A Simulator for Concept Detector Output

Robin Aly and Djoerd Hiemstra
Database Group
Computer Science Faculty
University Twente
P.O. Box 217
7500AE Enschede
The Netherlands

November 20, 2009

Abstract

Concept based video retrieval is a promising search paradigm because it is fully automated and it investigates the fine grained content of a video, which is normally not captured by human annotations. Concepts are captured by so-called concept detectors. However, since these detectors do not yet show a sufficient performance, the evaluation of retrieval systems, which are built on top of the detector output, is difficult. In this report we describe a software package which generates simulated detector output for a specified performance level. Afterwards, this output can be used to execute a search run and ultimately to evaluate the performance of the proposed retrieval method, which is normally done through comparison to a baseline. The probabilistic model of the detectors are two Gaussians, one for the positive and one for the negative class. Thus, the parameters for the simulation are the two means and deviations plus the prior probability of the concept in the dataset.

Classification

H.3.3 [Information Storage and Retrieval]: Query formulation

Keywords

Video Information Retrieval, Detector Simulation

Software Location

`http://detectsim.sourceforge.net`

1 Introduction

Concept based video retrieval currently enjoys a lot of attention in the multimedia community [14]. The retrieval process consist now of two steps: (1) The concept detection step in which a computer program, the concept detector, tries to detect predefined concepts in video shots and (2) the retrieval step, which uses the output of step (1) is used to answer a specific information need. However, the detection of concepts is still far from being perfect and varies significantly between concepts and datasets, and therefore is the evaluation of methods proposed for step (2) difficult. In order to overcome this problem we propose in [1] a method to simulate the output of a concept detector to be able to generate output of a specified performance. Using this method, novel retrieval methods can be evaluated and compared at several performance levels. This report introduces a software package which generates such output to simplify the evaluation process for other researchers.

The probabilistic model which is used for the simulation, consists of two Gaussians, one for the positive and one for the negative class. The simulation of detector outputs of a concept lexicon is specified with the mean and deviations of each class and the prior of each concept. Based on concept annotations, a simulation run generates for each concept in each shot a confidence score, which would be the output of a support vector machine (SVM) [16] in reality. This score can then be outputted directly or be converted into a binary classification or a posterior probability [11]. In order to get a feeling for the quality of the detector output the mean average precision is reported, following the evaluation methodology from TRECVID [12]. On this data set, one or more search methods can be executed and any performance measure could be used. However, since a single simulation run could show accidentally too good or too bad performance the results have to be repeated multiple times, until the average of the search performance measure does not change significantly anymore. This procedure can be repeated with different model parameters to assess the search performance at different performance levels.

Although we see in the provided software a good tool to evaluate concept detector based retrieval methods without a real detector set we also want to emphasize that we do not encourage the exclusive evaluation of retrieval methods with this software. For example, the presented simulator outputs very "clean" scores and omits several influential factors, which will play a role in reality, such as the dependence between detector results within a shot or between adjacent shots. Therefore, the search results should always only be used together with the evaluation on a real dataset.

This report proceeds as follows: Section 2 gives a brief overview of the detector model. This is the a slightly shortened version of Section 3 of the work this report is build upon [1]. The data accompanying the simulator is described in Section 3. Section 4 describes how the simulator is invoked and how scientist should cite this work. Section 5 concludes this report.

2 Detector Simulation

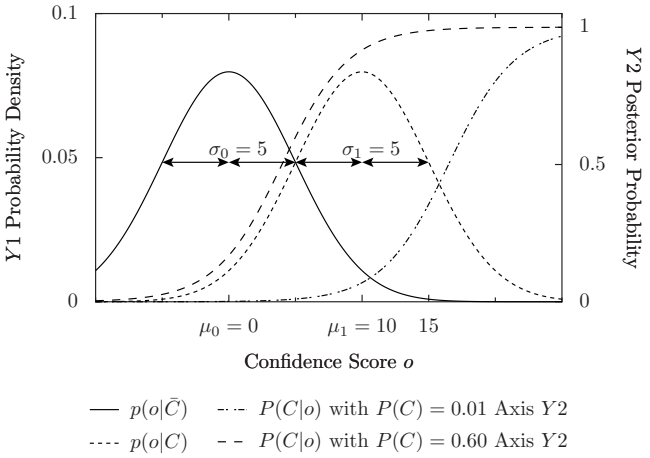

Figure 1: Detector Model

2.1 Detector Model

In this section we describe our probabilistic model of a detector, which is later used for the randomization of confidence scores. We define a probabilistic model of a detector set: We assume that the confidence scores of different detectors for a single shot are independent from each other and that they are normally distributed in the positive and negative class. Each concept C has a different prior probability $P(C)$. To simplify the model, we assume that all concepts share the same mean μ_1 and standard deviation σ_1 for the positive and μ_0 and σ_0 for the negative class respectively. Note, that this assumption is strong and certainly does not hold in reality. However, because we focus here on the principle behavior of the detectors we leave the exploration of a more realistic model, which investigates different parameter settings for each detector, to future work. Also, while the investigation of different means and deviations is an important aspect for building detectors, we argue that the intersection of the areas under the probability density curves has a much higher influence on the performance than the absolute ranges of the confidence scores. Clearly, the smaller the area of the intersection the better the detector is. Our model can adequately simulate this effect by either moving the means apart or by varying the deviation of the classes.

Figure 1 shows the model of a single detector. We also plot the posterior probability of observing the concept given the confidence score using two different priors, one of $P(C) = 0.01$ and one for $P(C) = 0.50$. Considering a confidence score of 15 the posterior probability for a concept with the prior of

0.50 is close to certainty while for a concept with a prior of 0.01 it is undecided (50%) - with all other parameters equal. Therefore, our model does not have the limitation that all detectors have the same performance as in [7].

2.2 Posterior Probability

As noted by Platt [11], the assumption of two Gaussians for the negative and positive class can lead to unwanted effects, namely a non-monotonic posterior probability function. To prevent this effect we use an improved version of the algorithm from Platt [11], suggested by Lin et al. [9], to fit a sigmoid function to the confidence scores. The sigmoid function is defined by the two parameters A and B:

$$P(C|o) = \frac{1}{1 + \exp\left(A\, o + B\right)}$$

2.3 Simulation Process

In this section we discuss the actual simulation process which is defined in Algorithm 1. The algorithm uses an annotated collection (which carries labels for each concept in each shot where 0 means concept is not present and 1 that it occurs). The input parameters of the algorithm are the means μ_0, μ_1 and standard deviations σ_0, σ_1 of the positive and negative class and the number of training examples to fit the posterior function. A Gaussian distribution with mean μ and standard deviation σ is denoted as $N(\mu, \sigma)$.

From the annotated collection we calculate the prior probability $P(C)$ of the dataset. We then generate confidence scores for the positive and negative class using the prior probability and S samples. Now we use the algorithm described by Lin et al. [9] to fit the sigmoid posterior probability function to the generated samples. After the determination of the sigmoid parameters we iterate over all shots in the annotated collection. For each shot we determine for each concept in the lexicon whether it occurs and draw a random confidence score o from the corresponding normal distribution. Afterwards, we calculate the posterior probability of this concept in the shot using the sigmoid function with the previously determined parameters A_C and B_C. For combination methods which use binary classifications we assume a positive occurrence if the posterior probability is above 0.5. This is justified by decision theory, see for example [5].

After the randomization, we determine the detector performance MAP of the detector output ($DMAP_i$). We then execute a search run for each combination method using the randomized collection. Afterwards, we evaluate the resulting ranking using relevance judgments to obtain the search MAP ($SMAP_i$) for this run. This process is repeated N times to rule out random effects and the results are averaged.

3 Data

As mentioned above, the simulation process requires complete truth annotations of all concept in the lexicon for all shots. Furthermore, to evaluate the search performance, relevance judgments for a query set are needed. The concept annotations and relevance judgments included in this software package are shown in Table 1. Note, we try to be consistent with the abbreviations in the table.

Collection (col)	Concept Annotations (cset)	Relevance Judgments (qset)
tv05d	lsc (446) [10] mm101c (101) [15]	tv05q (24) [17]
tv07d	tv070809c (65) [4] + bw (1) [13]	tv07q (24) [this paper]

Table 1: Data Annotations: Short Name (count) [Source]

Here tv05d stands for the TRECVID 2005 development collection (43907 shots), consisting of News Broad Casts, and tv07d for the TRECVID 2007 development collection (18120 shots), with general Dutch television. On the tv05d corpus following annotations are included: The LSCOM (lsc) [10] concept annotations consist of 446 concepts of which 374 concepts where selected by the Vireo team [8] (vireo374c). Furthermore, Mediamill published concept annotations of 101 concepts (mm101c) for this data set [13]. For the TRECVID 2007 dataset, which was released by Netherlands Institute for Sound And Vision, collaborative annotation efforts [4] in 2007, 2008 and 2009 resulted in the annotation of 65 concepts (tv070809c). Additionally, the Mediamill group kindly provided the ground truth for a "black and white" concept [13], which is useful for some of the queries. However, the Mediamill group points out that only positive examples where annotated and that not judged samples could still contain the concept. After investigating the not annotated data we decided that the annotations are good enough and we include them as a fully annotated set into the dataset. We call the concept lexicon consisting of the collaborative concepts and the black and white concept tv070809bwc.

The relevance judgments for the official TRECVID 2005 queries (tv05q) on tv05d were kindly provided by Rong Yan [17], formally at CMU. While testing the judgments for consistency we found that few shots were not judged correctly (especially for the query "0169 one or more tanks or other military vehicles"). We performed this check by using the annotations for the concept *Tank* from mm101c to validate the judgments. The relevance judgments for the official TRECVID 2007 queries (tv07q) on tv07d were created by us. We used the tv070809c concept annotations as "perfect detectors" and used our retrieval model [3] to retrieve results. The initial weights were set according to [2]. We then judged 100 shots for each query and updated the weights. We repeated this until around 1300 shots were judged for each query. For some queries, we stopped early since it was hardly possible to find another results (for example, if there are no more shots annotated with the concept kitchen, when looking for relevant shots for the query "0219 Find shots that contain the Cook character in the Klokhuis series"). Given the percentage of judged

data and used methodology, we suggest that the quality of the judgments is high enough, to use it in the simulation experiments. In the distribution all the above data is in the "data/" subdirectory.

Note, because of the simplified handling we modified the shot identifiers in both datasets so that they always follow the pattern "shot0000_000". This was mainly done to simplify sorting. However, the software package also works with other identifier formats and the original identifiers can be restored by following perl command:

```
perl -wpe 's/shot(\d+)_(\d+)/sprintf("shot%d_%d",$1,$2)/eg' \
     *mtx  *.qrel
```

4 Usage

We now describe how the a detector score set is generated.

4.1 Requirements

The simulation tool is written in Java 1.6. Therefore, this runtime environment is needed. For the generation of random numbers we employed the (pseudo) random number generator [6], which we include in the distribution.

4.2 Data Generation

The generation of data is performed by invoking the simulator with following arguments:

```
java -Xmx800M -esa -ea -jar  detectsim-v1.0.jar \
-s data/col-tv07d-cset-tv070809bwc.truth-mtx \
-cs data/col-tv07d-cset-tv070809bwc.truth-stat \
-cf data/cset-tv070809bwc.schema \
-doPlatt 2000 \
-seed 100000 \
-mean1 2 \
-sigma1 1 \
-prefix out/col-tv07d-cset-tv070809bwc \
-N 8 \
-rsfF Posterior,Score,Classification
```

Here $N = 8$ detector sets based on the tv07d collection are created. The parameters have following meaning. Note, this command can be simply cut and pasted in a unix-like shell. However, we are not aware of a method how to enter such long commands in windows shell.

-s The ground truth annotation in a matrix form: "shot-identifier Annotation($\{0, 1\}$)+

-cs The collection statistics of the above dataset, which is used to obtain the concept priors.

-cf The "physical schema" of the dataset. That is, what concept are available and in which column (1 based) are they found in the file specified under -s.

-doPlatt How many samples to use to fit the posterior probability function (see Equation)

-seed The initial seed to use with the random generator. This seed can be used to regenerate dataset and therefore make results comparable to other publications.

-mean1 The mean of the positive class (the mean of the negative class is set to 0.

-sigma1 The standard deviation of the positive class (default: 1).

-prefix What prefix to prepend to the file name (otherwise only the specified parameters are used as a filename).

-N How many datasets should be used.

-rsfF What scores should be produced (Available formats are: **Posterior**, the posterior probability; **Score**, the raw score; **Classification**, the binary classification decision $\{0, 1\}$.

Seed To be able for other scientist to reproduce the reported results the data should be generated with a specified seed which should get mentioned with the evaluation results. Important: all datasets for the given parameters should be created with one call to the software. Multiple calls (one for each dataset) with the same seed would create the exact same dataset multiple times.

4.3 Output

The above command will emit a set of files all staring with:

```
col-tv07d-cset-tv070809bwc-dset-sim-version-1.0- \
 seed-100000-m0-0.00-s0-1.00-m1-2.00-s1-1.00-p-2000-Ni-000
```

The filename consists of name-value pairs. Most of the the components have been already introduced or are self explanatory, except Ni which is the dataset number. Furthermore, files with following extension are created:

concepteval This file contains the evaluation of the concept (using the data from .truth-mtx as ground truth). The content looks as follows:

```
# Concept   AP      No   TP    TN     FP    FN
airplane    0.17421 33   0     18087  0     33
...
MAP         0.28209 -    26049 1143116 6702  20053
```

Here, the concept *airplane* has a average precision of 0.17421 there were 33 positives taken into account (only interesting for the case of $No > 2000$, see [1]. The rest are statistics for the case that the detector was used as a classifier: Truepositives, True Negatives, False Positives and False Negatives. The last line contains the Mean Average Precision and a sum of the counts above. This mean average precision should be used (as an average) to display the expected mean average precision.

plattparameters In this file the parameters from the sigmoid fitting are stored. Normally these should not be needed.

priorestimations This file contains estimates from the generated data about the prior of the concepts, defined as $P(C) = \frac{\{d|C(d)=1}{N}$, thus the number of documents containing the concept divided by the number of all documents (N). However, since we do not know the data, we estimate the expected prior as follows:

$$E[P(C)] = \frac{\sum_d P(C|o_d)}{N}$$

This estimate is used for example in following retrieval models from Zheng et al. [18] and us [3].

{prob,score,classification}-mtx These files contain the actual dataset in the specified format (raw simulated score, posterior probability, classification).

For example, the shorted first line of a prob-mtx file could look as follows:

```
shot0001_001 0.00018 0.00069 0.00023 0.02563 0.00717 0.00663
```

To interpret these numbers the corresponding .schema file (Parameter -cs above) is needed. For example, the first line in cset-tv070809c.schema is:

```
001 airplane
```

Therefore, the posterior probability of having an airplane in shot0001_001 is 0.00018.

4.4 Sequential Generation

The generation of datasets can also be automated, for example in a shell command:

```
for mu in 1 2 3 4; do \
java -Xmx800M -esa -ea -jar  detectsim-v1.0.jar \
-s data/col-tv07d-cset-tv070809bwc.truth-mtx \
-cs data/col-tv07d-cset-tv070809bwc.truth-stat \
-cf data/cset-tv070809bwc.schema \
-doPlatt 2000 \
-seed 100000 \
-mean1 $mu \
-sigma1 1 \
-prefix out/col-tv07d \
-N 8 \
-rsfF Posterior,Score,Classification; \
done
```

4.5 Search Runs

After the generation of the datasets the search run(s) should be performed, resulting in the specified (parameter -N) number of rankings. We recommend to name the rankings similar to the dataset (for example prepending the model parameters of the retrieval model). Afterwards, the search mean average precisions can be aggregated by using the accompanying script:

```
averageMAP.pl col-tv07d-qset-tv07d-by-UT.qrel map *.trec > results
```

This tool assumes trec_eval is installed.

4.6 Reporting Results

When using this software we suggest a wording like:

> We investigated the performance of our retrieval method with a simulated detector set, described in \cite{Aly2009}. For all runs we used a seed of X and N samples of the collection.

with following bibtex entry.

```
@INPROCEEDINGS{Aly2009,
  author = {Aly, Robin and Hiemstra, Djoerd},
  title = {Concept detectors: how good is good enough?},
  booktitle = {MM '09: Proceedings of the seventeen
               ACM international conference on Multimedia},
  year = {2009},
  pages = {233--242},
```

```
  address = {New York, NY, USA},
  publisher = {ACM},
  doi = {http://doi.acm.org/10.1145/1631272.1631306},
  isbn = {978-1-60558-608-3},
  location = {Beijing, China},
}
```

5 Conclusions

In this report we described a simulator software which generates detector output, resembling SVM detectors, with specified performance parameters. After giving a brief introduction into the theory (which is explained in full in [1]) the usage of the software was detailed.

References

[1] Robin Aly and Djoerd Hiemstra. Concept detectors: how good is good enough? In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 233–242, New York, NY, USA, 2009. ACM.

[2] Robin Aly, Djoerd Hiemstra, and Arjen P. de Vries. Reusing annotation labor for concept selection. In *CIVR '09: Proceedings of the International Conference on Content-Based Image and Video Retrieval 2009*. ACM, 2009.

[3] Robin Aly, Djoerd Hiemstra, Arjen P. de Vries, and Franciska de Jong. A probabilistic ranking framework using unobservable binary events for video search. In *CIVR '08: Proceedings of the International Conference on Content-Based Image and Video Retrieval 2008*, pages 349–358, 2008.

[4] Stéphane Ayache and Georges Quénot. Video corpus annotation using active learning. In *30h European Conference on Information Retrieval (ECIR'08)*, pages 187–198, March 30 2008.

[5] J. Bather. *Decision Theory. An Introduction to Dynamic Programming and Sequential Decisions*. Wiley-Interscience Series in Systems and Optimisation. John Wiley and Sons, West Sussex, England, 2000.

[6] Michael Thomas Flanagan. Psrandom class: Generation of random deviates. Appeared on Website, 11 2009. `http://www.ee.ucl.ac.uk/~mflanaga/java/PsRandom.html`.

[7] A. Hauptmann, Rong Yan, Wei-Hao Lin, M. Christel, and H. Wactlar. Can high-level concepts fill the semantic gap in video retrieval? a case study with broadcast news. In *IEEE Transactions on Multimedia*, volume 9-5, pages 958–966, August 2007.

[8] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 494–501, New York, NY, USA, 2007. ACM.

[9] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276, October 2007.

[10] M. Naphade, J.R. Smith, J. Tesic, Shih-Fu Chang, W. Hsu, L. Kennedy, A. Hauptmann, and J. Curtis. Large-scale concept ontology for multimedia. *IEEE MultiMedia*, 13(3):86–91, 2006.

[11] J. Platt. *Advances in Large Margin Classifiers*, chapter Probabilistic outputs for support vector machines and comparison to regularized likelihood methods, pages 61–74. MIT Press, Cambridge, MA, 2000.

[12] A.F. Smeaton, P. Over, and W. Kraaij. High level feature detection from video in TRECVid: a 5-year retrospective of achievements. In Ajay Divakaran, editor, *Multimedia Content Analysis, Theory and Applications*. Springer, 2008.

[13] Cees G. M. Snoek, Jan C. van Gemert, Theo Gevers, Bouke Huurnink, Dennis C. Koelma, Michiel van Liempt, Ork de Rooij, Koen E. A. van de Sande, Frank J. Seinstra, Arnold W. M. Smeulders, Andrew H. C. Thean, Cor J. Veenman, and Marcel Worring. The mediamill trecvid 2007 semantic video search engine. In *Proceedings of the 7th TRECVID Workshop*, Gaithersburg, USA, October 2007.

[14] Cees G. M. Snoek and Marcel Worring. Concept-based video retrieval. *Foundations and Trends in Information Retrieval*, 4(2):215–322, 2009.

[15] Cees G. M. Snoek, Marcel Worring, Jan C. van Gemert, Jan-Mark Geusebroek, and Arnold W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 421–430, New York, NY, USA, 2006. ACM Press.

[16] V. N. Vapik. *Learning Theory: Inference from Small Samples.* Wiley, 1998.

[17] Rong Yan. *Probabilistic Models for Combining Diverse Knowledge Sources in Multimedia Retrieval*. PhD thesis, Canegie Mellon University, 2006.

[18] Wujie Zheng, Jianmin Li, Zhangzhang Si, Fuzong Lin, and Bo Zhang. Using high-level semantic features in video retrieval. In *Image and Video Retrieval*, volume Volume 4071/2006, pages 370–379. Springer Berlin / Heidelberg, 2006.

Data: Annotated Collection **C**, Lexicon **L**
Input: $N, S, \mu_0, \sigma_0, \mu_1, \sigma_1$
Result: Randomized collection

// Randomize Prior Estimate
foreach *Concept C in Lexicon* ***L*** **do**
  Calculate $P(C)$ from annotations in **C**
  generate $\lceil SP(C) \rceil$ positive samples from $N(\mu_1, \sigma_1)$
  generate $S - \lceil SP(C) \rceil$ negative samples $N(\mu_0, \sigma_0)$
  determine A_C and B_C according to [9]
end
// Randomize Detection Output
for *Repetition* $i \in [1..N]$ **do**
  foreach *Shot s in Collection* ***C*** **do**
    foreach *Concept C in Lexicon* ***L*** **do**
      if *C occurs in s* **then**
        draw o from $N(\mu_1, \sigma_1)$
      else
        draw o from $N(\mu_0, \sigma_0)$
      end
      // Calculate Posterior according [11]
      $P(C|o) = \frac{1}{1+exp(A_C o + B_C)}$
      // Transform to Binary Value
      if $P(C|o) > 0.5$ **then**
        $C = 1$
      else
        $C = 0$
      end
    end
  end
  Calculate Detector Performance $DMAP_i$
  Run Search with Combination Methods
  Calculate Search Performance $SMAP_i$
end
Report $DMAP = \frac{\sum_i DMAP_i}{N}, SMAP = \frac{\sum_i SMAP_i}{N}$

Algorithm 1: Algorithm for a Simulation Run. N: Number of Repetitions, S: Sample size for Sigmoid fitting, $\mu_0, \sigma_0, \mu_1, \sigma_1$: Model parameters