

Building a Dynamic Classifier for Large Text Data Collections

Pavel Kalinov, Bela Stantic, Abdul Sattar
Institute for Integrated and Intelligent Systems (IIIS)
Griffith University, Brisbane, Australia
iiis.griffith.edu.au

21st Australasian Database Conference
January 18-22, 2010, Brisbane, Australia

Background and motivation

Users: Need to find information

- ▶ One information-finding need is **locating**:
we know it exists, need to find where it is.
- ▶ Another information-finding need is **discovery**:
we don't know it exists, we need to find out that it does.

Solutions: Search Engines and Web Directories

- ▶ A **Search Engine** is a machine for locating information.
It uses machine learning techniques and is automated.
- ▶ A **Web Directory** allows browsing (information discovery).
Even state-of-the art directories are maintained manually.

Search Engines and Web Directories

- ▶ A **Search Engine** presumes that the user knows what he wants, and is able to describe it in several words.
Reality: the user often has a very vague idea of what he wants, and cannot describe it with relevant keywords.
- ▶ A **Web Directory** is very expensive to maintain.
Even state-of-the art directories are on the decline. They are practically extinct.
- ▶ **Users** have a problem...
 - ▶ The information **locating** need is met by search engines.
 - ▶ The information **discovery** part is lacking.

Solution: Automated Web Directory

- ▶ Uses a spider such as the search engines use, to download billions of documents.
- ▶ Uses machine learning to classify them into a hierarchical structure.
- ▶ Overcomes the economic problems of manual labour currently employed.
- ▶ Satisfies the users' **information discovery** need.

State of the art

Why is this not done already?

- ▶ Momentum of old business model (Yahoo! Directory).
- ▶ No business model at all (DMOZ).
- ▶ No business at all due to competition from Google (most other directories).
- ▶ Major scientific effort going to search engines.

So... What's the actual problem?

Why don't you just use...

- ▶ Text classification is an old Machine Learning task.
- ▶ Many algorithms exist, and they work well.
- ▶ But... they work with static data only.

What is our data then?

- ▶ Dynamic as composition: documents added and removed constantly.
- ▶ Dynamic as content: document texts change very often.
- ▶ Dynamic as classification: document labels change often.

Classification task

What do we need to adapt?

- ▶ Detach document collection management from the indexing/classification part.
- ▶ Make the system incremental, so changes in the document collection do not necessitate full retraining.
- ▶ Make the system robust in the face of changed labeling.
- ▶ Keep classifying indefinitely! As opposed to the standard algorithms, our work is never done.

What we did to cope with the above?

- ▶ We have a web spider constantly updating the document collection. Classification is an independent process over a frozen snapshot of the data.
- ▶ We use a Multinomial Naïve Bayes (MNB) classifier which deals easily with small changes in document texts.
- ▶ We have a separate classifier for each branched node (current prototype has 3 levels, 474 classifiers, 6368 classes).

MNB: Good enough?

To classify a document using “maximum likelihood”, we find:

$$\operatorname{argmax}_c p(C = c) = \ln \frac{p(C|D)}{p(\neg C|D)} = \ln \frac{p(C)}{p(\neg C)} + \sum_i \ln \frac{p(w_i|C)}{p(w_i|\neg C)}$$

(the likelihood \mathbf{p} that document \mathbf{D} belongs to class \mathbf{C} is the sum of the log likelihood of the class itself and the sum of log likelihoods of all words \mathbf{w} in the document to belong to the class; find the biggest likelihood - you have found the **winner class**)

Naïve: What does the algorithm assume?

Independent feature model: word occurrences in a document are independent of each other (violates Zipf’s Law).

An implicit assumption is also that labels are correct.

When normalizing (e.g. - TF/IDF), we also assume:

- ▶ Similar length (word counts) of documents across classes.
- ▶ Similar variety (unique word counts) of documents across classes.
- ▶ Similar number of documents in each class.

MNB: Not good enough...

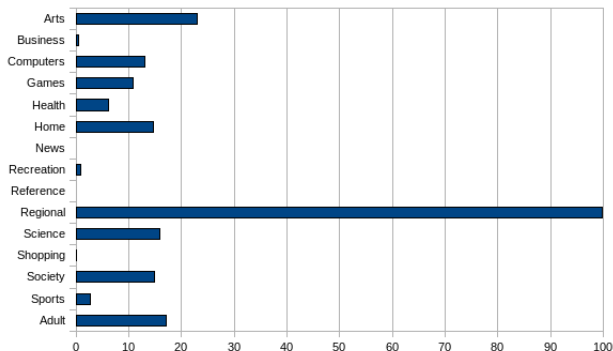
We find all assumptions broken:

- ▶ Zipf's Law **does** work - features are not independent.
- ▶ Document lengths vary greatly between classes, harming normalization.
- ▶ Unique word counts per document vary greatly between classes (235 words in *News* : 96 in *Business*).
- ▶ Document counts per class are drastically different (1.1 mln instances in *Regional* : 9 thousand in *News*).
- ▶ Very high labeling (wrong classification) and other noise due to *spam* and *web decay* (in the order of 15%).

Result from assumptions broken

Algorithm performance:

- ▶ We achieve only 47.91% accuracy in classification.
- ▶ It puts almost everything in the dominant class.
- ▶ The dominant class has 99.87% accuracy. Some others have 0.00%.
- ▶ Accuracy deviation between classes is 0.24.



MNB: Mitigation techniques

Word count normalization, proposed by Frank and Bouckaert:
(Naïve Bayes for Text Classification with Unbalanced Classes,

2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases)

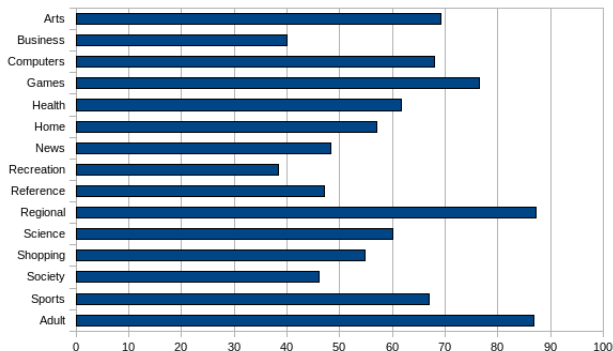
$$n'_{wd} = \alpha \times \frac{n_w d}{\sum_{w'} \sum_{d \in D_c} n_{w' d}}$$

where $n_{w' d}$ are class-specific word counts (occurrences of the word in documents of class c), replacing w_i in *Eq.1* and $\alpha = 1$.

Result from normalization

Results are still unbalanced:

- ▶ Overall accuracy goes from 47.91% to 69.44%.
- ▶ The algorithm is very good in the dominant class (*Regional*): 87.29%.
- ▶ Still very bad in some others: only 39.92% in *Business*.
- ▶ Overall deviation in accuracy: 0.15.



Why is this a problem?

Why we still need improvement:

- ▶ The subjective impact on users is much stronger than the objective statistical error (i.e. - if the classifier doesn't work correctly in **my** favourite category, I think it's totally useless).
- ▶ In a hierarchical structure, error propagates downwards and is multiplied at each level.
40% accuracy at the top level means 6.4 % only two levels down - we'd better just assign instances randomly...

What to do?

Take some ideas from spam filters:

- ▶ **Train-on-error** policy: spam filters only learn from instances where they make an error.
- ▶ This manipulates training: the filter sees a subset of the data; word count statistics and prior distributions are skewed.
- ▶ It may be unprincipled, but works better than **Train-on-everything**.

But still:

- ▶ Just as the classic MNB (with or without improvements), it's still static.
- ▶ Learns slowly, never fully unlearns.

Our approach

We made the system truly dynamic:

- ▶ We classify continuously.
- ▶ We apply weight decay after each pass - guaranteed unlearning.
- ▶ Some of our static measures (prior distribution) are calculated dynamically - over the last 10 000 instances.
- ▶ We **Train-on-error** - so it is actually last 10 000 errors.

Our contribution

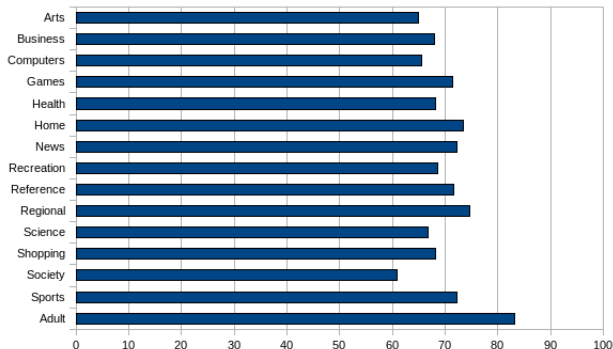
We have introduced a negative feedback loop:

- ▶ A class where the classifier has problems gets over-represented.
- ▶ Its prior distributions gets an artificial boost.
- ▶ In subsequent classification it's more probable to put instances into it.
- ▶ We have compensated for the problem, and we don't even need to know what it is.
- ▶ We have introduced stochastic learning into a previously static algorithm.

Results

Much better:

- ▶ Overall accuracy increased from 47.91% (69.44%) to 70.44%.
- ▶ More importantly: balanced results. Worst class results in *Society*: 60.82%, best in *Adult*: 83.22%.
- ▶ Overall deviation in accuracy is down **three times**: 0.05.



Conclusions

We have improvement:

- ▶ The classifier now learns from its mistakes:
we get better results with consecutive passes.
- ▶ It minimizes error variation between classes:
it is equally reliable in all classes.
- ▶ Faster in training:
it trains on a subset of the data, so is 4.13 times faster.

Limitations and issues still to be addressed

- ▶ Word count normalization is in the divisor of the equation: cannot apply the technique to it since we get a positive feedback loop.
- ▶ Accuracy improvement cannot continue indefinitely: when we achieve an even error distribution, improvement stops.
- ▶ 1.46 times slower in classification due to dynamic computations overhead (can be improved with caching).
- ▶ Errors are still high due to noisy data.
- ▶ We still need manual labour, and always will.

Questions

- ▶ Questions?