Using naive Bayes method for learning object identification

Giedrius BALBIERIS
Computer Networking Department, Kaunas University of Technology
Studentų 50-416, LT-3031 Kaunas, Lithuania
e-mail: balbgied@pit.ktu.lt

Abstract

After establishment of Internet based learning, during last 10 years a great variety of e-learning courses had been created. They involve many different technologies ready-made platforms and resemble a chaotic mass. XML based e-learning standards provide answers for this situation and solve most important tasks of content interoperability and reuse. In this paper authors overview those issues and propose a naive Bayes based solution for identification of learning objects within a course. Authors review main steps of algorithm and summarize results of executed experiments.

Computer aided learning (CAL) is around for couple decades. During this time lots of e-learning courses incorporating valuable training pieces, learning tools were created. Those courses use best-breed products and technologies of that time, but indeed that does not mean interoperability and reuse. Fortunately current e-learning standards like IMS and SCORM [1, 2] provide ways to stop reinventing bicycle over and over again.

Standards

They achieve this by standardizing e-learning to a common denominator - data structures and processes. One of the core facets of this panacea is structured content design, which when wrapped into metadata becomes a set of well-known learning objects. It is not the aim of this paper to describe what are the learning standards, objects and benefits. For such information please read [3, 4, 1, 2]. Indeed there is a variety of terms used to address learning objects. From our point of view we see four levels of objectivity:

1. Learning component. Structured content design is not a novelty indeed. Breaking down of content into small pieces adding interaction, personalizing the content after evaluation etc – these are well known and ever used educational content design principles. Thus learning components had existed from the beginnings of CBL, but they had no means for interoperability with other learning components and no possibility of reuse in other learning environments.
2. Learning object. When learning component is annotated with metadata in a standard way it becomes a learning object. Such learning object is ready to

interoperate and be reused.

3. Shared learning object. After establishing learning object repository and placing learning object into it, indexing it, providing means to preview, buy, download and deploy it, then it becomes a shared learning object.
4. Network learning object. Provided we have a shared learning object, but time goes, there are new versions of them published, but already deployed into course they are prone to grow old from the beginning. In case if there is a kind of GRID built on top of learning object repository, providing functions of secure tunels, certificate and load balancing, trust schemes etc; then learning object need not to be duplicated locally – it may be deployed for direct access over GRID network services.

Current e-learning is yet in objectivity phase one, but there is huge development in this area and most importantly technological premises are already in place for all types of implementation. Moving forward with metadata description is not an easy step. Provided institutions use one or another e-learning platform such as WebCT, Luvit, Lotus Learning Space or Blackboard, creating learning objects becomes a big-dig – all of current environments so far support only course level granularity and there are no objective premises even to separate learning components one from each other. Because of that learning objectivity is a two sided issue:

1. Learning component identification,
2. Description of learning component with metadata and learning object creation.

In this paper we primarily focus on first question. Description with metadata is achieved by using retrieved information (built-in technical metadata and generated keywords). Thus second question completely depends on solution of first and on technical implementation. By this paper we lay the grounds for possible automation and reducing the workload for future learning object metadata librarians.

Method premises

Courses consist of diverse data contained in various types of files and databases. As most e-learning courses use web published documents, others use compiled interactive components built with any possible tools. Yet others use files having proprietory formats which some-times are outside-locked, encrypted or password protected (such as some Acrobat PDF files). Advocating for open standards we base the success of this method on convertibility of courses into XHTML and open access for keyword scanning.

Next we elaborate on mathematical method for learning component identification and describe implementation algorithm used in our experiments.

Naive Bayes method for learning object identification

Provided we have course packed as a XHTML compatible file structure, we define:

$$\text{FILES} = \{ f_1, f_2, f_3, ..., f_n \} \qquad (1)$$

We define WORDS (2), which represents a set of words retrieved from FILES:

$$\text{WORDS} = \{ w_1, w_2, w_3, ..., w_m \} \qquad (2)$$

Considering the different frequencies of unique words contained in FILES, we denote:

$$\text{STATS}(f,w) = \{ s_1, s_2, s_3, ..., s_k \} = , \qquad (3)$$

where $k = n * m$.

In fact STATS is a square matrix where rows are file characteristic vectors also known in data mining and information retrieval sciences as inverse document frequency (IDF) vector [14]. Columns in (4) are weighted word frequency distribution across all course files.

$$STATS_{nm} = \begin{bmatrix} s_{11} & s_{21} & s_{31} & \cdots & s_{n1} \\ s_{12} & s_{22} & s_{32} & \cdots & s_{n2} \\ s_{13} & s_{23} & s_{33} & \cdots & s_{n3} \\ \cdots & \cdots & \cdots & s_{ij} & \cdots \\ s_{1m} & s_{2m} & s_{3m} & \cdots & s_{nm} \end{bmatrix} \qquad (4)$$

Also we define normalized distribution NSTATS of words in files:

$$\text{NSTATS}(f, w) = \{ ns_1, ns_2, ns_3, ..., ns_k \}, \qquad (5)$$

where $k= n * m$ and each $ns_i = s_i / Max_j$, here $i=1,k$.

Similarly to STATS NSTATS is also a square matrix, but has normalized (< 1) weights in all of ns_{ij} values (6).

$$NSTATS_{nm} = \begin{bmatrix} ns_{11} & ns_{21} & ns_{31} & \dots & ns_{n1} \\ ns_{12} & ns_{22} & ns_{32} & \dots & ns_{n2} \\ ns_{13} & ns_{23} & ns_{33} & \dots & ns_{n3} \\ \dots & \dots & \dots & ns_{ij} & \dots \\ ns_{1m} & ns_{2m} & ns_{3m} & \dots & ns_{nm} \end{bmatrix} \quad (6)$$

Here Max_j may be calculated to different values which are to be chosen empirically from several options:

1. $Max_j = const\ maxs = max(s_i)$, $i=1,k$, (7)
   Here Max_j is maximal frequency of some word across all files.
2. $Max_j = max_f = max(s_f)$, $s_f = STATS(f,*)$, (8)
   Here Max_j is maximal frequency of some word in currently normalized file.
3. $Max_j = max_w = max(s_w)$, $s_w = STATS(*,w)$. (9)
   Here Max_j is maximal frequency of particular normalized word across all files.

Based on which variant of calculation of Max_j we emphasize: equal weighting (7) – we used it in our experiments, file despite size equal weighting (8) or word despite occurrence number equal weighting (9).

Several other criteria maybe borrowed from large collection data mining and information retrieval systems [5] such as collection frequency coefficient f representing selective power of specific word (10) or cosine normalization coefficient c compensating for short file weighting on formating information clusters (11).

$$f_w = \log (N / n_w), \quad (10)$$

where f_w - selective power coefficient for word w, N – total number of documents in course and n_w is number of documents containing particular word w.

$$c_f = \frac{1}{\sqrt{\sum_{file\ f} s_i^2}} \quad (11)$$

where c_f – cosine coefficient for file f, s_i – weight of particular term in file f.

In generic information retrieval tasks there are huge data loads and due to that several are simplifications made; this decreases processing speed and needed memory size.

Usually huge collections [6] drop frequency characteristics and just use binary word in file presence matrix. This is quite different in much smaller e-learning courses. The novelty of mining data in smaller targeted data collections such as e-learning courses is in:

1. using already built in metadata for weighting and increasing selective power of meaningful words.
2. using format-adaptive varying term weighting.

Because of that in our method we introduced XHTML tag based weighting where words get multiplied by coefficient t depending within which place in the document they are located (for example t(title)=10, t(Heading 1)=5 etc) and using built-in meta tags for metadata generation.

Having normalized STATS into NSTATS we elaborate on calculating word cross relationships (12):

$$\mathrm{WORDREL}(i,j) = \mathrm{NSTATS}(*,w_i)\ \mathrm{NSTATS}(*.w_j)^T, \qquad (12)$$

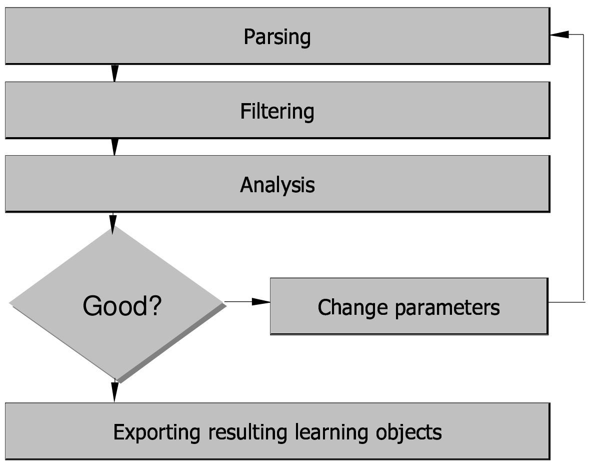
where $i \neq j$; here direct multiplication across files is used. This is done based on naive Bayes principle simplifying that there is no probabilistic dependency between words i.e. $P(w|f) = \prod_i P(w_i|f)$. This makes calculation of relationships possible to $n * m^2$ time.

Provided that we have calculated relationships between words in course, now we are able to cluster NSTATS information into a number of clusters (this could be called topic discovery). This is achieved by sorting the relationship matrix WORDREL and removing all relationships lower than a defined threshold value and for each word leaving only two most relative words (a number of experiments with more edges between words did not show significant difference for clusterization task [8, 10, 12]). Least significant relationships are removed until satisfactory granularity is achieved:

```
function Clusterize(WORDREL, const minClusters)
   while (numClusters(WORDREL) <= minClusters)
      deleteMinRelationship(sort(WORDREL))
   end # while
end function
```

Provided we have a satisfactory number of information clusters we calculate closest neighbor cluster for each file and produce the stand-alone learning objects. Note: since e-learning contents is usually organized from educational modules - information + activities + evaluation [13], the wording of each pedagogic parts usually differ a lot. Because of that automatic learning object identification system tends to group files according to contents and possibly would group different phases of one pedagogic module into separate learning objects. In order to correct this erroneous behaviour few other parameters should be considered: grouping similarity coefficient for files in same directory and analysis of cross linking provided it is present.

Algorithm



Picture 1 Learning object identification algorithm

The main principles of algorithm (Pic.1) for learning object identification are following:

1. Files are scanned recursively in all course directories :
   a) Each document is converted into common denominator (XHTML),
   b) Each file is scanned and words are indexed into IDF vector. Words are weighted upon chosen weighting formula (7-9). Stop-words are removed.
2. Statistical parameters for the course are calculated.
3. Based on statistical parameters some words are designated as errors, others – as stop words.
4. Filtering is used to define clusterization sensitive words.
5. Clusterization is applied to the index of words. During this process relationships

between words are calculated, clusters of words (thematic topics) are formated. If the number of clusters is too small, weakest links between words are broken. If the number of clusters is too big, merging of clusters based on nearest neighbor is used.

6. Classification of files in clusters is done. This is achieved by calculating nearest neighbor cluster to the file IDF vector.
7. Resulting learning objects are exported producing standard required metadata descriptors. Otherwise parsing, filtering, clusterization and classification parameters are modified and process is repeated from the beginning.

Experiment

At Computer Network dept., Kaunas University of Technology above mentioned algorithm was implemented using Perl programming language and browser based course upload, analysis and reporting engine.

Despite of algorithm optimization introduced in the method, all cycle of learning object identification (course import, statistical analysis, normalization, calculation of relationships and clusterization, classification) used to take around one hour. We used courses consisting of average 200 documents.

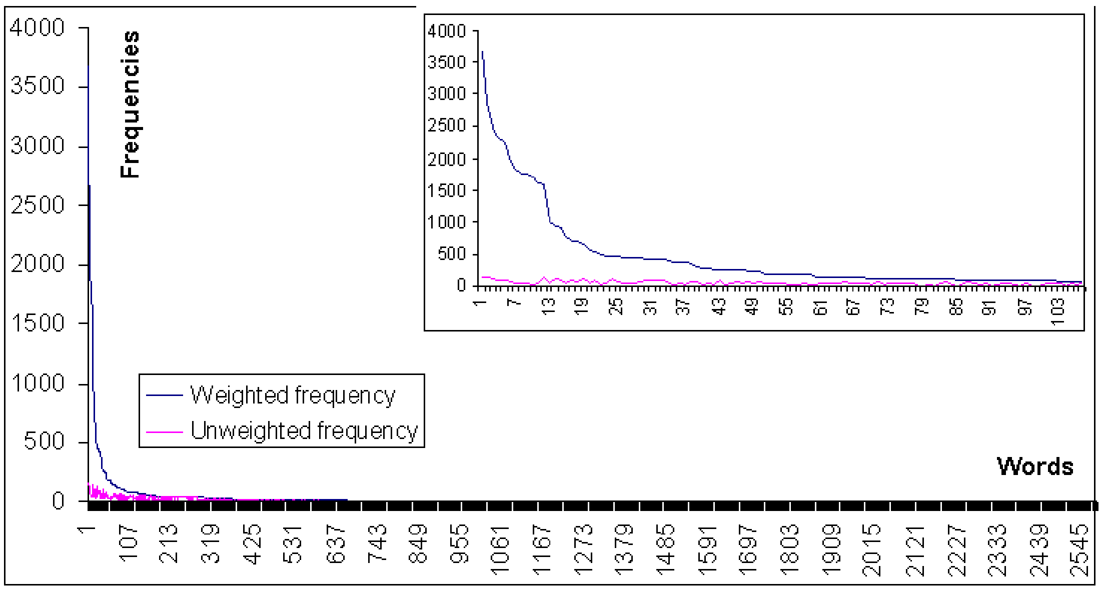
For stop words we used existing stop word collections:

1. Berkeley University collection:
   http://sunsite2.berkeley.edu:8000/html/help/stopwords.html
2. RAPIC – US Dept. of Energy collection:
   http://www.em.doe.gov/rapic/66stop.html
3. EPNet collection:
   http://bll.epnet.com/help/ehost/Stop_Words.htm

Experiment has shown that some words tend to form small bipolar, tripolar clusters, which need to be dumped and minimum file per cluster threshold need to be assigned.

For example course 'Multimedia in Education' had 776 files, 190 document files, 17712 words, 3222 unique words, 8281 stop words. Most files already contained metadata providing means for more precise classification of words: 194 titles, 27 headers, 0 table headers, in different runs 10-20 learning objects were identified roughly 10 documents per object.

For calculating WORDREL we used empirical constants maxFreq = 1000 and minFreq = 30 (Pic.2). By these numbers we defined selective window removing potential stop-words (maxFreq), clusterization-insensitive words (minFreq) and reducing time required for word cross relationship calculation. By that we achieved reduction to 1000 words. In that case total calculation time was 200 files x 1000 words x 1000 words i.e. 200 million iterations and at first runs it took 45 minutes on Pentium 4 2.4 Ghz Linux machine with 512MB RAM. After optimizing to 100 millions clusterization used to take ~20 minutes.



Picture 2 Word frequencies in course 'Multimedia in Education'

As we see in picture 2 there is a big difference comparing weighted and not weighted words. This shows the core benefit of using term weighting in smaller data collections as e-learning object identification. Term weighting helped improving not only in formating selective window, but also for metadata formation afterwards.

In current stage method provided good learning object identification suggestive level, although some assistance in tuning the identification parameters is still needed.

Summary and future steps

In this paper we presented the core principles of applying naive Bayes method for learning component identification. Indeed there is still space to grow.

We used yet another constant - directory structuring threshold. This approach helped

solving multidimensionality of learning information and is aplicable only to courses where content is structured into directories. Nevertheless in courses where all content is placed in a single directory hyper-link and cross-link analysis needs to be used. This would help exploring learning process sequencing, identifying learning components, using this information in clustering and classification processes. Cross link analysis would help solving not only single directory issue, but also would aid in identification of sub-learning objects on tinier granularity levels.
Also there are more possibilities to investigate and choose a best normalization formula, find how it depends on course size, file size variation or possibly involve Neural networks for choosing it.
Next improvement could be in using Neural Nets for automatic calculation of selective window and subsequent better clusterization and algorithm performance.

References

1. IMS Standard, Internet: http://www.imsproject.org (checked 5/25/03)
2. SCORM Standard, Internet: http://www.adlnet.org (checked 5/25/03)
3. Longmire, W., A Primer on Learning Objects, 2000, available on Internet: http://www.learningcircuits.org/mar2000/primer.html (checked 5/25/03)
4. Willey, D., The Instructional use of Learning Objects, 2000, available on Internet: http://www.reusability.org/read/ (checked 5/25/03)
5. Salton, G., Buckley, Ch., Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management 24(5): 513-523 (1988)
6. Strzalkowski, T., Robust Text Processing in Automated Information Retrieval, in K. Sparck-Jones and P. Willet (eds.) Readings in Information Retrieval, Morgan Kaufman Publishers, pp. 317-322, 1997
7. S.K. Pal, V. Talwar and P. Mitra, Web Mining in Soft Computing Framework : Relevance, State of the Art and Future Directions", IEEE Trans. Neural Networks, vol 13, no. 5, pp. 1163-1177. (2002)
8. Sahami, M., Learning Limited Dependence Bayesian Classifiers. In KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 335-338, Menlo Park, CA: AAAI Press. (1996)
9. Goldszmidt, M., and Sahami, M., A Probabilistic Approach to Full-Text

Document Clustering. Technical Report ITAD-433-MS-98-044, SRI International. (1998)

10. Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E., A Bayesian Approach to Filtering Junk E-Mail. In Learning for Text Categorization: Papers from the 1998 Workshop. AAAI Technical Report WS-98-05. (1998)
11. Dominich, S., Mathematical/Formal Methods in Information Retrieval. Technology Letters (special edition), vol. 4, number 1, United Kingdom. Proceedings of the ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval ACM SIGIR MF/IR 2000, Athens, Greece. (2000)
12. Sahami, M., Using Machine Learning to Improve Information Access. PhD Thesis, Stanford University, Computer Science Department. STAN-CS-TR-98-1615. (1999)
13. Healey, M. & Jenkins, A. Kolb' s Experiential Learning Theory and Its Application in Geography in Higher Education, Journal of Geography, 99, pp.185-195 (2000)
14. Lucarella, D., A search strategy for large document bases, Electronic Publishing, vol. 1(2), 105-116. (1988)