UNIVERSITÀ DI PISA

Data Mining and Machine Learning

Educational Version Project

2020

D. M. Maggi

M. Gomez Gomez

July 29, 2020

Contents

Chapter 2

Introduction

2.1 WikiHow

The dataset is made up of WikiHow articles. Our idea born because the web site of WikiHow have a human community for manually classify the articles published. However, manually creating labels is not only time-consuming but also subject to human errors, and eventually, becomes impossible for a very large amount of data.
To minimize the human effort in labeling, we propose a unified multi-class active learning approach for automatically labeling articles.
We include extending active learning from string classes to multiple classes and evaluating several practical sample selection strategies.
The experimental results show that the proposed approach works effectively even with a significantly reduced amount of labeled data.

In WikiHow, each article belongs to one and only one of 19 macro-categories:

1. *Arts and Entertainment*
2. *Cars and Other Vehicles*
3. *Travel*
4. *Work World*
5. *Relationships*
6. *Philosophy and Religion*

7. *Family Life*
8. *Finance and Business*
9. *Computers and Electronics*
10. *Food and Entertaining*
11. *Home and Garden*
12. *Youth*
13. *Education and Communications*
14. *Health*
15. *Hobbies and Crafts*
16. *Personal Care and Style*
17. *Sports and Fitness*
18. *Holidays and Traditions*
19. *Pets and Animals*

Each macro-category may have several subsections, but this hasn't been taken into account during our work, as WikiHow's taxonomy has a strict multi-tree single-root structure. Each article is characterized by:

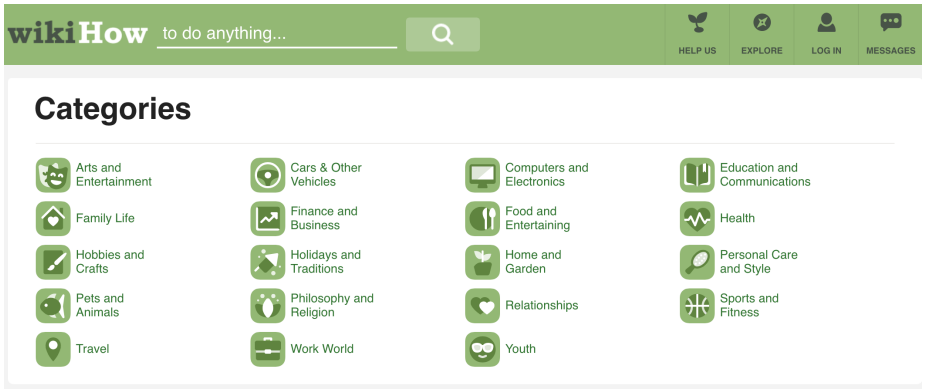

Figure 2.1: WikiHow's main categories

- its *title*, always beginning with the prefix "How to..."

- a short *summary* at the top of the page, containing the main arguments of the article.
- the *text* of the article, divided into numbered paragraphs.

At the moment, the categorization of the articles is entirely manual: the content creator has the responsibility to label the submitted work. A team of volunteers routinely double-checks the newly created or modified articles.
Our interest lies in providing an automatic tool for classifying a new document into one of the 19 macro categories.

Chapter 3

Data Processing

3.1 Dataset building

3.1.1 Scraping

Data have been scraped using two Python scripts, *parser.py* and $link_h arvester.py$. For each category, the relative paths to subcategory pages have been extracted and saved in auxiliary .txt files (which can be found in parser/texts) in order to facilitate the scraping procedure.
Below an example of the category-subcategory relationship.

```
/Category:Arts-and-Entertainment
   /Category:Amusement-and-Theme-Parks
   /Category:Carnivals
   /Category:Disneyland-and-Disney-World
   /Category:Roller-Coasters
   /Category:Water-Parks
   /Category:Artwork
   /Category:Art-Appreciation
   /Category:Art-Collection
   /Category:Art-Equipment
   /Category:Art-Journals
   /Category:Art-Media
   /Category:Art-Studies
   ...
```

The subcategories has been retrieved according to the following procedure:

```
FOR EACH CATEGORY IN \textit{CATEGORY LIST}:
    GET THE HTML PAGE OF THE SUBCATEGORY
    PARSE THE HTML
    FIND THE "subcats" <div>
    GET THE LINKS
    FOR EACH CATEGORY.TXT
        WRITE THE SUBCATEGORY RELATIVE PATH
```

For each subcategory, we proceed to scrape WikiHow pages according to this general scheme:

```
FOR EACH CATEGORY IN CATEGORY LIST:
    PARSE THE CATEGORY.TXT FILE
    FOR EACH SUBCATEGORY;
        PARSE ARTICLE PAGE
        EXTRACT TITLE; SUMMARY; DESCRIPTION
        PRE-PROCESS TITLE; SUMMARY; TEXT
        SAVE [TITLE; SUMMARY; DESCRIPTION; CATEGORY] TO
        CATEGORY.CSV FILE
```

In order to balance the dataset, from each of the 19 separate .csv files 350 rows have been extracted and merged into a single *AllCategories.csv* file, which was our balanced, final dataset.
The pre-processing phase is analyzed in greater detail in the following section.

3.1.2 Pre-processing phase

After gathering all of the documents making up our dataset, we have applied to the instances the following operations:

- lowercase the text and remove everything that isn't a letter (using *re* package).
- remove all the English *stop-words*, that is, those considered too common to be significant, present in the text (using *nltk.corpus*).
- words have been tokenized and reduced to *stems*, that is, separate suffixes from root of the words (using *SnowballStemmer* from *nltk.stem.snowball* package).

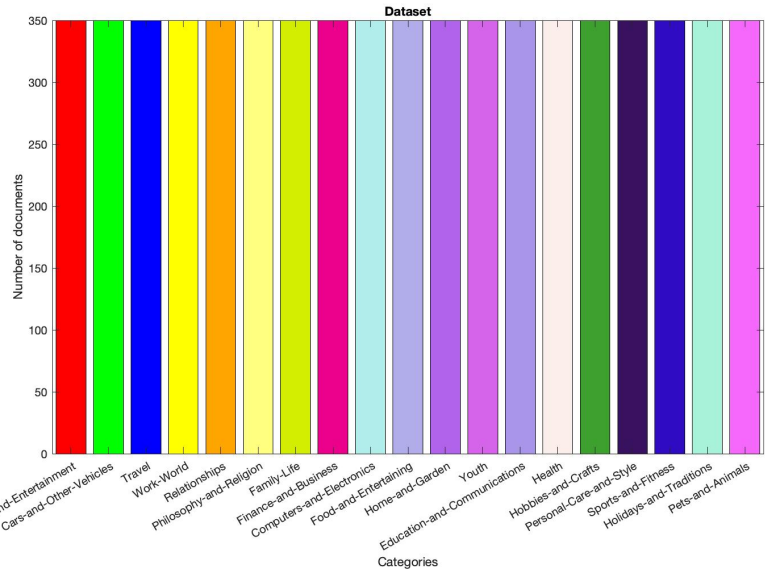

Figure 3.1: A balanced dataset has been obtained

- normalization, entity extraction and spelling and grammar correction have been carried out.

After this initial clean-up, the newly-balanced dataset is stored in *All_Categories350.csv*.

AllCategories

TITLE	SUMMARY	TEXT	CATEGORY
talent multipl area	set increas talent abil multipl disciplin audaci endeavor also feasit	practic whatev tri talent know practic especi true hope talent multipl ar	Arts-and-Entertainment
make illumin manuscript	middl age book written hand meant imag paint hand illumin manu	gather piec parchment classic manuscript made thick white parchment	Arts-and-Entertainment
peopl watch	peopl watch art cultur old citi like pari neur french word someon s	find crowd street cafe cafe classic peopl watch destin mani peopl visit	Arts-and-Entertainment
reduc entertain expens	spend money movi special event cost surpris amount time averag	purcha cabl altern first step reduc home entertain cost ditch cabl instea	Arts-and-Entertainment
support art	art allow peopl public express fun creativ meaning way art music	donat money nation art organ help big scale creativ project need fund s	Arts-and-Entertainment
contact art galleri	reach art galleri easi take lot courag confid organ well strong sens	arrang recent art sampl physic digit portfolio collect realli high qualiti pi	Arts-and-Entertainment
read ulyss	come ulyss consid mani second hardest book english languag m	understand ulyss learn read ulyss know get ulyss compri episod episo	Arts-and-Entertainment
wear cosplay wig	cosplay wig worn make intend charact come life cosplay wig high	make sure wig right size head may alreadi figur wig right size head cas	Arts-and-Entertainment
good stage presenc	good stage presenc crucial creat engag last perform talent practi	take everi opportun practic get practic make feel confid big show venu	Arts-and-Entertainment

Figure 3.2: Snapshot from AllCategories350.csv

3.1.3 Validation

The results of the various classifiers employed have been evaluated by selecting randomly 20% of the dataset entries as a *testing* set and the remaining 80% as a *training* set.

3.1.4 Filtering

The following filters have been applied:

- Nominal to string - for converting the first three nominal attributes to strings
- string to word vector - for converting the newly obtained string attributes into a set of numeric attributes representing word occurrence information from the text contained in the strings.

3.2 Data Mining

The one posed in this project is a *multi-class classification* problem, because each article can belong to one and only one class. A multi-class model is employed for training for each type of classifier.
Our aim was to compare different classifier and evaluate their results on our dataset. In order to do so, we will compare the results of each classifier with respect to their *precision*, *recall*, *f-measure* and *accuracy*.

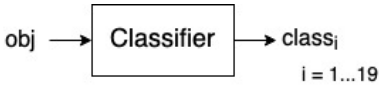

Figure 3.3: Classification pipeline in a multi-class classification scenario

3.2.1 Random Forest

Random Forest is a classification *ensemble method*. The "forest" is made up of an ensemble of *decision tree* classifiers. Each tree depends on the values of a random vector sampled independently and sharing the same distributions as all the other trees in the forest.
During the classification phase, each tree expresses its vote and the class returned globally is the one reaching the highest consensus.

3.2.2 Naïve Bayes

Statistical classifier expressing class membership as a *probability*.
Let X be the *evidence* whose class label is unknown.
Let H be the *hypothesis* that X belongs to class C.

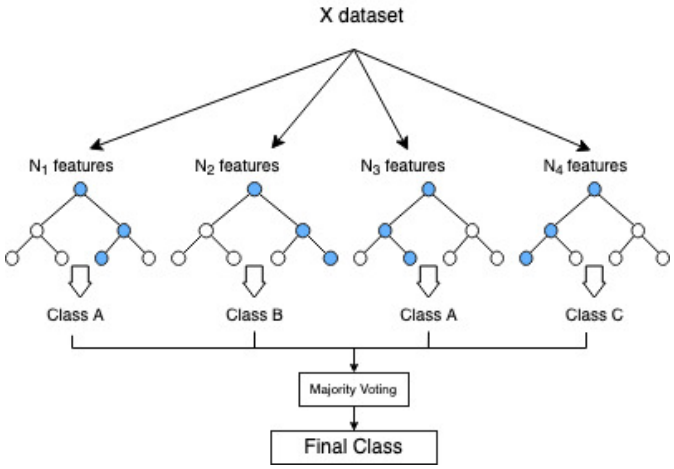

Figure 3.4: Random Forest ensemble voting procedure

Bayes theorem helps computing $P(H|X)$, that is the probability that X belongs to class C, given its characteristics:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

The aim of a Bayesian classifier will be to determine the class C_i having the higher posterior probability $P(C_i|X)$.
In the context of multi-class text classification, this can be reformulated as finding

$$\arg\max_k P(class_k|doc_i), \quad i = 1 \dots k$$

In other words, for each class k, the probability that document i belongs to it is computed and the class with the highest value is selected.

3.2.3 SMO

The Sequential Minimal Optimization (SMO) algorithm is employed for solving the quadratic problem arising in the training of Support Vector Machines (SVM).
It employs heuristics to partition the training problem into smaller problems that can be solved analytically.

Chapter 4

Implementation

4.1 Pre-processing with Python

The *parser* section of this project (data harvesting and pre-processing) has been developed in Python, while the *data mining* module (classification task and application) has been implemented in Java, exploiting Weka APIs.

4.1.1 Models

The following section explains the scripting of the data and the prepossessing .

4.1.2 Code structure - *parser*

The *parser* module of this project carries out the data harvesting and preprocessing tasks. It is made up of three Python scripts, *link_harvester.py*, *spider.py* and *cleanCSV.py*.

link_harvester.py

This script scrapes the */Special* : *CategoryListing* page of WikiHow in order to gather, for all the 19 macro-categories, the relative paths of the subcategories. The results are saved in */texts* folder.
The paradigm is the one described in section 1.1.

spider.py

This scripts uses the relative paths retrieved by *link_harvester.py* and contained in */texts* to scrape the articles from the (sub-)categories of interest.
The information of the article is saved on the relative .csv file in the following format:

TITLE SUMMARY TEXT CATEGORY

The *main* is the following:

```
lista=category
for cat in lista:
    glbCategory = cat
    with open('./texts/' + cat + '.txt', 'r') as fp:
        articleInfo = {}
        line = fp.readline()
        while line:
            result=parseCat("https://www.wikihow.com"+line)
            line = fp.readline()
        fp.close()
```

- **parseCat** harvests all the links to WikiHow articles for a particular (sub-)category of category *cat*. Method **parsePage** is invoked on all the harvested links.

```
def parseCat(url):
    html_text = requests.get(url).text
    soup=BeautifulSoup(html_text, 'html.parser')
    li = soup.find_all("div", {"class": "responsive_thumb"})
    links = [div.find('a')['href'][23:] for div in li]
    [parsePage(url) for url in links]
```

- **parsePage** retrieves the HTML of the page located at *url* and extracts from it the *title*, the *summary* and *text*. These components are then pre-processed by the means of **cleanDescription** before the row about the document can be added to the appropriate .scv.

```
def parsePage(url):
    try:
        html_text = requests.get("https://www.wikihow.com"+url).text
        soup=BeautifulSoup(html_text, 'html.parser')
        title=soup.find_all("a", {"href": "https://www.wikihow.com"
        +url}, id=False)
        if title != [] and len(title) == 1:
            summary = soup.find("div", {"id": "mf-section-0"})
            if summary is None:
                print("None Summary")
            else:
                summary = summary.getText().lower()
                summary = cleanDescription(summary)
                text=getText(soup)
                text = cleanDescription(text)
                title = cleanDescription(title[0].getText())
                transformToCSV(title, summary, text,
                    glbCategory)
    except requests.ConnectionError:
        print("Connection Error")
```

- **cleanDescription** is the method invoked for cleaning a text, removing from it all the *stop-words*, tokenize and stem it. Regular expressions are used for removing all the lexicographical elements which aren't lower-case letters.

```
def cleanDescription(description):
    description = description.lower()
    description = re.sub("\\b\\w{0,2}\\b|[^a-zA-Z ]",
        " ", str(description))
    description = re.sub("([^a-zA-Z\\s+\\w]|\\s+)",
        " ", str(description))
    description = re.sub("\\W", " ", str(description))
    description = re.sub("\\s+", " ", str(description))
    description = re.sub("^\\s+", "", str(description))
    description = re.sub("\\s+$", "", str(description))
    if description != ' ' and description != '':
        lang = detect(description)
        if lang == 'en':
            stop_words = set(stopwords.words('english'))
            word_tokens = word_tokenize(description)
            filtered_sentence = []
            [filtered_sentence.append(w)
                for w in word_tokens
                    if w not in stop_words ]
            stemmer = SnowballStemmer('english')
            stemSentence = ''
            for word in filtered_sentence:
                stem = stemmer.stem(word)
                stemSentence += stem
                stemSentence += ' '
            description = re.sub("\\b\\w{0,2}\\b|[^a-zA-Z ]",
                " ", str(stemSentence))
            description = re.sub("([^a-zA-Z\\s+\\w]|\\s+)",
                " ", str(description))
            description = re.sub("\\W", " ", str(description))
            description = re.sub("\\s+", " ", str(description))
            description = re.sub("^\\s+", "", str(description))
            description = re.sub("\\s+$", "", str(description))
            return description
        else:
            return ' '
```

```
    else:
        return ' '
```

- **getText** is an auxiliary method invoked for cleaning the text of the article, which is retrieved in paragraphs as per WikiHow's HTML document structure.

```
def getText(soup):
    tex=soup.find_all("div", {"class": "step"})
    tex = [cleanDescription(it.getText()) for it in tex]
    return " ".join(tex)
```

cleanCSV.py

This script builds the final dataset, artificially balanced by selecting 350 entries for each of the 19 categories. This has been done for two reasons:

1. to avoid over-fitting the model (*curse of dimensionality*).
2. to draw a representative sample from each macro-category and still obtain a balanced dataset.

4.1.3 Use Case and UML

The UML figure can be seen at figure 1.13.

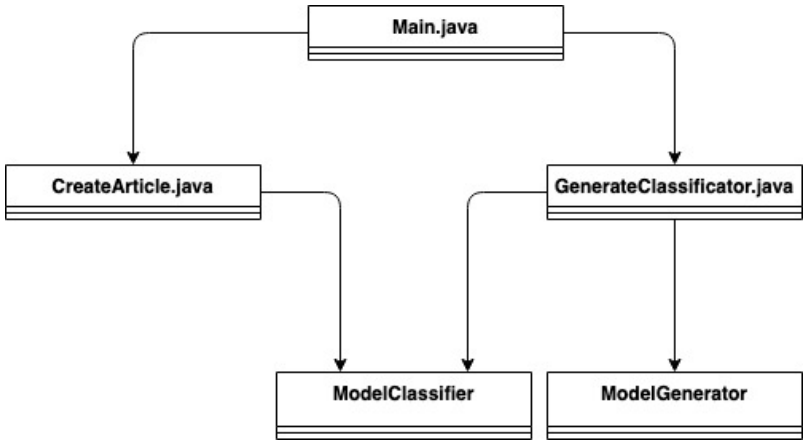

Figure 4.1: UML diagram

- **Main.java** is the main thread of the application. It instantiates the main frame of the application, as well as the children ones, the one connected to CreateArticle.java and the one connected to GenerateClassificator.java (accessible by clicking the buttons in figure 1.14).
- **ModelClassifier.java** contains the logic for applying the classifiers.
- **ModelGenerator.java** generates the model for the chosen classifier, based on a user-given train.arff file.
- **CreateArticle** is the class responsible for handling the creation of a new article and its automatic categorization. Its frame is in figure 1.15
- **GeneraterClassificator** is the class responsible for generating the model corresponding to the application of a given classifier to a user-given train.arff file.

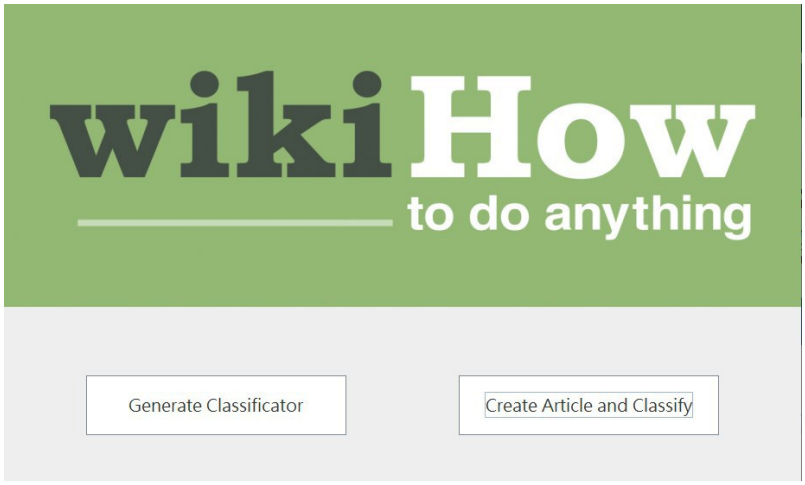

Figure 4.2: Initial frame of the application

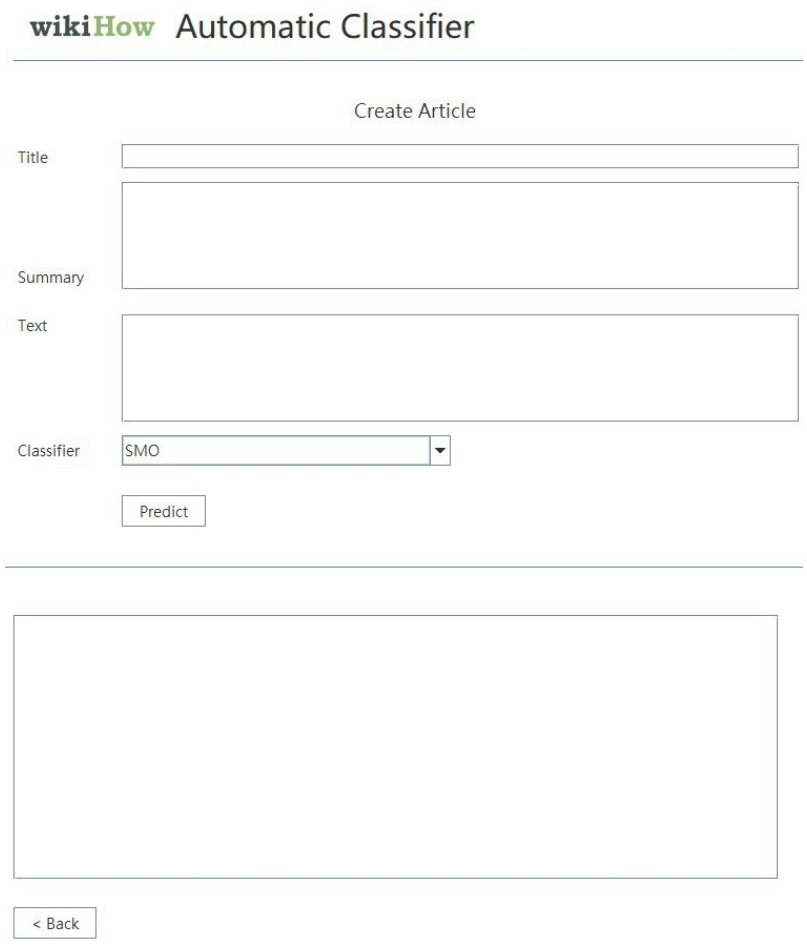

Figure 4.3: Here you can enter a new article, choose a classifier and classify it

Figure 4.4: Here you create the model

Chapter 5

Results

5.1 Random Forest

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	<--	classified as
177	2	40	17	5	5	0	7	9	8	8	5	7	16	13	18	6	6	1	|	a = Arts-and-Entertainment
0	**235**	11	4	1	25	3	2	5	9	0	9	2	0	3	18	12	11	0	|	b = Cars-and-Other-Vehicles
13	9	**293**	13	0	2	1	1	2	2	0	1	3	1	0	8	1	0	0	|	c = Computers-and-Electronics
11	2	30	**176**	6	11	5	5	12	6	10	6	2	15	11	3	6	22	11	|	d = Education-and-Communications
1	2	1	4	**195**	31	1	5	4	2	5	7	2	20	40	3	5	6	16	|	e = Family-Life
1	1	10	7	2	**286**	1	0	1	16	0	2	4	2	2	0	1	14	0	|	f = Finance-and-Business
1	3	4	3	1	1	**300**	13	4	2	3	3	1	1	2	1	3	4	0	|	g = Food-and-Entertaining
4	3	3	7	8	4	25	**224**	3	0	2	5	7	21	6	17	0	4	7	|	h = Health
4	5	4	6	7	3	8	6	**272**	4	0	8	4	6	2	3	1	5	1	|	i = Pets-and-Animals
8	17	34	9	2	25	10	1	31	**159**	7	4	12	5	3	11	5	5	2	|	j = Hobbies-and-Crafts
7	1	7	3	2	4	34	2	21	7	**210**	1	4	13	17	1	8	3	5	|	k = Holidays-and-Traditions
0	22	12	2	5	15	19	6	17	6	5	**191**	8	5	1	12	15	8	1	|	l = Home-and-Garden
10	8	3	12	2	18	16	24	22	11	2	7	**137**	6	12	21	1	24	14	|	m = Personal-Care-and-Style
2	2	5	6	14	8	5	12	0	0	8	1	0	**244**	25	1	1	8	8	|	n = Philosophy-and-Religion
0	0	3	0	15	0	0	0	2	0	3	0	2	3	**313**	0	1	3	5	|	o = Relationships
6	12	7	2	5	2	6	14	11	8	1	8	6	5	3	**242**	2	8	2	|	p = Sports-and-Fitness
7	8	11	1	6	9	3	5	2	2	8	7	2	9	5	2	**258**	3	1	|	q = Travel
2	2	12	17	11	43	2	7	0	7	3	2	8	16	18	4	9	**181**	6	|	r = Work-World
3	0	1	3	21	1	6	25	2	1	4	2	10	10	46	8	3	4	**200**	|	s = Youth

Figure 5.1: Random Forest Confusion Matrix

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	<--	Class
	0,506	0,013	0,689	0,506	0,583	0,571	0,920	0,628	l	Arts-and-Entertainment
	0,671	0,016	0,704	0,671	0,687	0,670	0,957	0,754	l	Cars-and-Other-Vehicles
	0,837	0,031	0,597	0,837	0,697	0,688	0,977	0,768	l	Computers-and-Electronics
	0,503	0,018	0,603	0,503	0,548	0,528	0,898	0,481	l	Education-and-Communications
	0,557	0,018	0,633	0,557	0,593	0,573	0,920	0,593	l	Family-Life
	0,817	0,033	0,580	0,817	0,679	0,668	0,965	0,760	l	Finance-and-Business
	0,857	0,023	0,674	0,857	0,755	0,745	0,973	0,851	l	Food-and-Entertaining
	0,640	0,021	0,624	0,640	0,632	0,611	0,940	0,663	l	Health
	0,779	0,023	0,648	0,779	0,707	0,693	0,965	0,782	l	Pets-and-Animals
	0,454	0,014	0,636	0,454	0,530	0,516	0,870	0,508	l	Hobbies-and-Crafts
	0,600	0,011	0,753	0,600	0,668	0,656	0,951	0,763	l	Holidays-and-Traditions
	0,546	0,012	0,710	0,546	0,617	0,604	0,930	0,642	l	Home-and-Garden
	0,391	0,013	0,620	0,391	0,480	0,471	0,901	0,506	l	Personal-Care-and-Style
	0,697	0,024	0,613	0,697	0,652	0,633	0,955	0,709	l	Philosophy-and-Religion
	0,894	0,033	0,600	0,894	0,718	0,715	0,977	0,815	l	Relationships
	0,691	0,021	0,649	0,691	0,669	0,651	0,948	0,682	l	Sports-and-Fitness
	0,739	0,013	0,763	0,739	0,751	0,738	0,950	0,771	l	Travel
	0,517	0,022	0,567	0,517	0,541	0,517	0,920	0,544	l	Work-World
	0,571	0,013	0,714	0,571	0,635	0,621	0,938	0,675	l	Youth
Weighted Avg.	0,646	0,020	0,651	0,646	0,639	0,625	0,940	0,679	l	

Figure 5.2: Random Forest Results

5.2 Naïve Bayes

```
  a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   <-- classified as
180   2  43  13   2   3   0   4   3  13   9   5   7  10   6  22   8  18   2 |   a = Arts-and-Entertainment
  3 203  18   8   1  21   0   1  19   6   0  17   1   0   2   7  11  31   1 |   b = Cars-and-Other-Vehicles
 27  15 273   2   0   1   0   2   5   9   0   5   1   0   0   8   0   0   2 |   c = Computers-and-Electronics
 12   3  30 166   6   4   0   2  22  25   8   5   5  12   3   2   5  31   9 |   d = Education-and-Communications
  2   0   5   3 206  22   1   4   1   4  13  12   2  22  17   0   8  13  15 |   e = Family-Life
  0   3  16   6   1 259   0   0   1  14   1   3   3   2   0   0   1  40   0 |   f = Finance-and-Business
  1   2   6   3   1   1 269  17   9   0  18   9   0   5   0   2   2   3   2 |   g = Food-and-Entertaining
  3   2   5   8   7   5  21 233   5   3   1   6   1  17   2  15   0   8   8 |   h = Health
  3   0  22  11   4   2   1   5 255   5   1  24   1   2   0   1   1   6   5 |   i = Pets-and-Animals
 14  10  25  10   2  13   3   0  34 185   9  14   4   2   1   5   1  16   2 |   j = Hobbies-and-Crafts
  5   0   9   3   1   1  22   0  29  11 209   9   6  14   8   0   6   9   8 |   k = Holidays-and-Traditions
  0  15   9   0   6  18   7   1  32   8   6 214   4   2   0   6   8   7   7 |   l = Home-and-Garden
 10   7   4   4   1   6  11  29  38  13   2   8 129   4   7  17   3  45  12 |   m = Personal-Care-and-Style
  1   1  10   5   8   2   2  11   2   6  15   2   1 247  10   2   1   9  15 |   n = Philosophy-and-Religion
  1   0   3   1  18   0   0   0   0   0   4   0   4  11 291   0   1   2  14 |   o = Relationships
  3  13  11   5   0   1   3  15  11   5   6  11   6   0   0 244   0   9   7 |   p = Sports-and-Fitness
  5   8   7   3   6   9   2   5   2   8  14   9   3  11   2   2 245   7   1 |   q = Travel
  6   6  15  17   8  20   0   3   1   6   3   3  13  21   6   3   4 197  18 |   r = Work-World
  1   1   3   2  17   0   4  29   6   0  14   2  12  10  20   2   1   4 222 |   s = Youth
```

Figure 5.3: bayes - confusion matrix

We can see the weighted on the last line.

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	<--	Class
0,514	0,015	0,650	0,514	0,574	0,558	0,896	0,567	|	Arts-and-Entertainment
0,580	0,014	0,698	0,580	0,633	0,618	0,943	0,668	|	Cars-and-Other-Vehicles
0,780	0,038	0,531	0,780	0,632	0,620	0,950	0,587	|	Computers-and-Electronics
0,474	0,017	0,615	0,474	0,535	0,518	0,882	0,512	|	Education-and-Communications
0,589	0,014	0,698	0,589	0,639	0,623	0,944	0,669	|	Family-Life
0,740	0,020	0,668	0,740	0,702	0,685	0,959	0,670	|	Finance-and-Business
0,769	0,012	0,777	0,769	0,773	0,760	0,957	0,756	|	Food-and-Entertaining
0,666	0,020	0,645	0,666	0,655	0,636	0,947	0,590	|	Health
0,731	0,035	0,537	0,731	0,619	0,602	0,943	0,675	|	Pets-and-Animals
0,529	0,022	0,576	0,529	0,551	0,528	0,905	0,591	|	Hobbies-and-Crafts
0,597	0,020	0,628	0,597	0,612	0,591	0,922	0,601	|	Holidays-and-Traditions
0,611	0,023	0,598	0,611	0,605	0,582	0,943	0,579	|	Home-and-Garden
0,369	0,012	0,635	0,369	0,467	0,463	0,893	0,504	|	Personal-Care-and-Style
0,706	0,023	0,630	0,706	0,666	0,647	0,947	0,701	|	Philosophy-and-Religion
0,831	0,013	0,776	0,831	0,803	0,792	0,983	0,838	|	Relationships
0,697	0,015	0,722	0,697	0,709	0,694	0,941	0,696	|	Sports-and-Fitness
0,702	0,010	0,801	0,702	0,748	0,737	0,956	0,789	|	Travel
0,563	0,041	0,433	0,563	0,489	0,462	0,902	0,421	|	Work-World
0,634	0,020	0,634	0,634	0,634	0,614	0,935	0,602	|	Youth
Weighted Avg. 0,636	0,020	0,645	0,636	0,634	0,617	0,934	0,632		

Figure 5.4: Bayes Results

5.3 SMO

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	<--	classified as
256	0	22	21	2	2	0	2	1	12	1	3	5	2	3	4	4	9	1	|	a = Arts-and-Entertainment
1	**285**	5	1	2	9	1	0	2	6	2	10	3	0	1	10	8	4	0	|	b = Cars-and-Other-Vehicles
13	6	**296**	15	0	4	0	1	0	9	0	1	0	0	0	5	0	0	0	|	c = Computers-and-Electronics
14	2	19	**247**	2	4	0	5	1	13	6	4	1	5	2	0	4	13	8	|	d = Education-and-Communications
1	0	1	4	**267**	8	2	4	1	3	2	7	3	12	12	0	3	6	14	|	e = Family-Life
0	4	3	7	4	**279**	0	0	0	22	0	2	3	1	0	0	3	22	0	|	f = Finance-and-Business
2	1	2	5	1	0	**302**	7	2	3	5	5	7	1	0	2	0	3	2	|	g = Food-and-Entertaining
2	1	0	6	9	4	12	**264**	4	0	0	5	9	10	1	14	1	4	4	|	h = Health
3	4	1	3	3	0	2	4	**300**	9	4	8	3	1	0	0	1	2	1	|	i = Pets-and-Animals
14	13	17	12	2	20	0	0	14	**219**	6	7	13	1	2	3	2	3	2	|	j = Hobbies-and-Crafts
9	1	3	3	3	2	12	2	9	1	**269**	3	4	10	5	0	7	5	2	|	k = Holidays-and-Traditions
3	11	2	0	6	7	7	4	7	6	4	**268**	9	1	0	6	3	4	2	|	l = Home-and-Garden
4	2	2	2	1	3	0	15	1	17	1	9	**240**	5	6	8	1	19	14	|	m = Personal-Care-and-Style
4	0	1	4	11	5	3	9	0	2	9	2	1	**274**	9	2	2	11	1	|	n = Philosophy-and-Religion
0	0	2	1	19	0	0	1	0	0	5	0	6	1	**300**	0	1	1	13	|	o = Relationships
3	16	1	0	3	1	2	12	4	5	0	9	10	0	0	**275**	3	4	2	|	p = Sports-and-Fitness
6	11	2	7	8	5	2	5	1	4	4	6	3	3	0	1	**274**	2	5	|	q = Travel
3	7	7	11	8	18	0	6	0	5	1	1	14	15	6	0	3	**240**	5	|	r = Work-World
2	1	0	9	14	0	2	16	1	0	5	3	8	4	18	5	6	2	**254**	|	s = Youth

Figure 5.5: SMO Confusion Matrix

We can see the weighted on the last line.

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	<--	Class
	0,731	0,013	0,753	0,731	0,742	0,728	0,952	0,659	|	Arts-and-Entertainment
	0,814	0,013	0,781	0,814	0,797	0,786	0,977	0,733	|	Cars-and-Other-Vehicles
	0,846	0,014	0,767	0,846	0,804	0,794	0,988	0,752	|	Computers-and-Electronics
	0,706	0,018	0,690	0,706	0,698	0,681	0,942	0,580	|	Education-and-Communications
	0,763	0,016	0,732	0,763	0,747	0,733	0,951	0,642	|	Family-Life
	0,797	0,015	0,752	0,797	0,774	0,761	0,975	0,689	|	Finance-and-Business
	0,863	0,007	0,870	0,863	0,867	0,859	0,976	0,817	|	Food-and-Entertaining
	0,754	0,015	0,739	0,754	0,747	0,733	0,963	0,646	|	Health
	0,860	0,008	0,862	0,860	0,861	0,853	0,981	0,802	|	Pets-and-Animals
	0,626	0,019	0,652	0,626	0,638	0,619	0,931	0,505	|	Hobbies-and-Crafts
	0,769	0,009	0,830	0,769	0,798	0,788	0,971	0,724	|	Holidays-and-Traditions
	0,766	0,013	0,759	0,766	0,762	0,749	0,961	0,659	|	Home-and-Garden
	0,686	0,016	0,702	0,686	0,694	0,677	0,942	0,579	|	Personal-Care-and-Style
	0,783	0,011	0,792	0,783	0,787	0,776	0,967	0,713	|	Philosophy-and-Religion
	0,857	0,010	0,822	0,857	0,839	0,830	0,986	0,770	|	Relationships
	0,786	0,010	0,821	0,786	0,803	0,792	0,966	0,724	|	Sports-and-Fitness
	0,785	0,008	0,840	0,785	0,812	0,802	0,969	0,750	|	Travel
	0,686	0,018	0,678	0,686	0,682	0,664	0,940	0,543	|	Work-World
	0,726	0,012	0,770	0,726	0,747	0,734	0,962	0,646	|	Youth
Weighted Avg.	0,769	0,013	0,769	0,769	0,768	0,756	0,963	0,681		

Figure 5.6: SMO Results

Chapter 6

Conclusion

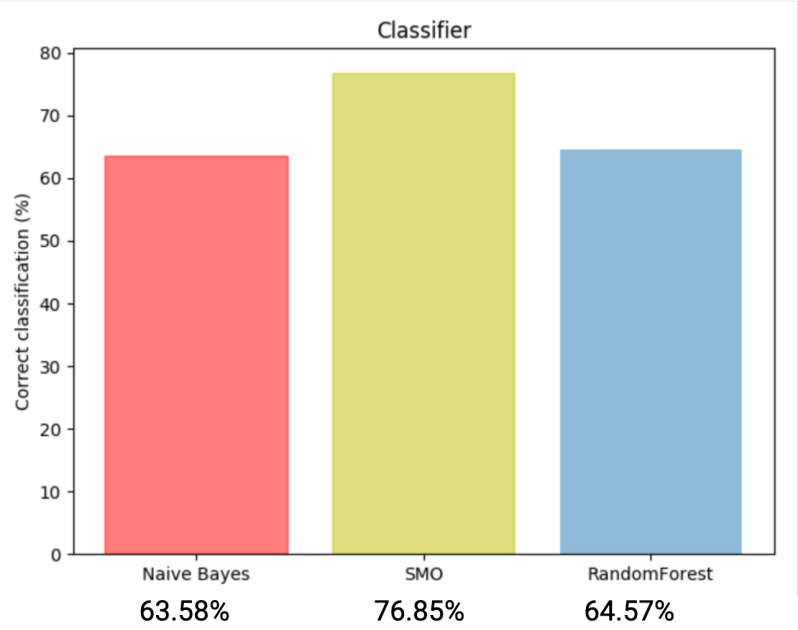

Figure 6.1: Comparison of results

6.1 Experiment Conclusions

Our experiments demonstrate that an active learner with careful sample selection can achieve good performance (23.15% labeling error) with much less

human labeling effort. Also, an active learner with the proposed sample selection strategies can do much better than one with only a random sampling strategy, which yields an over two-fold error reduction.

In some cases, the test set might have some problems with the data new labels which do not exist in the training set. This makes the learning problems. For example, two documents belonging to the same macro-category may actually be located in different branches of the single-root tree growing from it, and thus be different, to a point, from one another.