Generation of page classification Ruleset from Corpus statistics

Introduction

The idea of automatic ruleset generation comes from availability of large, pre-classified corpus of pages which URLs, titles and assigned categories are known. A question how such corpus was tagged with interest categories is tangential - plenty of full-text categorization methods exist to enable such tagging. Instead, we focus entirely on how to generate high-precision, high-recall, small-size ruleset that can categorize pages based ONLY on URL and title (which is what's available to a browser).

Suppose that the corpus contains 10,000 stories from gizmodo.com, and puts 90% of the gizmodo pages into Technology category, then we know that an arbitrary page from gizmodo is about Technology with 90% accuracy. More importantly, we are assured of a favorable user reaction: since gizmodo.com is predominantly about technology, then a user is not surprised when Interest Dashboard (ID) associates his technology interest with visits to gizmodo. We are OK to make an explict, "**topical**" rule that uncoditionally places any visit to gizmodo.com into Technology.

Having just the "topical" sites will not be sufficient for large coverage of a user behavior. Mass user often consumes content on large, all embracing sites like news.yahoo.com or cnn.com. These sites, however, have topical sections that are often identified by pattern in a story url. For example, political stories on cnn.com and nytimes.com will have /politics/ in thier urls. Tennis

sections are likely to have /sports/tennis/ pattern, while fashion stories will have /fashion/, or /showbiz/ or /style/ in their urls.

Each of such site sections pages that tend to belong to a single subject. Hence, we can blindly generate patterns from site URLs and choose those whose pages are on single topic. This procedure effectively generates URL path matching rules for "**topical**" sections of the site:

- nytimes.com.*/politics/ → politics
- cnn.com.*/showbiz/ → life style
- news.yahoo.com.*/style/ → fashion
- ….

This techniques exploit the fact that editors often place topical word into a story url to assist navigation across the site.

The same idea can be applied to use of "topical" terms and bigrams from page title. For example, "ebola", when mentioned in a news title, is highly indicative to "health" topic. Just like consistent visits to gizmodo.com put a user into Technology category, the consistent reading of news that mention "ebola" in title - is a good indication that a user has Health interest.

These examples suggest a strategy that uses corpus statistics to generate potential rule candidates and select the ones that provide for high coverage at prescribed precision level. A more formal description is given below.

Rule Generation

Rules are blindly generated from the corpus of already classified documents. Every rules has a form of a pattern that can be matched in URL or title of a page, total number of document the pattern matches in the corpus, and the empirical counts of documents that matched in each category.

A typical rule may have a form:

obama_T 1000 {"business":120,"politics":990,"society":990,"science":110}

The rule above says that there are 1000 documents in the corpus that mention obama in title. Out of these 1000: classified into "business":120, "politics":990, "society":990, and "science":110. It's often more convenient to use a precision of a rule instead of counts. The precision is a percentage of ratio between category document count and the total count. The example above will have this precision representation:

obama_T 1000 {"business":12,"**politics**":99,"society":99,"science":11}

The category precision is a probability of that a document matched by a given rule pattern belongs to a particular category. For the **obama_T** rule: if a story mentions obama in the title, then it's certain that the story is political.

Types of Rules

The following rules are generated from every url and title

- **Domain Rules**: where by the pattern is just a domain of the url
- **Hosts Rules**: where the pattern is all the subdomains found in the the url
- **Path Rules**: where domain is paired which every pathname /chunk/
- **Title Rules**: generated from single terms and bigrams of tokenized page title
- **Url Rules**: generated from single terms and bigrams of tokenized url

These are the examples of each rule in precision form

URL term: archives_U 24487 {"personal finance":93,"personal finance/stocks":92}
URL bigram: business+wire_U 24035 {"business":98}
Title term: gas_T 23552 {"business":96,"business/energy":95}
Domain rule: sec.gov_D 23061 {"personal finance":100,"personal finance/stocks":100}
Host rule: espn.go.com_H 16951 {"sports":99}
Path rule: kansascity.com/sports_P 15603 {"sports":100}

Selecting Rules

Given millions of rules generated from large corpus, the task of selecting best subset of rules that fits into prescribed size is non-trivial. The choice of rules is limited by these factors:

- the client ruleSet should be rich enough to provide high recall/precision for in-browser categorization
- the client ruleSet should be small enough to comfortably exists across many browser configurations including low-memory ones.

There are two types of algorithms available for rule selection: brute force and information-gain selection.

Brute force rule selection

- order rules by total count
- select rules above precision threshold
- select preset number of rules

The obvious advantage of this approach is simplicity. These are the first few rules generated in brute force fashion for 90% precision threshold:

```
business_U 249633 {"business":94}
sports_U 153862 {"sports":99}
sport_U 94464 {"sports":100}
ebola_T 87797 {"science":99,"health & fitness":99}
quarter_T 82855 {"personal finance":96,"personal finance/stocks":94}
businessweek.com_D 74242 {"business":98}
results_T 72736 {"personal finance":90}
business-standard.com_D 71238 {"business":98}
digitaljournal.com_D 70363 {"business":96}
football_U 69965 {"sports":100}
entertainment_U 69442 {"arts & entertainment":99}
investing_U 66535 {"business":90}
politics_U 64015 {"politics":95,"society":100}
finance_U 58957 {"personal finance":98}
ajc.com_D 52631 {"business":94}
obama_T 52267 {"politics":99,"society":99}
nasdaq.com_D 49968 {"personal finance":100}
financial_T 46636 {"personal finance":99}
earnings_T 45268 {"personal finance":100,"personal finance/stocks":99}
bank_T 43904 {"personal finance":93}
```

The rules seem to be sound, and show good coverage and precision. In fact, the performance and size of 10,000 rules selected this way is acceptable.

The size of the resulting ruleset file is under 0.5M uncompressed and overall document-wide precision/recall is acceptable on folded tests (e.i. testing documents removed from training corpus). For a ruleSet generated on data published between Sep 20, 2014 and Nov 30, 2014, test results are below.

Date Range	Precision	Recall	Docs
12/01/2014-12/10/2014	92	90	521633
12/10/2014-12/20/2014	92	90	709711
12/20/2014-12/31/2014	91	90	457696
12/31/2014-01/10/2015	92	90	543491

Recall and precision do not change significantly in any of these intervals, and are sufficient for reliable user interest prediction from the news.

The problem with brute-force approach is that many rules are redundant. Consider the rules per category found in our ruleSet:

```
3005 "sports"
1733 "personal finance"
1672 "society"
1597 "business"
931 "technology & computing"
781 "arts & entertainment"
556 "personal finance/stocks"
538 "sports/soccer"
438 "politics"
410 "science"
....
```

Is it really necessary to carry 3000 sports rules. Perhaps we can get by with only 1000 sport rules or less? This way we could free up space for rules that classify less populous category outnumbered by redundant rules from well-populated categories. LIke the ones below:

```
  4 "sports/martial arts"
  4 "health & fitness/aids hiv"
  4 "arts & entertainment/radio"
  3 "personal finance/retirement planning"
  3 "food & drink/cooking"
  2 "society/dating"
  1 "technology & computing/unix"
  1 "sports/surfing"
  1 "sports/rowing"
  1 "sports/gymnastics"
  1 "sports/cycling"
  1 "science/physics"
  1 "pets"
  1 "food & drink/coffee"
```

To filter out redundant rules, we will employ a notion of information gain achieved by an individual rule.

Selection based on information gain.

Information Gain is defined as the difference between information entropy of the original system and the entropy of that system after a particular event occurred. The information entropy is expectation of negative log of probability. More formally, for a probability distribution X, define information entropy as $H(X) = E[-\log_2(P(X))]$.

Suppose a given document has a particular distribution of probabilities associated with categories it may be belong to. For example, a hypothetical document may have this distribution of probabilities: {“politics”: 0.5, “sports”: 0.3, “fashion”: 0.1, “unknown”: 0.1}.

The information entropy of such document (and its associated probability distribution) is:
$E[-\log(P(X))] = - 0.2 * \log_2(0.2) - 0.3 * \log_2(0.3) - 0.1 * \log_2(0.1) - 0.1 * \log_2(0.1) = 1.185$

Suppose that after a document is matched by a particular rule (R1) with respective probabilities equal to {“politics”: 0.9, “fashion”: 0.05, “unknown”: 0.05}

The uncertainty about categorization of the document is obviously reduced, as the information entropy of new categorization decreases:

$E[-\log(P(X))] = - 0.9 * \log_2(0.9) - 2 * 0.05 * \log_2(0.05) = 0.578$

The information gain (G1) introduced by the R1 is equal to entropy reduction caused by changing one categorial distribution to another:

$G1 = 1.185 - 0.578 = 0.607$

Now suppose that there’s yet another rule (R2), that makes precision even higher: { {“politics”: 0.95, “unknown”: 0.05}. Suppose that rule R2 is applied after R1 - what’s the information gain generated by R2 over the document already classified by R1:

$G2 = 0.578 - (0.95 * \log_2(0.95) + 0.05 * \log_2(0.05)) = 0.291$

The gain generated by the second rule is less than the first rule because the uncertainty reduction between R1 and R2 distributions is less than between the original (apriori) distribution and R1.

This gives an insight into how rule selection algorithm may proceed: at every step the algorithm chooses a rule that generates the largest information gain over the documents it matches. This way the documents already classified will not contribute (or contribute less) to uncertainty reduction than the documents tagged with categories which rules have not been seen yet. As a result, the algorithm will filter out rules that do not contribute to populous categories information gain (because they are superseded by previous rules), while categories for which rules are scarce will be given preferential treatment.

Formal Description

- Based on empirical counts of categorized documents compute a-priori distribution of categories across the corpus

- Assign a-priori distribution to each document in the corpus
- Iterate over potential ruleSet and
    - compute information gain for each rule in the set by summing information gain added by rule application to each document where the rule is applicable
    - choose a rule with the largest information gain
    - change the rule documents categorical distribution to reflect the chosen rule
    - repeat iteration

This algorithm implies that information gain is recomputed for each rule in the set on every iteration. Given thousands of rules and millions of documents - the task is clearly hard.

Algorithm Simplification

A simpler approach is to combine a brute force algorithm with information-gain algorithm. Recall that rules with precision are called topical rules and place a document into a category unconditionally.

If only the topical rules are considered, computing information gain is relatively easy. The information gain of a topical rule is the number of **new documents** it places into a category bucket. The **new documents** simply mean this: If rule R1 was applied before rule R2, then R2's information gain can not count the documents R1 already classified. Only the documents unclassifed by previously applied rules make the information gain of rules considered and each iteration.

The algorithm is formulated below:

- Iterate over ruleSet of topical rules
- compute information gain for each rule in by finding how many **new** documents it can categorise
- choose a rule with the largest information gain
- repeat iteration

Implementing this algorithm allowed for drastic reduction of ruleset size without noticeable loss of recall. We selected 5,000 most informative rules and compared to 10,000 rules performance:

performance metrics	10K rules		5K rules	
	precision	recall	precision	recall
over all test documents for all categories	91	90	91	89
over all test documents for top categories	90	90	91	89

over all test documents for sub categories	93	74	93	73
average per category	85.5	59.7	86	57.5
average per 500 most popular sites	91	89.3	91	87.9

Beyond Topical Rules - Bayesian model for in-browser interest classification.

We studied topical rules, where the probability of a category is very high. However, a number of rules are still highly informative but not topical:

newsweek.com_D 1002 {"science":17,"politics":38,"society":72,"military":13,"business":11,"health & fitness":13}
popsugar.com_D 888 {"fashion":15,"technology & computing":31,"arts & entertainment":50}
fox2now.com_D 794 {"law/crime":24,"sports":14,"politics":26,"law":25,"society":63}

We can not say with high confidence that a visitor to newsweek.com is intersted in society, but a visitor is definetly NOT interested in fashion or law. While fox2now.com visitor would have an entirely different collection of possibilities, and it's not business for sure. Perhaps, rules do not have to be topical to be useful. Especially if multiple rules are matched by the same page. Consider these three rules:

cbslocal.com_D 26611 {"law":11,"law/crime":11,
"sports/american football":15,
"society":37,
"arts & entertainment":16,
"sports":28}
film_T 12839 {"arts & entertainment":72, "business":13, "society":13}
actor_T 1859 {"society":22,"business":15,"arts & entertainment":68}

None of the rules is topical: rules' categories do not have high enough precision to claim topicality. However, if two of them happen together, the probability of "arts & entertainment" highly increases, hence a combined occurrences of multiple rules may disambiguate page enough to place it into a proper category. To say differently, co-occurrence on non-topical rules may find a topical page.

The question is how to combine multiple rules matching same page into probabilistic model. Let's assume each rule match could be viewed as independent evidence for a particular category, then the probability of a given category under multiple rule matching is given by a Bayes law. The law defines the probability of a category C provided an evidence R (a matching rule in our case) is seen:

$$P(C|R) = P(R|C) * P(C) / P(R)$$

This is a convenient notion as all the probabilities on right side are directly computable from the corpus statistics. For clarity, we should define few important notions:

R - is a rule
C - is a category
R documents - a set of documents that a particular rule **R** matches
C documents - a set of documents tagged with a category **C**
R of **C** documents - a set of documents matched by **R** and tagged by **C**
T - a set of ALL documents (full corpus)

Let's rewrite Bayes probabilities in terms of sizes of appropriate document sets:

Probability of matching **R** if document is tagged with **C** - P(**R|C**) = |**R of C**| / |**C**|
Probability that a document is tagged with **C** - P(**C**) = |**C**| / |**T**|
Probability that a document is matched by **R** - P(**R**) = |**R**| / |**T**|

Rewriting Bayesian expression in terms of doc sets sizes gives an interesting result:

$$P(C|R) = \frac{|R\ of\ C|}{|C|} \cdot \frac{|C|}{|T|} \div \frac{|R|}{|T|} = \frac{|R\ of\ C|}{|R|}$$

Which is exactly the precision of a single rule. Now suppose multiple rules are matched and the rules are independent in which case a probability of matching multiple rules is a product of their individual probabilities:

$$P(C|R1\ \&\ R2\ \&\ ...\ \&\ Rn) = P(R1\ \&\ R2\ \&\ ...\ \&\ Rn|C) \cdot P(C) \div P(R) = \prod_{i=1}^{n} P(R_i|C) \cdot P(C) \div \prod_{i=1}^{n} P(R_i)$$

Using doc sets ratio, we arrive to formula below:

$$P(C|R1\ \&\ R2\ \&\ ...\ \&\ Rn) = \prod_{i=1}^{n}(P(R_i|C) \div P(R_i)) \cdot P(C) = \prod_{i=1}^{n}(\frac{|R_i\ of\ C|}{|C|} \div \frac{|R_i|}{|T|}) \cdot \frac{|C|}{|T|} = \prod_{i=1}^{n} \frac{|R_i\ of\ C|}{|R_i|} \cdot \left(\frac{|T|}{|C|}\right)^{n-1}$$

$$P(C|R1\ \&\ R2\ \&\ ...\ \&\ Rn) = \prod_{i=1}^{n} \frac{|R_i\ of\ C|}{|R_i|} \cdot \left(\frac{1}{P(C)}\right)^{n-1}$$

This gives an algorithm to combine matching rules together:

1. given a page, match all the rules
2. for all matched rules multiple probabilities of identical categories in each rule.
3. if category is missing in any rule, set it's probability to 0
4. multiply product for each category C by $\left(\frac{1}{P(C)|}\right)^{n-1}$ where n is the number of rules
5. Choose category(s) with resulting probability above given threshold (90% for example)

Note that smaller categories become highly more probable as number of matched rules for that category increases, which seems a proper behavior - it's a lot harder to make rules from a small category to co-occur, hence a document has much higher chance to belong to it.

The formula implies that rules are independent, which is very incorrect. "Actor" and "film" have a lot bigger chance to co-occur than "actor" and "car". They are clearly correlated, yet the formula is still meaningful, and it is also empirically sound.

Let's work the example above, the corresponding a priori probabilities of interest categories are:

"law":3.4%,
"law/crime":2.8%,
"sports/american football":4.7%,
"society":27.9%,
"arts & entertainment": 9%,
"sports": 20.1%
"business": 32.7%

Suppose rules **cbslocal.com_D** and **film_T** both match same page, the categorical probabilities computed using bayesian formula are:

P("society") = 0.13 * 0.37 * (1/0.279) = 0.172 = 17%
P("arts & entertainment") = 0.72 * 0.16 * (1/0.09) = 1.28 = 128%

Wow! **The probability is above 1**! Recall, that we applied the bayesian law to independent events, which **cbslocal.com_D** and **film_T** are definitely not! Hence the formula breaks. But it breaks in the right way - it did select the correct "arts & entertainment" category and downgraded "society".

To make sure that the method works for populous categories, let's consider a new rule occurring along with **cbslocal.com_D** :

korea_T 7481 {"law":12,"business":30,"politics":31,"military":10,"personal finance":18,"society":53}

P("society") = 0.53 * 0.37 * (1/0.279) = 0.7 = 70%
P("law") = 0.12 * 0.11 * (1/0.034) = 0.38 = 38%

Again - the results are meaningful, as a story about Korea on sbslocal.com is more likely to be about society than the law.

Performance Testing

Rules were generated via brute-force method from Moreover news corpus for dates between 09/20/2014-11/30/2014. Four rulesets were tested:

- hostRules.dfr - domain, host and path rules
- titleTerms.dfr - title terms and bigram
- urlTerms.dfr - term and bigrams generated from URL
- combined ruleset

The testing data consisted of Moreover documents published between 12/01/2014 and 01/10/2015. The results are given below:

ruleset	docs	prec	recall
urlTerms.dfr	2265577	71	62
titleTerms.dfr	2265577	91	71
hostRules.dfr	2265577	95	58
combined	2265577	79	93

Beyond the overall results, few other metrics were computed, such as:

- average performance per category
- average performance for top sites
- performance for individual categories
- performance for individual sites

The full report is here

It's noticeable that urlTerms.dfr performs poorly compared to the rest of rule sets. Hence, another ruleset that combined titleTerms.dfr and hostRules.dfr was generated and tested. The results are here. The overall performance of this ruleset is 92% precision and 90% recall, which made us exclude urlTerms from the final version of the ruleset integrated with ID.

Scoping rules to genre

When we first integrated the host and title ruleset into ID, the result were surprisingly poor. One particular disturbance was highest rank of "**technology & computing**" in **any** user profile. What was going on?! The answer was that rules good for news are good for **only** for news. The applicability of news-generated rules does not extend to generic web behavior. Consider this title rule:

google_T 30862 {"business":28,**"technology & computing":99**,"society":14}

30862 evidences undeniably tells us that **google** in title means "**technology & computing**" - there are 30862 news articles published in 2 months that ALL are tagged with "**technology &**

computing”. However, when we apply this rule to a real user history, any google search gets classified into “**technology & computing**”, because it contains word **google** in it. Another striking example is:

manchester_T 11193 {"sports":94,"sports/soccer":93}

When used in news context Manchester almost always means Manchester United - a soccer team! Could we extend such rule to any url… doubtfully.

So, we have to scope news rules to news sites, and that’s exactly what was done with the __SCOPES keyword in rule processing engine. The title rules will only trigger if the domain or subdomain of the URL is found in the list of domains associated with a particular subsection of the ruleset. __SCOPES keyword allows limiting unbound rules (like title or url) to the sites that corresponds to genre of the corpus the rules were generated from.

Scoping news rules to news site resulted in significant increase in recall without noticeable loss in precision.

Code repositories

Interest Dashboard

- ID repo
- ruleSet
- rule matching implementation

Corpus collection and analysis repo

- full repo
- analytical and testing scripts
    - rule generator
    - ruleset tester
    - rule selector
    - mysql reports and database population scripts