



U.S. Federal Institute of Museum and Library Services
National Leadership Grant

**Realizing the Vision of Networked Access to
Library Resources**

*An Applied Research and Demonstration Project to
Establish and Operate a Z39.50 Interoperability Testbed*

Data Normalization Procedures on Decomposed MARC 21 Records

Ed Kim

<ed2kim@hotmail.com >

Z-Interop Research Assistant

&

William E. Moen, Ph.D.

<wemoen@unt.edu>

Principal Investigator

School of Library and Information Sciences

Texas Center for Digital Knowledge

University of North Texas

Denton, TX 76203

October 25, 2001

Revised January 1, 2002

Table of Contents

- 1. Introduction**
- 2. Overview**
- 3. Leading and Trailing Punctuations**
- 4. Elimination of Non-Word Strings**
- 5. Disregarding Capitalization**
- 6. Fragmented Words Treatment**
- 7. Additional Normalization**
- 8. Script to Normalize Data**

Data Normalization Procedures on Decomposed MARC 21 Record

1. Introduction

In this document we present some aspects of data normalization of the decomposed records to improve the results of analysis. The data normalization processes use pattern-matching techniques to eliminate and/or generalize anomalous characters and terms. Since the unit of analysis in preparing the test dataset of 400,000 MARC 21 records is a "word," there was a need for data normalization to provide reliability in the subsequent analysis.

2. Overview

First, we analyzed the general patterns of terms by inspecting frequency counts of words in the decomposed records (see Z-Interop document, **Decomposing MARC 21 Records for Analysis** for information about the decomposed records). After discovering some patterns, we developed certain logical normalization rules. These normalization rules have been constructed into our shell script. The filtering tool 'grep' and string substitution tool 'sed' were mainly used in the script. These tools are integral part of Unix commands. Since some of the additional options were required, we used GNU textutils that can be found from <http://www.gnu.org>. These tools allow pattern matching techniques known as regular expressions. Normalization of terms include several operations:

- Eliminate leading and trailing punctuations
- Eliminate non-word strings
- Disregard capitalization
- Treat fragmented words

3. Leading and Trailing Punctuations

The words field consists of sequence of non-space characters bounded by spaces. Since each term is decomposed based on spaces, the most obvious problems that exist in the "words" in the initial decomposed records are that they contained numerous leading and trailing punctuations. Our initial approach was to eliminate some of the anomalous characters after inspecting 9992 records. Leading and trailing punctuations such as period, parenthesis, ampersand, quote were removed from the data collection initially, realizing that removing some of the punctuations such as \$ can lead to semantic problems for information retrieval. It can be debated that such punctuation marks are desirable in some instances to preserve the original semantics of a word. Since many vendor systems use arbitrary method of choosing which punctuation marks to index, we decided to generalize the normalization even further so that the words will accept only numeric digits [0-9], strings of alphabets [a-z] or [A-Z].

4. Elimination of Non-Word Strings

We decide to eliminate some strings of characters that do not appear to be a conventional word from the 9992 record set.

The following anomalies were found in the 9992 record set, and they have been removed from the listing:

```
LINES THAT CONTAIN DOUBLE SLASH //  
LINES THAT CONTAIN 20010215.  
LINES THAT STARTS WITH 690.* AND ENDS WITH eng
```

LINES THAT STARTS WITH 6 AND 8 CHARACTERS LONG AND STARTS WITH 6
LINES THAT STARTS WITH ocm0

5. Disregarding Capitalization

We decided to disregard capitalization rule when we generated frequency counts of words. Instead, all characters were transformed into UPPER CASE letters for subsequent processing.

6. Fragmented Words Treatment

In addition, we found some fragmented patterns of words. Two fragments of words were joint by the patter "period hyphen hyphen" i.e., "- -". Since these fragments will cause problems for retrieval of data in MySQL, we applied additional rules to treat these anomalous records. We decide to further split these terms and include them in the alphabetically sorted frequency report. We have decided to include these exceptional words in a dictionary file so that these words are not selected as a candidate term.

7. Additional Normalization

Once 9992 records are normalized, we inspected frequency counts of alphabetically sorted words of 33+ million records. Additional normalization rules had to be developed. These rules are still being developed.

For example, if a word consists of numeric numbers, accept only patterns of words with four numeric digits or less. Otherwise, delete the lines from the record set

8. Script to Normalize Data

The following Korn shell script normalizes the data and saves the results to the file called 'sample_norm'. It also generates the frequency report file 'sample.freq.report'. Use the following command at the Unix prompt where normalizeit.4 is the current version of the script and top9992 is the name of the input file that contains the 9992 sample records

```
Shell> ./normalizeit.4 top9992
```

```
#!/bin/ksh
#
#
# SCRIPT TO NORMALIZE THE SAMPLE DATA
# DELETE LINES THAT CONTAIN DOUBLE SLASH AND SAVE IT TO $1.1.txt
grep -v '//.' $1 > $1.1.txt

# DELETE LINES THAT STARTS WITH 20010215. AND SAVE IT TO $1.2.txt
grep -v '20010215.*\' $1.1.txt > $1.2.txt

# DELETE LINES THAT STARTS WITH 690.* AND ENDS WITH eng
# AND ALSO DELETE LINES THAT STARTS WITH 6 AND 8 CHARACTERS LONG AND
SAVE
# THE RESULTS TO $1.3.txt
```

```
grep -v '690.*eng' $1.2.txt | grep -v '\<6[0-9][0-9][0-9]....\>' >
$1.3.txt

# DELETE LINES THAT STARTS WITH ocm0 AND SAVE THE RESULTS TO $1.4.txt
grep -v 'ocm0' $1.3.txt > $1.4.txt

# DELETE LINES THAT CONTAIN MORE THAN 4 NUMERIC CHARACTERS
grep -v '[0-9][0-9][0-9][0-9][0-9]*[0-9]' $1.4.txt > $1.5.txt

# SELECT ONLY THE WORD COLUMN AND SAVE IT TO sword.txt FOR LATER USE
cut -f9 $1.5.txt > sword.txt
#
integer x=0;
while [ x -le 1 ];
do
#
# ELIMINATE LEADING AND TRAILING PUNCTUATIONS
sed 's/[^0-9a-zA-Z]*'// sword.txt > sword1.txt
sed 's/[^0-9a-zA-Z]$'// sword1.txt > sword3.txt

# PATCH BACK THE NORMALIZED DATA. USE sample_norm IF USED BY MYSQL
# IN MYSQL PERFORM OPERATIONS THAT ARE EQUIVALENT TO BELOW SCRIPT
if [ x -eq 0 ]; then
    cut -f1-8 $1.4.txt > nosword.txt
    paste -d'\t' nosword.txt sword3.txt > sample_norm;
fi;
#
# ADDITIONAL SCRIPT FOR REPORTING
#
# DELETE LINES WITH BLANK LINES
grep -v '^$' sword3.txt > sword4.txt

# SPLITT .-- TO TWO TOKENS
grep -v '.--' sword4.txt > sword5.txt
grep '.--' sword3.txt | sed 's/.--.*'// > sword3.1
grep '.--' sword3.txt | sed 's/.*.--'// > sword3.2

# APPEND THE SPLIT WORDS AGAIN
cat sword3.1 >> sword3.2
cat sword3.2 >> sword5.txt

# PREPARE FOR THE LOOP AND RUN IT AGAIN
cp sword5.txt sword.txt
x=x+1
done

# SORT AND GENERATE FREQUENCY REPORT
sort sword.txt | /usr/local/bin/uniq -ic > sample.freq.report
```