Automatic Bookmark Classification Final Report

Tianyin Wei, Yuan Ma
Instructor: Prof. Yi Lu. Murphey

ECE 579 Intelligent System
Department of Electrical and Computer Engineering
University of Michigan-Dearborn

I. Introduction

Bookmarks, being a widely used strategy to relocate sites of interest, allow the user to create a personalized URL repository. Most current browsers support hierarchical bookmarking schemes that enable users to manage their collection of URLs in a hierarchy of folders. However, with growing size of the repository, difficulties in organizing and maintaining the hierarchical structure arise, as the classification of new bookmarks, for example finding and creating an appropriate folder to store them.

The aim of our project is to establish a system that can automatically categorize bookmarks and arrange them in the proper hierarchical folders with only one click.

This paper aims at a novel approach to automate the bookmark classification process. The passage first gives a brief idea of bookmark classification and then presents the detailed process of data training, testing and classification on a technological level. The central contribution of this passage is to present a content-based algorithm that categorizes a random webpage into an appropriate folder.

II. Background Knowledge

With the problems mentioned in the previous section, we consider the classification of the data contained in the bookmark. An individual book consists of many parts that contains useful information for classification.

A. Bookmark Data Structure

There are many kinds of Data Structure used for describe bookmark information. In Chrome, a bookmark contains the following information:

- **URL** - the location of the document
- **ADD_DATE** - an integer that represents the number of seconds elapsed since midnight January 1, 1970
- **LAST_VISIT** - an integer that represents when the document was last visited.
- **LAST_MODIFIED** - an integer that represents when the document was modified last.
- **title** - the title of this document

Following is an example shows how bookmark information is stored in *eXtensible Bookmarks Exchange Language (XBEL)* file.

```
<XBEL>
  <title> Title of the page </title>
  <Folder>
    <title> Folder title </title>
    <bookmark  href = "address of the web page"
    >
    <title> Bookmark Title </title>
    <desc>Bookmark Description</desc>
    </bookmark>
  </folder>
</XBEL>
```

Listing 1: represent of one bookmark in XBEL format

The mostly used available information of a bookmark consists of the following three parts: *URL*, *title* and *description*. Therefore, the basic idea is to classify bookmarks using only the information in URL and title. However this may cause the following problems. The text of title is minimal but not particularly informative, for example: *http://www.CNN.com*, where the letters CNN does not provide a literal meaning related to news report.

What is more, there might ambiguous information. Like the word *'Java'*, it can refer to a programming language, an island and a fern at the same time. So *content-based method* should be used for getting more useful information from bookmark which will make the result of classification more reasonable.

B. Text Categorization

Text categorization(TC) has often been defined as the content-based assignment of one or more predefined

categories to text. TC has become important from two points of view. First, it is important from the Information Retrieval(IR) viewpoint because of the rapid growth of textual information sources. Secondly, it is important from the Machine Learning viewpoint (ML) as text categorisation provides ML with an application field. The approach that ML takes in automatic classification is to generate a mean of classification by the use of induction over examples that have been categorised previously (a form of supervised learning).

This textual documents classification method involves two main phases.

1) Training: It first examines the document, retrieves keywords deemed important and reduces dimensionality. Then it outputs an approach to classify future bookmark, inducted from the previous classification.

2) Classification: It uses the classification approach from the training process to classify new documents.

C. Feature Extraction

Text Analysis is a major application field for machine learning algorithms. However the raw data, a sequence of symbols cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.

1) The Bag of Words representation: We call vectorization the general process of turning a collection of text documents into numerical feature vectors. This specific strategy (tokenization, counting and normalization) is called the Bag of Words or "Bag of n-grams" representation. Documents are described by word occurrences while completely ignoring the relative position information of the words in the document.

2) Tf–idf term weighting: A corpus of documents can thus be represented by a matrix with one row per document and one column per token (e.g. word) occurring in the corpus.In a large text corpus, some words will be very present (e.g. "the", "a", "is" in English) hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier those very frequent terms would shadow the frequencies of rarer yet more interesting terms.

In order to re-weight the count features into floating point values suitable for usage by a classifier it is very common to use the tf–idf transform. **Tf** means term-frequency while **tf–idf** means term-frequency times inverse document-frequency. This is a originally a term weighting scheme developed for information retrieval (as a ranking function for search engines results), that has also found good use in document classification and clustering.

D. Strategies for Classification

There are many text classification strategies, the most common ones being *rule-based*, *vector-based* and *probabilistic techniques*.

1) Rule-based: For rule-based approaches to classification, a set of rules and an appropriate classifier are required. Each individual rule has a set of preconditions and an associated decision. If a document matches the preconditions, then it is classified according to the decision value.

One example of rule-based approach is Boolean Inexact Model. The main advantage of BIM is its fast speed for the computations are simple, and can be quite accurate. The weak point is that the different meanings of a single word are hard to be distinguished.

2) Vector-based: The Vector Space Model (VSM) considers document representatives as binary vectors embedded in an n-dimensional Euclidean space (n is the total number of keywords).

One example of vector-based approach is K-Nearest Neighbors(K-NN) Algorithm. This algorithm is a measurement of the similarity or distance between document and each example.

3) Probabilistic strategies: Another classic retrieval and classification method is the probabilistic retrieval technique, where the probability that a specific document will e judged relevant to a specific query, is based on the assumption that the terms are distributed in relevant and non relevant documents.

The probability formula is usually derived from Bayes's theorem which can be used to determine the probability that an example belongs to a certain class given a prior probability.Naive Bayesian Method which is based on Bayes theorem. The formula is shown as below.

$$P(\omega_i|x) = \frac{P(x|w_i)P(\omega_i)}{P(x)} \qquad i = 1, 2$$

E. Evaluation Standard for Classifications

The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$\text{F1} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Where in pattern recognition and information retrieval with binary classification, precision is the fraction of retrieved instances that are relevant, while recall

is the fraction of relevant instances that are retrieved. Both precision and recall are therefore based on an understanding and measure of relevance. Formulized as:

$$\text{precision} = \frac{\text{relevant documents} \cap \text{retrieved documents}}{\text{retrieved documents}}$$

$$\text{recall} = \frac{\text{relevant documents} \cap \text{retrieved documents}}{\text{relevant documents}}$$

F. Work Description

In this project, a simple software is especially implemented for classifying bookmarks (.html) and other context-based files, such as text/ rich-text files. Mutitype classifiers are supported in this software which can be chose according to users preference. *Python* programming language and libraries are used for seed up the development of the program. *Scikit-learn* (formerly scikits.learn) is an open source machine learning library for the *Python* programming language. It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries *NumPy* and *SciPy*. What's more, *PyQt*, a Python binding of the cross-platform GUI toolkit Qt, is used for developing graphical user interface. Like Qt. *PyQt* is free software and is implemented as a Python plug-in.

III. System Implementation

A. Implementation Patterns

There are two patterns of implementing the system: *stand-alone* and *collaborative*. *Stand-alone* pattern is based on training data. It is fast yet not accurate and hard to update. On the other hand, *Collaborative* pattern is based on the internet and shared bookmarks. It provides a potentially valuable information resource so it is more accurate, but it may compromise personal privacy. When implementing the system, the only resource we have is the public dataset, so *stand-alone* pattern is chosen for system implementation.

B. Dataset Description

Two well-known dataset are provided in system as choices, one is *webkb* dataset and the other is *20 Newsgroups*. Both of these two dataset are widely used for Text Categorization and Web page Classification projects.

1) Webkb Dataset: This data set contains WWW-pages collected from computer science departments of various universities in January 1997 by the World Wide Knowledge Base (Web->Kb) project of the CMU text learning group. The 8,282 pages were manually classified into the following categories:

- student (1641)
- faculty (1124)
- staff (137)
- department (182)
- course (930)
- project (504)
- other (3764)

The class **other** is a collection of pages that were not deemed the "main page" representing an instance of the previous six classes. (For example, a particular faculty member may be represented by home page, a publications list, a vitae and several research interests pages. Only the faculty member's home page was placed in the faculty class. The publications list, vitae and research interests pages were all placed in the other category.)

For each class the dataset contains pages from the four universities

- Cornell (867)
- Texas (827)
- Washington (1205)
- Wisconsin (1263)
- and 4,120 miscellaneous pages collected from other universities.

The files are organized into a directory structure, one directory for each class. Each of these seven directories contains 5 subdirectories, one for each of the 4 universities and one for the miscellaneous pages. These directories in turn contain the Web-pages.

2) 20 Newsgroups Dataset: The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. To the best of our knowledge, it was originally collected by Ken Lang, probably for his paper "Newsweeder: Learning to filter netnews," though he does not explicitly mention this collection. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.

The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. **comp.sys.ibm.pc.hardware / comp.sys.mac.hardware**), while others are highly unrelated (e.g. **misc.forsale / soc.religion.christian**). Here is a list of the 20 news-

groups, partitioned (more or less) according to subject matter:

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Fig. 1: 20 newsgroups dataset

C. System Framework

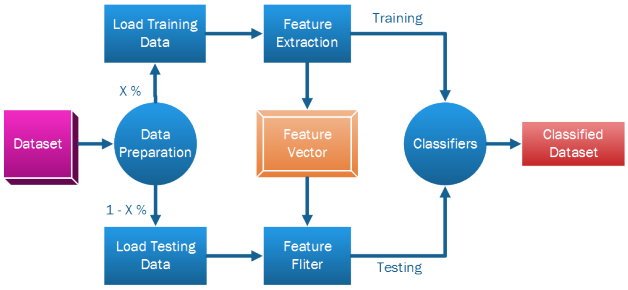

Fig. 2: System Framework

D. Module Description

1) Data Preparation: First the data is split into two parts in specific fraction(70% for training, 30% for testing.Then the data is randomly selected and moved into test folder and training folder correspondingly. This is implemented by the basic File I/O operations in Python.

2) Data Loading: The separated dataset is loaded to RAM. This is implemented by sklearn.datasets.load_files.

3) Feature Extraction: In order to extract feature, *scikit-learn* provides utilities for the most common ways to extract numerical features from text content, namely:

- **tokenizing** strings and giving an integer id for each possible token, for instance by using white-spaces and punctuation as token separators.
- **counting** the occurrences of tokens in each document.
- **normalizing** and weighting with diminishing importance tokens that occur in the majority of samples / documents.

In this scheme, features and samples are defined as follows:

- each **individual token occurrence frequency** (normalized or not) is treated as a **feature**.
- the vector of all the token frequencies for a given **document** is considered a multivariate **sample**.

As presented in the Background Knowledge section, the Tf=idf normalization is implemented by the text.TfidfTransformer class:

from sklearn.feature_extraction.text import TfidfTransformer

transformer = TfidfTransformer

transformer

TfidfTransformer(norm=...'l2', smooth_idf=True, sublinear_tf=False, use_idf=True)

4) Classification:

Implementation of multiple Classifiers

Classifier	Implementation
Decision Tree	sklearn.tree.DecisionTreeClassifier
Support Vector Machine	sklearn.svm.LinearSVC
Naive Bayes (Bernoulli)	sklearn.naive_bayes.BernoulliNB
Naive Bayes (Multinomial)	sklearn.naive_bayes.MultinomialNB

One example of sklearn class is the Support Vector Machine classifier.The definition of the SVM classifier is:

```
class sklearn.svm.SVC(C=1.0,kernel='rbf',degree
    =3,gamma=0.0,coef0=0.0,shrinking=True,
    probability=False,tol=0.001,cache_size=200,
    class_weight=None,verbose=False,max_iter
    =-1, random_state=None)
```

Core parameters

Parameter	Description
C	Penalty parameter C of the error term
kernel	Specifies the kernel type to be used in the algorithm
gamma	Kernel coefficient for 'rbf', 'poly' and 'sigm'
tol	Tolerance for stopping criterion.

E. Software Implementation

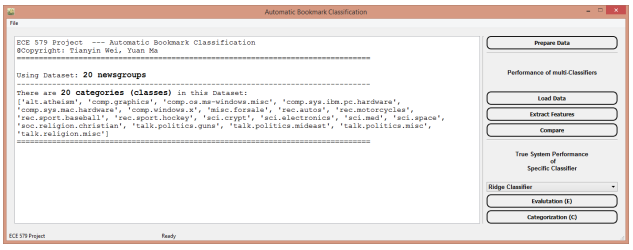

Fig. 3: Software Implementation

1) Implementing multiple classifiers: First the system loads the dataset of 20 newsgroups. Then it chooses 70% of the dataset as training data, randomly by the algorithm. Then it extracts the features of the training data using vectorizor function performed by various kinds of classifiers. Then it tests the remaining 30% of the dataset by the classifiers. The classifiers are scored according to sklearn.metrics.f1_score(y_true, y_pred, labels=None, pos_label=1, average='weighted'). By such a scoring mechanism can the classifiers be compared in a standardized method.

2) Implementing a specific classifier: After choosing a certain type of classifier in the combo box, the system chooses 70% of the whole dataset randomly as training data and the remaining 30% of the dataset as testing data, and classifies them through the chosen classifier. Then the above process is repeated three times to ensure the randomness and fairness. The webpages are therefore classified into the relevant folders, the names of which are the same as that of the classes accordingly. Finally, the system assesses the classifier by calculating the average of the three scores of the three times.

IV. System Evaluation

A. Performance of multi-classifiers

As presented in fig.4, the training time of LinearSVC is long yet the test time is relatively short comparing with the other types of classifiers. This is good because classification process usually requires training for once and test for several times. This means that once it is trained, it does not need very long time to finish the testing processes which usually last more than once. Concrete comparison is presented as below.

- Naive Bayes methods (BernoulliNB and MultinomialNB) and NearestCentroid:Less training time, much more testing time, f1 score undesirable.
- LinearSVC and PassiveAggressiveClassifier:More training time, less testing time.
- SGDClassifer:Much more training time, but the highest in f1 score.

In conclusion, RidgeClassifier and Perceptron have better performance

B. Performance of a certain classifier

Example:RidgeClassifier

```
Parameters alpha=1.0, class_weight=None, copy_X
    =True, fit_intercept=True, max_iter=None,
    normalize=False, solver=lsqr, tol=0.01
```

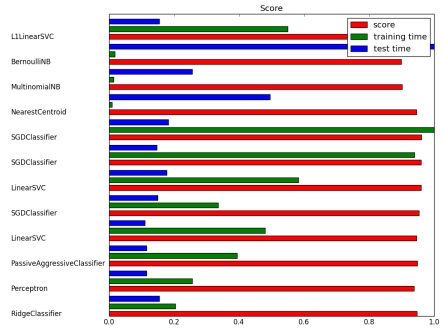

Fig. 4: performance of multi-classifiers

RidgeClassifier

Evaluation number	Training Time	Testing Time	F1 score
No. 1	6.350	0.080	0.946
No. 2	8.347	0.112	0.951
No. 3	6.941	0.083	0.949

1) Situation without error: As shown in Fig.5, the claass contains only webpages of baseball. It is because Baseball is a certain kind of sport thus other webpages can be clearly excluded.

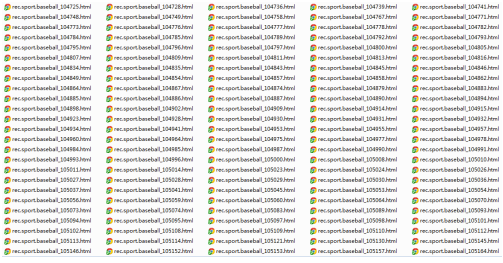

Fig. 5: Situation without error

2) Situation with minor errors: As can be seen from Fig.6, the two classes sci.crypt and talk.politics.guns are fairly far with each other, which means that they have few overlap in contents so they can be easily distinguished.

3) Situation with huge errors: As can be seen from Fig.7, the two classes alt.atheism and talk.religion.misc are in the same block, which means that they have a lot in common regarding the webpage contents. Therefore they are hard to distinguish from each another.

V. What we learnt

Through this project, we have greatly enhanced our abilities in the following several aspects. Firstly, we learnt how to apply a new third-part open source

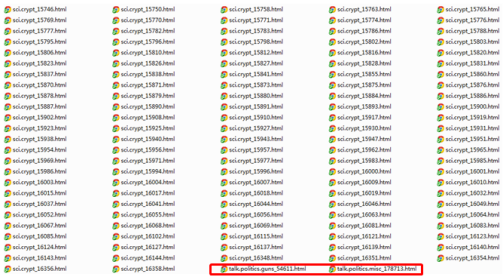

Fig. 6: Situation with minor errors

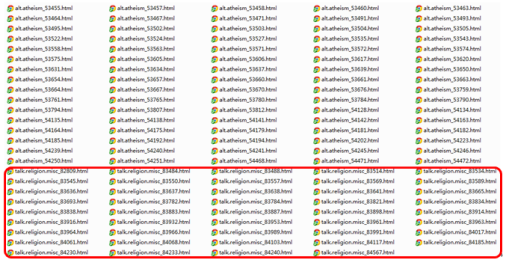

Fig. 7: Situation with huge errors

library to building a complete system in python language. Secondly, we had a better and deeper understanding in the basic procedure of machine learning both technologically and theoretically, especially in implementing as well as evaluating various types of classifiers. Moreover, we learnt to analyze the output of our system objectively and compare that with the theoretic results. We also gained our first experience with manipulating a large dataset.

In a word, we successfully applied theoretic knowledge in machine learning to solving practical problem of automatically classifying web bookmarks. Finally, we appreciate working with our partners.

VI. Conclusion

There is some justification for content-based method. It is, to some extent, both accurate and understandable. But it is raw somehow, for once there are familiar contents in different webpages which are supposed to be categorized in different classes, the classification can be quite vague.

Moreover, the process of feature extraction is time-consuming for the dimensionality of feature vectors may be so large that they become intractable.

Additionally, the dataset is too small to be representative of the real world.

In order to solve the above issues respectively, more parameters combining with webpage contents should be taken into account in the first place, such as URL and the web title. Such elements, besides webpage contents, can be essential to justifiably classifying bookmarks as well.

Dimension reduction could be utilized to speed up the whole system.

Complex dataset could be applied to enlarge the present one. For instance, Wikipedia-page articles.

The GUI can be implemented in a collaborative pattern based on Cloud computation to share the application.

References

[1] Richard Jensen. *"A Rough Set-Aided System for Sorting WWW Bookmarks."* MSc in Artificial Intelligence Division of Informatics, The University of Edinburgh, 2000.

[2] Dominik Benz, Karen H. L. Tso, Lars Schmidt-Thieme. *"Automatic Bookmark Classification A Collaborative Approach"* Computer-based New Media Group (CGNM), Department of Computer Science, University of Freiburg, 2006.

[3] Soon-Hee Myoung, Jung-Min Choi, In-Cheol. *"BClassifier A Personal Agent for Bookmark Classification"* Department of Computer Science, Kyonggi University, 2001

[4] Bachelor-Thesis von Bastian Christoph aus Fritzlar. *"Multilabel Learning for personalized Bookmark Classification"* TECHNISCHE UNIVERSITAT DARMSTART, 2013

[5] Weiling Cai, Songcan Chen, and Daoqiang Zhang. *"A Multiobjective Simultaneous Learning Framework for Clustering and Classification"* IEEE TRANSACTIONS NEURAL NETWORKS VOL 21. NO 2. FEBRUARY 2010

[6] Ronen Feldman, James Sanger. *"THE TEXT MINING HANDBOOK - Advanced Approaches in Analyzing Unstructured Data"* CAMBERAGE UNIVERSITY PRESS, 32 Avenue of Americas, New York, NY 10013-2473, USA

[7] Steven Bird, Ewan Klein, and Edward Loper. *"Natural Language Processing with Python"* O'Reilly Medi, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.