Vanderbilt Football Sports Writing Data Collection and Feature Extraction

Authors
Thaddeus Hatcher
Cole McKiever
Ben Petersen
Don Tran

Abstract

In this assignment, we were tasked with collecting 20 articles relating to Vanderbilt University's 2018 football season games on a week-by-week basis. The contents of the articles were placed into individual text files to serve as author samples. We wrote a feature extraction algorithm using Python 2.7 that goes through each text file and generates character unigram feature vectors. In this paper, we will provide an in-depth discussion about the article selection process, the development of the feature extraction algorithm, and challenges we faced along the way.

Introduction

To start off our project, we began by splitting up the project into different sections. One section would be collecting newspaper articles, another would be writing the code, and finally we would write our documentation for what we found. We put 2 of our group members, Cole and Thaddeus, onto the task of finding newspaper articles. Meanwhile, Don authored the source code, and Ben focused on the documentation. Once everyone knew what they were supposed to do, we set off on our tasks.

Newspapers And Authors

The ideal scope for article selection was to select those written by newspapers within the state of Tennessee. This was the standpoint from which we began searching for articles. We attempted to find as many articles as we could that were related to a Vanderbilt football game of a particular week and were published by a newspaper within the state of Tennessee. It was a challenge finding 20 articles matching this description, so we eventually had to widen our scope to newspapers that were not based in Tennessee. An additional ideal constraint placed on the selection process was that the 20 articles be from 20 different authors. This proved to be quite a difficult constraint to work under, as well. After widening our scope to newspapers outside the state of Tennessee we were able to attain 20 articles for week one. However, these 20 articles were written by less than 20 authors, meaning that we had repeating authors within week one. In order to meet the constraint of having each article within a given week be from a different author, we omitted articles from authors in our week one samples whom had not written articles for the second and third weeks. Then, we chose only one article per author to keep from the remaining group of articles. At the end of this process, we had completed our final dataset consisting of nine articles written by nine different authors for each of the three weeks. Across all three weeks, we have articles from these same nine authors.

Feature Extraction

Our feature extraction source code performs file I/O and contains logic for character unigram and feature vector analysis on a given subsequence of text. This analysis is to determine the character frequency on ACII values 0x20 (inclusive) through 0x7F (exclusive). File I/O is performed on text files within a specified subdirectory. To do this, we defined the directories that we are using for inputs from the CASIS-25 dataset and our Vanderbilt dataset. As well as defined the directories our outputs would be written to. Our first function processes each incoming file into a feature vector and then a normalized feature vector.

The main functionality of our source code comes from the vectorProcessFile function which creates the feature vector and normalized feature vector from the text file name passed to it. We implemented a dictionary via key/value pairs which would map relevant ASCII values to a character frequency counter. This dictionary's keys were each of the ASCII characters from decimal 32 (inclusive) through 127 (exclusive) and its values are the count of each ASCII character found in the file. If the character in the file does not correspond to a key in the dictionary, it was simply ignored. Otherwise, the key that matches the character in the text file had its value incremented by 1. We continue this process until the entire file has been traversed. Once, we have reached the end of the file, the completed feature vector was appended to a list containing feature vectors for the specified subdirectory. In order to create the normalized vector, we used the the same process which we used to compute the raw feature vector and divided each vector dimension by the magnitude of the vector. Similarly, the normalized vector was appended to a separate list containing normalized character unigram vectors for a specified dataset.

In order to output our vectors to text files, we use a func-

tion called outputFileDatasetResults. In this function, we first clear the stored feature vector and normalized vector lists from any previously computed vector data to prevent data corruption and overlap. We then get all the files in the directory specified in the arguments and open each one to process it with vectorProcessFile. Once all of the files in the given directory have been processed, we write the feature vectors list and the normalized vectors list to their own files.

We run this procedure for both the CASIS-25 dataset and the Vanderbuilt SEC sport's writers dataset that we collected–each from the initial three weeks of the continuing regular football season.

CASIS-25 Dataset

In order to ensure baseline performance and accuracy of our feature extraction algorithm, we performed an algorithmic output comparison between our newly computed texfile output and the expected output from the CASIS-25 provided dataset. We were provided the dataset and text output files for their respective character unigram features which we used for comparison to validate our algorithmic output.

Upon closer evaluation, we observed that the dataset we were provided had minor character encoding issues (presumably from WINDOWS-1252 formatted text, converted to UTF-8) which revealed some characters meant for punctuation appeared as extended ASCII values. This produced minor deviations from our expected and actual baseline outputs, but similar enough results such that we were able to effectively ignore any errors. We wrote a function called printCasisOutputSimilarity which computed the Euclidean and Manhattan distances between both combined feature vector dataset outputs and found that both outputs were within 2 and 4 frequency points of accuracy, respectively.

Our SEC SportsWriters Dataset

Here is a list of the authors we used for our samples and the newspapers for which they wrote:

- Adam Sparks - Tennessean
- Betsy Goodfriend - Vanderbilt Hustler
- Christian D'Andrea - Anchor of Gold
- David Boclair - Nashville Post
- Pete Fiutak - College Football News
- Shawner Allen - Anchor Of Gold
- Stetson Marlin - Tennessee Truth
- Teresa Walker - Associated Press News
- Tom Stephenson - Anchor Of Gold

Dataset Statistics

CASIS-25 Dataset Character Average: 1640.18
CASIS-25 Dataset Word Average: 274.5
CASIS-25 Dataset Rough Sentence Average: 13.08

VANDERBILT Dataset Character Average: 3571.148
VANDERBILT Dataset Word Average: 611.741
VANDERBILT Dataset Rough Sentence Average: 39.296

Breakdown of the Work

Finding sports articles -
- Thaddeus Hatcher
- Cole McKiever

Writing source code -
- Don Tran

Reviewing source code -
- Ben Petersen

Writing of documentation -
- Thaddeus Hatcher
- Ben Petersen

References

- File I/O Python - `https://www.tutorialspoint.com/python_files_io_htm`
- Reading all files in a specified directory - `https://askubuntu.com/352202`
- About built-in Python function chr(i) - `https://docs.python.org/3/library/functions.html#chr`
- Vector normalization documentation - `https://www.docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.norm.html`
- Implementation of numpy linalg.norm() - `https://stackoverflow.com/a/1401828/9718306`
- Delete file contents before writing or appending (Zero-indexing) - `https://stackoverflow.com/a/41915174/9718306`
- Iteratively reading an entire sub-directory as sorted *** - `https://stackoverflow.com/a/12093995/9718306`
- Helpful list-list iteration - `https://stackoverflow.com/a/6340411/9718306`
- How to count number of words in a file (Helpful for collecting stats) - `https://www.sanfoundry.com/python-program-count-number-words-characters-fi`
- How to compute Euclidean and Manhattan Distance between two vectors which "should" be equal xRef: function printCasisOutputSimilarity() for usage `http://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implement`