td>
</tr>
</tbody>
</table>
REFERENCES


