*Experiment Design for Data Science* – WS 2019/20
**Exercise 2: Reproduce experimental results from a paper**

Deadline for submission: Thu, January 30th, 2020, 23:59

---

For this assignment, you will work in **groups of three** and **reproduce** the experimental setup, experiments, and results as explained in a scientific paper. The task of interest is – based on careful experimental design – to confirm the numbers and findings reported in the paper or, alternatively, uncover inconsistencies and therefore challenge the conclusions drawn.

Specifically, by reproducing the experiments, it can be checked whether

- the information given in the paper is sufficient to reproduce the results reported
- statistically significant differences can be made out between the different settings, in particular when this is not reported in the paper (no significance tests, confidence intervals, p values, etc. or not even variance)
- the values reported in the paper could stem from the distribution sampled by the reproduced experiments

Your efforts need to be documented in a report that should be accompanied by the code and workflow you used to reproduce the experiments. To do so, first, identify the experimental setup and strategy taken in the paper and outline the steps necessary for you to reproduce results. Try to stay to the described steps as close as possible, i.e., use the same implementations and settings wherever possible. If the exact implementation is unknown or unavailable, find a reasonable substitute. You might have to consult additional sources to get the full picture of the used data or methods.

In each step, document which information is given in the paper and which measures you took to implement this step, i.e., which implementation and which parameters you used. Report the numbers obtained in each intermediate step. Identify deviations from the numbers reported in the original paper, their origin, and estimate whether they will have a significant impact on subsequent steps.

For the results, perform adequate tests to test for statistical significance. Justify your choice. Can you confirm the findings of the paper? Did you identify a flaw in their setup? Could you correct the flaw? How did it impact the results and findings of the experiments?

You need to produce the **results of one scientific paper out of a choice of three**. More details are found following this task description. The corresponding papers are also attached to this document.

**For writing the report, follow the ACM formatting guidelines, using the templates provided at** https://www.acm.org/publications/proceedings-template. (Proceedings Style File: LaTeX2e - Strict Adherence to SIGS style; LaTeX recommended, but Word/OpenOffice is also ok). **Report page limit: Maximum 6 pages!** Focus on the key aspects!

In class on January 9/16, 2020, each group will have to give a short **presentation on the status of the work** (Strict time limit: 5 minutes!). To this end, prepare a deck of max. 5 slides (including first slide containing group number, members, and chosen option), presenting your strategies, encountered difficulties and key findings, as well as your conclusion about the experimental design, intermediate results, conclusions and remaining work and **submit** these as PDF until January 8/15, 2020, resp., no later than 4pm. Registration for the presentation time slots will be made available on TUWEL.

The final report should detail the same aspects: strategies, difficulties, key findings, as conclusion about the experimental design, results, and conclusions.

**Submit** your report as PDF together with any code, workflow, configurations, etc. you used, compressed in a ZIP archive via TUWEL.

Grading scheme:

| | |
|---|---|
| • Inclusion of title, names, and registration numbers: | max. 5% |
| • Quality/clarity of report and presentation: | max. 10% |
| • Reproduction of results: | max. 60% |
| • Description, interpretation, statistical testing: | max. 25% |

**Option 1: Predicting the suitability of movies for an inflight viewing context**

Based on the paper "TUD-MMC at MediaEval 2016: Context of Experience task" by Wang and Liem, your task is to reproduce their results on individual classifiers and combinations of classifiers (tables 1, 2, and 3).
The dataset referred to in the paper can be downloaded from https://www.dropbox.com/sh/j7nunecnzfjrp2r/AAC1BAf5JEv-rGUW9h02L2X2a?dl=0

Note that for this option, the main workload originates from feature selection, combination, and classifier configuration.

**Option 2: Prediction of music genre across different taxonomies**

Based on the paper "MediaEval 2018 AcousticBrainz Genre Task: A Baseline Combining Deep Feature Embeddings Across Datasets" by Oramas et al., your task is to reproduce their results on predicting genres from different taxonomies using stacked neural networks. Reproduce the results from both Table 1 and 2. (Note: the restricted part on AllMusic data will be shared within the class.)

Note that for this option, the main workload originates from additional feature extraction, classifier building, and model training. Some prior experience with neural networks is advised.

**Option 3: Classification of news and medical texts**

Based on the paper "Text Categorization with Support Vector Machines: Learning with Many Relevant Features" by Joachims, your task is to reproduce the results on predicting categories of texts using different classifiers and in particular SVMs with different parameter settings. For the Reuters set, reproduce the results in Fig. 2, for Ohsumed the results mentioned in the text.

Note that for this option, the main workload originates from consulting additional sources, feature preparation, experiment setup, and classification optimization.

# TUD-MMC at MediaEval 2016: Context of Experience task

Bo Wang
Delft University of Technology
Delft, The Netherlands
b.wang-6@student.tudelft.nl

Cynthia C. S. Liem
Delft University of Technology
Delft, The Netherlands
C.C.S.Liem@tudelft.nl

## ABSTRACT

This paper provides a three-step framework to predict user assessment of the suitability of movies for an inflight viewing context. For this, we employed classifier stacking strategies. First of all, using the different modalities of training data, twenty-one classifiers were trained together with a feature selection algorithm. Final predictions were then obtained by applying three classifier stacking strategies. Our results reveal that different stacking strategies lead to different evaluation results. A considerable improvement can be found for the F1-score when using the label stacking strategy.

## 1. INTRODUCTION

A substantial amount of research has been conducted in recommender systems that focus on user preference prediction. Here, taking contextual information into account can have significant positive impact on the performance of recommender systems [1].

The MediaEval *Context of Experience* task focuses on a specific type of context: the viewing context of the user. The challenge considers predicting the multimedia content that users find most fitting to watch in a specific viewing condition, more specifically, while being on a plane.

## 2. DATASET DESCRIPTION AND INITIAL EXPERIMENTS

The dataset for the *Context of Experience* (CoE) task[5] contains metadata and pre-extracted features for 318 movies [6]. Features are multimodal and include textual features, visual features and audio features. The training set contains 95 labeled movies, which are labeled as 0 (bad for airplane) or 1 (good for airplane).

A set of initial experiments has been conducted in order to evaluate the usefulness of the various modalities in the *CoE* dataset [6]. A rule-based PART classifier was employed to evaluate the feature performance in terms of Precision, Recall and F1 Score, the result can be found in Table 1.

| Features used | Precision | Recall | F1 |
|---|---|---|---|
| User rating | 0.371 | 0.609 | 0.461 |
| Visual | 0.447 | 0.476 | 0.458 |
| Metadata | 0.524 | 0.516 | 0.519 |
| Metadata + user rating | 0.581 | 0.6 | 0.583 |
| Metadata + visual | 0.584 | 0.6 | 0.586 |

**Table 1: Results obtained by applying a rule-based PART classifier to the *Right Inflight* dataset.**

## 3. MULTIMODAL CLASSIFIER STACKING

Ensemble learning uses a combination of different classifiers, usually getting a much better generalization ability. This particularly is the case for *weak learners*, which can be defined as learning algorithms that perform just slightly better than random guessing by themselves, but can be jointly grouped into an algorithm with arbitrarily high accuracy [2].

Therefore, we were interested in taking a multimodal classifier stacking approach to the given problem, and use a combination of multiple weak learners to 'boost' them into a strong learner.

The process can be separated into three stages: classifier selection, feature selection and classifier stacking.

### 3.1 Classifier Selection

First of all , we want to select base classifiers that will be useful candidates in a stacking approach. For this, we use the following classifier selection procedure:

1. Initialize a list of candidate classifiers. For each modality, we consider the following classifiers: k-nearest neighbor, nearest mean, decision tree, logistic regression, SVM, bagging, random forest, AdaBoost, gradient boosting, and naive Bayes. We do not apply parameter tuning, but take the default parameter values as offered by scikit-learn[1].

2. Perform 10-fold cross-validation on the classifiers. As input data, we use the training data set and its ground truth labels, per single modality. For the audio MFCC features, we set NaN values to 0, and calculate the average of each MFCC coefficient over all frames.

3. If Precision and Recall and F1-Score > 0.5, keep the candidate classifier on the given modality as base classifier for our stacking approach.

The selected base classifiers and their relevant modalities can be found in Table 2. It should be noted that the performance of `Bagging` and `Random forest` is not stable. This is because Bagging tries to use different subset of instances in each run and RandomForest tries to use different subsets of instances and features in each run.

### 3.2 Feature Selection

For each classifier and corresponding modality, a better-performing subspace of features may optimize results further. Since we have

*MediaEval 2016 Workshop,* Oct. 20-21, 2016, Hilversum, Netherlands.

[1] http://scikit-learn.org/

| Classifier | Modality | Precision | Recall | F1 |
|---|---|---|---|---|
| k-Nearest neighbor | metadata | 0.607 | 0.654 | 0.630 |
| Nearest mean classifier | metadata | 0.603 | 0.579 | 0.591 |
| Decision tree | metadata | 0.538 | 0.591 | 0.563 |
| Logistic regression | metadata | 0.548 | 0.609 | 0.578 |
| SVM (Gaussian Kernel) | metadata | 0.501 | 0.672 | 0.574 |
| Bagging | metadata | 0.604 | 0.662 | 0.631 |
| Random Forest | metadata | 0.559 | 0.593 | 0.576 |
| AdaBoost | metadata | 0.511 | 0.563 | 0.536 |
| Gradient Boosting Tree | metadata | 0.544 | 0.596 | 0.569 |
| Naive Bayes | textual | 0.545 | 0.987 | 0.702 |
| k-Nearest neighbor | textual | 0.549 | 0.844 | 0.666 |
| SVM (Gaussian Kernel) | textual | 0.547 | 1.000 | 0.707 |
| k-Nearest neighbor | visual | 0.582 | 0.636 | 0.608 |
| Decision tree | visual | 0.521 | 0.550 | 0.535 |
| Logistic regression | visual | 0.616 | 0.600 | 0.608 |
| SVM (Gaussian Kernel) | visual | 0.511 | 0.670 | 0.580 |
| Random Forest | visual | 0.614 | 0.664 | 0.638 |
| AdaBoost | visual | 0.601 | 0.717 | 0.654 |
| Gradient Boosting Tree | visual | 0.561 | 0.616 | 0.587 |
| Logistic Regression | audio | 0.507 | 0.597 | 0.546 |
| Gradient Boosting Tree | audio | 0.560 | 0.617 | 0.587 |

**Table 2: Base classifier performance on multimodal dataset.**

multiple learners, we employed the *Las Vegas Wrapper*(LVW) [3] feature selection algorithm for a feature subset selection. For each run, LVW generate a list of random features and evaluate the learner's error rate for $n$ times, and select the best performing feature subspace as output.

In our case, we slightly modified LVW to optimize F1 score, where the original las vegas wrapper was developed for optimize accuracy.

For each base classifier, with the exception of the random forest classifier (as it already performs feature selection), we apply the LVW method, and achieve performance measures as listed in Table 2.

## 3.3 Classifier Stacking

In previous research, classifier stacking (or metalearning) has been proved beneficial for predictive performance by combining different learning systems which each have different inductive bias (e.g. representation, search heuristics, search space) [4]. By combining separately learned concepts, meta-learning is expected to derive a higher-level learned model that more accurately can predict than any of the individual learners. In our work, we consider three types of stacking strategies:

1. *Majority Voting:* this is the simplest case, where we select classifiers and feature subspaces through the steps above, and assign final predicted labels through majority voting on the labels of the 21 classifiers.

2. *Label Stacking:* Assume we have $n$ instances and $T$ base classifiers, then we can generate an $n$ by $T$ matrix consisting of predictions (labels) given by each classifier. Label combining strategy tries to build a second-level classifier based on this label matrix, and return a final prediction result for that.

3. *Label-Feature Stacking:* Similar to label stacking, label-feature stacking strategy uses both base-classifier predictions and features as training data to predict output.

# 4. RESULTS

We considered all prediction results by the 21 selected base classifiers, and then applied the three different classifier stacking strategies to the test data using 10-fold cross-validation. As results for label stacking vs. label attribute stacking were comparable on the training data, we only consider voting vs. label stacking on the test data.

All obtained results, on the training (development) and test dataset, are given in Table 3. On the training data, we notice significant improvement can be found in terms of Precision, Recall as well as F1 score in comparison to results obtained on individual modalities. The voting strategy results in the best precision score, but has bad performance in terms of recall. On the contrary, label stacking has higher recall and the highest F1 score.

Considering results obtained on the test dataset, we can conclude that *label stacking* is more robust than the voting strategy. For voting strategy, a significant decrease can be found in terms of precision on test set. This is because majority vote (and Bayesian averaging) tendency to over-fit derives from the likelihood's exponential sensitivity to random fluctuations in the sample, and increases with the number of models considered. Meanwhile, label stacking strategy performs reasonable well on test data.

| Stacking Strategy | Precision | Recall | F1 |
|---|---|---|---|
| Voting (cv) | 0.94 | 0.57 | 0.71 |
| Label Stacking (cv) | 0.72 | 0.86 | 0.78 |
| Label Attribute Stacking (cv) | 0.71 | 0.79 | 0.75 |
| Voting (test) | 0.62 | 0.80 | 0.70 |
| Label Stacking (test) | 0.62 | 0.90 | 0.73 |

**Table 3: Classifier Stacking results.**

# 5. CONCLUSIONS

In our entry for the MediaEval CoE task, we aimed to improve classifier performance by a combination of classifier selection, feature selection and classifier stacking. Results reveal that employing a ensemble approach can considerably increase the classification performance, and is suitable for treating the multimodal *Right Inflight* dataset.

The larger diversity of base classifiers is able to produce a more robust ensemble classifier. On the other hand, a blending of multiple classifiers may also have some drawbacks, e.g computational costs, and difficulty in traceable interpretation.

We expect better results for our method can still be obtained through parameter tuning, and by applying more robust classifier stacking methods, such as feature weighted linear stacking [7].

# 6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[2] Y. Freund and R. E. Schapire. A Decision-theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of computer and system sciences*, 55:119–139, 1997.

[3] H. Liu and R. Setiono. Feature selection and classification—a probabilistic wrapper approach. In *Proceedings of the 9th International Conference on Industrial and Engineering Applications of AI and ES*, pages 419–424, 1997.

[4] A. Prodromidis, P. Chan, and S. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In

*Advances in distributed and parallel knowledge discovery*, pages 81–114. MIT/AAAI Press, 2000.

[5] M. Riegler, , C. Spampinato, M. Larson, P. Halvorsen, and C. Griwodz. The mediaeval 2016 context of experience task: Recommending videos suiting a watching situation. In *Proceedings of the MediaEval 2016 Workshop*, 2016.

[6] M. Riegler, M. Larson, C. Spampinato, P. Halvorsen, M. Lux, J. Markussen, K. Pogorelov, C. Griwodz, and H. Stensland. Right inflight? A dataset for exploring the automatic prediction of movies suitable for a watching situation. In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 45:1–45:6. ACM, 2016.

[7] J. Sill, G. Takacs, L. Mackey, and D. Lin. Feature-weighted linear stacking. arXiv:0911.0460, 2009.

# MediaEval 2018 AcousticBrainz Genre Task: A baseline combining deep feature embeddings across datasets

Sergio Oramas[1], Dmitry Bogdanov[2], Alastair Porter[2]
[1]Pandora Media Inc., US
[2]Universitat Pompeu Fabra, Spain
soramas@pandora.com,dmitry.bogdanov@upf.edu,alastair.porter@upf.edu

## ABSTRACT

In this paper we present a baseline approach for the MediaEval 2018 AcousticBrainz Genre Task that takes advantage of stacking multiple feature embeddings learned on individual genre datasets by simple deep learning architectures. Although we employ basic neural networks, the combination of their deep feature embeddings provides a significant gain in performance compared to each individual network.

## 1 INTRODUCTION

This paper describes our baseline submission to the MediaEval 2018 AcousticBrainz Genre Task [1]. The goal of the task is to automatically classify music tracks by genres based on pre-computed audio content features provided by the organizers. Four different genre datasets coming from different annotation sources with different genre taxonomies are used in the challenge. For each dataset, training, validation, and testing splits are provided. This allows to build and evaluate classifier models for each genre dataset independently (Subtask 1) as well as explore combinations of genre sources in order to boost performance of the models (Subtask 2).

For this baseline, we decided to focus on demonstration of possibilities of merging different genre ground truth sources using a simple deep learning architecture. To this end, we explore how stacking deep feature embeddings obtained on different datasets can benefit genre recognition systems.

## 2 RELATED WORK

Submission to the previous edition of the task have explored late fusion of predictions made by classifier models trained for each genre source individually. In order to predict genres following a taxonomy of a target source, the proposed solutions applied genre mapping between taxonomies, either by computing genre co-occurrences on the intersection of all four training genre datasets [4], or by textual string matching [6].

In our baseline, we propose an alternative early fusion approach, similar to the one proposed in [7] for multimodal genre classification. The approach incorporates knowledge across datasets by stacking deep feature embeddings learned on each dataset individually and using those as an input to predict genres for each test dataset.

*MediaEval'18, 29-31 October 2018, Sophia Antipolis, France*

## 3 APPROACH

### 3.1 Input features

We use all available features extracted from music audio recordings using *Essentia* [2] and provided for the challenge. As a preprocessing step, we apply one-hot encoding for a few categorical features related to tonality (`key_key`, `key_scale`, `chords_key`, and `chords_scale`) and standardize all features (zero mean, unit variance). In total, this amounts to 2669 input features.

### 3.2 Neural network architecture

A simple feedforward network is used to predict the probabilities of each genre given a track. The network consists of an input layer of 2669 units (the size of the feature vector for an input recording), followed by a hidden dense layer of 256 units with ReLu activation, and the output layer where the number of units coincide with the number of genres to be predicted in each dataset. Dropout of 0.5 is applied after the input and the hidden layer. As the targeted genre classification task is multi-label, the output layer uses sigmoid activations and is evaluated with a binary cross-entropy loss.

Mini-batches of 32 items are randomly sampled from the training data to compute the gradient, and the Adam [3] optimizer is used to train the models, with the default suggested learning parameters. The networks are trained for a maximum of 100 epochs with early stopping. Once trained, we extract the 256-dimensional vectors from the hidden layer for the training, validation, and test sets.

The model architecture is used to train a multi-label genre classifier on each of the four datasets. The models are trained on 80% of the training set and validated after each epoch using the other 20% using the split script with release-group filtering provided by the organizers. Predictions are computed for the validation and test sets.

### 3.3 Embedding fusion approach

Following the described methodology, one model per dataset is trained and these models serve for predictions in Subtask 1. Then, the given models are used as feature extractors. All four models share the same input format, so input feature vectors from one dataset can be used as input to a model trained on other dataset. Thus, for each model we feed all tracks from the training, validation and test sets of each dataset, and obtain the activations of the hidden layer as a 256-dimensional feature embedding. Therefore, for each track in each dataset we obtain four different feature embeddings, coming from each of the four previously trained models.

Given the four feature embeddings of each track, we apply $l2$-norm to each of them and then stack them together into a single 1024-dimensional feature vector. Following this process, we obtain

**Table 1: ROC AUC on validation datasets**

| | AllMusic | Discogs | Lastfm | Tagtraum |
|---|---|---|---|---|
| Subtask 1 | 0.6476 | 0.7592 | 0.8276 | 0.8017 |
| Subtask 2 | 0.8122 | 0.8863 | 0.9064 | 0.8874 |

new feature vectors for every track in the training, validation and test sets of each dataset. Then, we use these feature vectors as input of a simple network where the input layer is directly connected to the output layer. Dropout of 0.5 is applied after the input layer. The output layer is exactly the same as in the network described in above, where sigmoid activation and binary cross-entropy loss are applied. The new network is trained following the same methodology described before, with Adam as the optimizer and mini-batches of 32 items randomly sampled. The network is trained on 80% of the training data and validated on the other 20%. Following this approach, we train a network per dataset, and obtain the genre probability predictions of the validation and test sets for Subtask 2.

### 3.4 Predictions thresholding

The predictions made by each model contain continuous values, while the task requires binary prediction of genre labels. We therefore apply a plug-in rule approach thresholding the prediction values in order to maximize the evaluation metrics. We decided to maximize the macro F-score, and applied thresholds individual for each genre label that we estimated on the validation data [5].

## 4 RESULTS AND ANALYSIS

We evaluated a single run for both Subtask 1 and 2. Table 1 presents the ROC AUC metric on the validation sets. Table 2 presents the final results after applying thresholding on the test datasets. As the general pattern, we can clearly see the benefit of models based on embedding fusion approach compared to the models trained individually on each dataset. While the individual models (Subtask 1) are hardly usable compared to the random and popularity baselines, the combined models got a significant improvement in performance, being competitive with last years' second ranked submission [6].

In our experiments, we focused on optimizing macro F-score, however choosing this metric for threshold optimization can have a negative effect on micro-averaged metrics. In the case of infrequent subgenre labels and an uninformative classifier, an optimal, but undesirable strategy may involve predicting those labels always [5]. Indeed, this was the case for the individual models, but the hybrid models did not have this issue.

## 5 DISCUSSION AND OUTLOOK

In our baseline approach we focused on Subtask 2 and demonstrated the advantage of fusing feature embeddings learned on individual genre datasets on the example of a simple feedforward network architecture. We may expect further improvements in performance by means of a more sophisticated network architecture (for example [4]). The code of the baseline is available online.[1]

**Table 2: Precision, recall and F-scores on test datasets**

| Average per | | Dataset | | | |
|---|---|---|---|---|---|
| | | AllMusic | Discogs | Lastfm | Tagtraum |
| Subtask 1 (individual models) | | | | | |
| Recording (all labels) | P | 0.0147 | 0.0591 | 0.0976 | 0.0992 |
| | R | 0.5753 | 0.5263 | 0.4512 | 0.5017 |
| | F | 0.0280 | 0.1035 | 0.1506 | 0.1623 |
| Recording (genres) | P | 0.2786 | 0.6305 | 0.3974 | 0.2991 |
| | R | 0.6960 | 0.7289 | 0.5966 | 0.6630 |
| | F | 0.3399 | 0.6382 | 0.4455 | 0.4040 |
| Recording (subgenres) | P | 0.0114 | 0.0256 | 0.0518 | 0.0636 |
| | R | 0.4861 | 0.3497 | 0.3295 | 0.4164 |
| | F | 0.0219 | 0.0467 | 0.0856 | 0.1083 |
| Label (all labels) | P | 0.0225 | 0.0744 | 0.0732 | 0.0951 |
| | R | 0.4943 | 0.2588 | 0.2330 | 0.2412 |
| | F | 0.0324 | 0.0935 | 0.0947 | 0.1141 |
| Label (genres) | P | 0.1658 | 0.3733 | 0.2321 | 0.2551 |
| | R | 0.4729 | 0.4229 | 0.3484 | 0.3573 |
| | F | 0.1938 | 0.3801 | 0.2546 | 0.2676 |
| Label (subgenres) | P | 0.0184 | 0.0595 | 0.0572 | 0.0764 |
| | R | 0.4949 | 0.2506 | 0.2213 | 0.2276 |
| | F | 0.0278 | 0.0792 | 0.0786 | 0.0962 |
| Subtask 2 (fusion models) | | | | | |
| Recording (all labels) | P | 0.1340 | 0.2775 | 0.2718 | 0.2972 |
| | R | 0.4809 | 0.5432 | 0.4762 | 0.5127 |
| | F | 0.1880 | 0.3320 | 0.3066 | 0.3451 |
| Recording (genres) | P | 0.5689 | 0.6877 | 0.5407 | 0.6061 |
| | R | 0.6905 | 0.7473 | 0.6335 | 0.6885 |
| | F | 0.5880 | 0.6845 | 0.5602 | 0.6243 |
| Recording (subgenres) | P | 0.0946 | 0.1472 | 0.1570 | 0.2022 |
| | R | 0.3251 | 0.3703 | 0.3368 | 0.4148 |
| | F | 0.1343 | 0.1892 | 0.1911 | 0.2465 |
| Label (all labels) | P | 0.0614 | 0.1087 | 0.1108 | 0.1235 |
| | R | 0.1640 | 0.2226 | 0.2168 | 0.2324 |
| | F | 0.0736 | 0.1247 | 0.1314 | 0.1400 |
| Label (genres) | P | 0.2907 | 0.4404 | 0.3077 | 0.2878 |
| | R | 0.3713 | 0.4713 | 0.3735 | 0.3565 |
| | F | 0.3080 | 0.4393 | 0.3246 | 0.3053 |
| Label (subgenres) | P | 0.0550 | 0.0922 | 0.0909 | 0.1043 |
| | R | 0.1582 | 0.2102 | 0.2009 | 0.2179 |
| | F | 0.0670 | 0.1089 | 0.1119 | 0.1206 |

## ACKNOWLEDGMENTS

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 688382 (AudioCommons) and 770376-2 (TROMPA), as well as the Ministry of Economy and Competitiveness of the Spanish Government (Reference: TIN2015-69935-P).

[1] https://github.com/MTG/acousticbrainz-mediaeval-baselines

## REFERENCES

[1] Dmitry Bogdanov, Alastair Porter, Julián Urbano, and Hendrik Schreiber. 2018. The MediaEval 2018 AcousticBrainz Genre Task: Content-based Music Genre Recognition from Multiple Sources. In *MediaEval 2018 Workshop*. Sophia Antipolis, France.

[2] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J.R. Zapata, and X. Serra. 2013. Essentia: An Audio Analysis Library for Music Information Retrieval. In *International Society for Music Information Retrieval (ISMIR'13) Conference*. Curitiba, Brazil, 493–498.

[3] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[4] Khaled Koutini, Alina Imenina, Matthias Dorfer, Alexander Rudolf Gruber, and Markus Schedl. 2017. MediaEval 2017 AcousticBrainz Genre Task: Multilayer Perceptron Approach. In *MediaEval 2017 Workshop*. Dublin, Ireland.

[5] Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. 2014. Optimal thresholding of classifiers to maximize F1 measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 225–239.

[6] Benjamin Murauer, Maximilian Mayerl, Michael Tschuggnall, Eva Zangerle, Martin Pichl, and GÃijnther Specht. 2017. Hierarchical Multilabel Classification and Voting for Genre Classification. In *MediaEval 2017 Workshop*. Dublin, Ireland.

[7] Sergio Oramas, Francesco Barbieri, Oriol Nieto, and Xavier Serra. 2018. Multimodal Deep Learning for Music Genre Classification. *Transactions of the International Society for Music Information Retrieval* 1, 1 (2018).

# Text Categorization with Support Vector Machines: Learning with Many Relevant Features

Thorsten Joachims

Universität Dortmund
Informatik LS8, Baroper Str. 301
44221 Dortmund, Germany

**Abstract.** This paper explores the use of Support Vector Machines (SVMs) for learning text classifiers from examples. It analyzes the particular properties of learning with text data and identifies why SVMs are appropriate for this task. Empirical results support the theoretical findings. SVMs achieve substantial improvements over the currently best performing methods and behave robustly over a variety of different learning tasks. Furthermore, they are fully automatic, eliminating the need for manual parameter tuning.

## 1 Introduction

With the rapid growth of online information, text categorization has become one of the key techniques for handling and organizing text data. Text categorization techniques are used to classify news stories, to find interesting information on the WWW, and to guide a user's search through hypertext. Since building text classifiers by hand is difficult and time-consuming, it is advantageous to learn classifiers from examples.

In this paper I will explore and identify the benefits of *Support Vector Machines (SVMs)* for text categorization. SVMs are a new learning method introduced by V. Vapnik et al. [9] [1]. They are well-founded in terms of computational learning theory and very open to theoretical understanding and analysis.

After reviewing the standard feature vector representation of text, I will identify the particular properties of text in this representation in section 4. I will argue that SVMs are very well suited for learning in this setting. The empirical results in section 5 will support this claim. Compared to state-of-the-art methods, SVMs show substantial performance gains. Moreover, in contrast to conventional text classification methods SVMs will prove to be very robust, eliminating the need for expensive parameter tuning.

## 2 Text Categorization

The goal of text categorization is the classification of documents into a fixed number of predefined categories. Each document can be in multiple, exactly one, or no category at all. Using machine learning, the objective is to learn classifiers from examples which perform the category assignments automatically. This is a supervised learning problem. Since categories may overlap, each category is treated as a separate binary classification problem.

The first step in text categorization is to transform documents, which typically are strings of characters, into a *representation suitable* for the learning algorithm and the classification task. Information Retrieval research suggests that word stems work well as representation units and that their ordering in a document is of minor importance for many tasks. This leads to an attribute-value representation of text. Each distinct word[1] $w_i$ corresponds to a feature, with the number of times word $w_i$ occurs in the document as its value. To avoid unnecessarily large feature vectors, words are considered as features only if they occur in the training data at least 3 times and if they are not "stop-words" (like "and", "or", etc.).

This representation scheme leads to very high-dimensional feature spaces containing 10000 dimensions and more. Many have noted the need for feature selection to make the use of conventional learning methods possible, to improve generalization accuracy, and to avoid "overfitting". Following the recommendation of [11], the *information gain* criterion will be used in this paper to select a subset of features.

Finally, from IR it is known that scaling the dimensions of the feature vector with their *inverse document frequency (IDF)* [8] improves performance. Here the "tfc" variant is used. To abstract from different document lengths, each document feature vector is normalized to unit length.

## 3 Support Vector Machines

Support vector machines are based on the *Structural Risk Minimization* principle [9] from computational learning theory. The idea of structural risk minimization is to find a hypothesis $h$ for which we can guarantee the lowest true error. The true error of $h$ is the probability that $h$ will make an error on an unseen and randomly selected test example. An upper bound can be used to connect the true error of a hypothesis $h$ with the error of $h$ on the training set and the complexity of $H$ (measured by VC-Dimension), the hypothesis space containing $h$ [9]. Support vector machines find the hypothesis $h$ which (approximately) minimizes this bound on the true error by effectively and efficiently controlling the VC-Dimension of $H$.

SVMs are very **universal learners**. In their basic form, SVMs learn linear threshold function. Nevertheless, by a simple "plug-in" of an appropriate kernel function, they can be used to learn polynomial classifiers, radial basic function (RBF) networks, and three-layer sigmoid neural nets.

One remarkable property of SVMs is that their ability to learn can be **independent of the dimensionality of the feature space.** SVMs measure the complexity of hypotheses based on the margin with which they separate the

[1] The terms "word" and "word stem" will be used synonymously in the following.

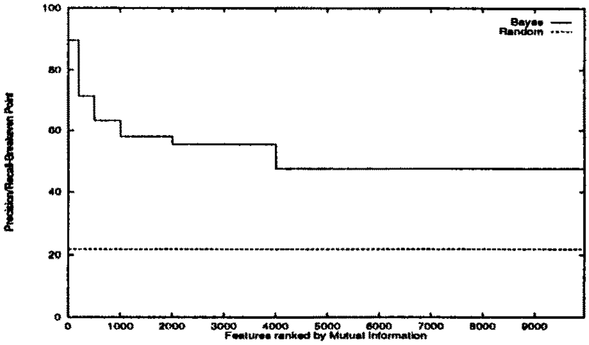

Fig. 1. Learning without using the "best" features.

data, not the number of features. This means that we can generalize even in the presence of very many features, if our data is separable with a wide margin using functions from the hypothesis space.

The same margin argument also suggest a heuristic for **selecting good parameter settings** for the learner (like the kernel width in an RBF network) [9]. The best parameter setting is the one which produces the hypothesis with the lowest VC-Dimension. This allows fully automatic parameter tuning without expensive cross-validation.

## 4 Why Should SVMs Work Well for Text Categorization?

To find out what methods are promising for learning text classifiers, we should find out more about the properties of text.

**High dimensional input space:** When learning text classifiers, one has to deal with very many (more than 10000) features. Since SVMs use overfitting protection, which does not necessarily depend on the number of features, they have the potential to handle these large feature spaces.

**Few irrelevant features:** One way to avoid these high dimensional input spaces is to assume that most of the features are irrelevant. Feature selection tries to determine these irrelevant features. Unfortunately, in text categorization there are only very few irrelevant features. Figure 1 shows the results of an experiment on the Reuters "acq" category (see section 5). All features are ranked according to their (binary) information gain. Then a naive Bayes classifier [2] is trained using only those features ranked 1-200, 201-500, 501-1000, 1001-2000, 2001-4000, 4001-9962. The results in figure 1 show that even features ranked lowest still contain considerable information and are somewhat relevant. A classifier using only those "worst" features has a performance much better than random. Since it seems unlikely that all those features are completely redundant, this leads to the conjecture that a good classifier should combine many features (learn a "dense" concept) and that aggressive feature selection may result in a loss of information.

**Document vectors are sparse:** For each document, the corresponding document vector contains only few entries which are not zero. Kivinen et al. [4] give both theoretical and empirical evidence for the mistake bound model that "additive" algorithms, which have a similar inductive bias like SVMs, are well suited for problems with dense concepts and sparse instances.

**Most text categorization problems are linearly separable:** All Ohsumed categories are linearly separable and so are many of the Reuters (see section 5) tasks. The idea of SVMs is to find such linear (or polynomial, RBF, etc.) separators.

These arguments give theoretical evidence that SVMs should perform well for text categorization.

## 5 Experiments

The following experiments compare the performance of SVMs using polynomial and RBF kernels with four conventional learning methods commonly used for text categorization. Each method represents a different machine learning approach: density estimation using a naive Bayes classifier [2], the Rocchio algorithm [7] as the most popular learning method from information retrieval, a distance weighted $k$-nearest neighbor classifier [5][10], and the C4.5 decision tree/rule learner [6]. SVM training is carried out with the $SVM^{light}$[2] package. The $SVM^{light}$ package will be described in a forthcoming paper.

*Test Collections:* The empirical evaluation is done on two test collection. The first one is the "ModApte" split of the Reuters-21578 dataset compiled by David Lewis. The "ModApte" split leads to a corpus of 9603 training documents and 3299 test documents. Of the 135 potential topic categories only those 90 are used for which there is at least one training and one test example. After preprocessing, the training corpus contains 9962 distinct terms.

The second test collection is taken from the Ohsumed corpus compiled by William Hersh. From the 50216 documents in 1991 which have abstracts, the first 10000 are used for training and the second 10000 are used for testing. The classification task considered here is to assign the documents to one or multiple categories of the 23 MeSH "diseases" categories. A document belongs to a category if it is indexed with at least one indexing term from that category. After preprocessing, the training corpus contains 15561 distinct terms.

*Results:* Figure 2 shows the results on the Reuters corpus. The *Precision/Recall-Breakeven Point* (see e. g. [3]) is used as a measure of performance and *microaveraging* [10][3] is applied to get a single performance value over all binary classification tasks. To make sure that the results for the conventional methods are not biased by an inappropriate choice of parameters, all four methods were run after selecting the 500 best, 1000 best, 2000 best, 5000 best, (10000 best,) or all features using information gain. At each number of features the values $\beta \in \{0, 0.1, 0.25, 0.5, 1.0\}$ for the Rocchio algorithm and $k \in \{1, 15, 30, 45, 60\}$

[2] http://www-ai.informatik.uni-dortmund.de/thorsten/svm_light.html

| | | | | | SVM (poly) | | | | | SVM (rbf) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | degree $d =$ | | | | | width $\gamma =$ | | | |
| | Bayes | Rocchio | C4.5 | k-NN | 1 | 2 | 3 | 4 | 5 | 0.6 | 0.8 | 1.0 | 1.2 |
| earn | 95.9 | 96.1 | 96.1 | 97.3 | 98.2 | 98.4 | **98.5** | 98.4 | 98.3 | **98.5** | 98.5 | 98.4 | 98.3 |
| acq | 91.5 | 92.1 | 85.3 | 92.0 | 92.6 | 94.6 | **95.2** | 95.2 | 95.3 | 95.0 | 95.3 | 95.3 | **95.4** |
| money-fx | 62.9 | 67.6 | 69.4 | 78.2 | 66.9 | 72.5 | 75.4 | 74.9 | **76.2** | 74.0 | 75.4 | **76.3** | 75.9 |
| grain | 72.5 | 79.5 | 89.1 | 82.2 | 91.3 | 93.1 | **92.4** | 91.3 | 89.9 | **93.1** | 91.9 | 91.9 | 90.6 |
| crude | 81.0 | 81.5 | 75.5 | 85.7 | 86.0 | 87.3 | 88.6 | **88.9** | 87.8 | **88.9** | 89.0 | 88.9 | 88.2 |
| trade | 50.0 | 77.4 | 59.2 | 77.4 | 69.2 | 75.5 | 76.6 | 77.3 | **77.1** | 76.9 | 78.0 | **77.8** | 76.8 |
| interest | 58.0 | 72.5 | 49.1 | 74.0 | 69.8 | 63.3 | 67.9 | 73.1 | **76.2** | 74.4 | 75.0 | **76.2** | 76.1 |
| ship | 78.7 | 83.1 | 80.9 | 79.2 | 82.0 | 85.4 | 86.0 | **86.5** | 86.0 | **85.4** | 86.5 | 87.6 | 87.1 |
| wheat | 60.6 | 79.4 | 85.5 | 76.6 | 83.1 | 84.5 | 85.2 | **85.9** | 83.8 | **85.2** | 85.9 | 85.9 | 85.9 |
| corn | 47.3 | 62.2 | 87.7 | 77.9 | 86.0 | 86.5 | 85.3 | **85.7** | 83.9 | **85.1** | 85.7 | 85.7 | 84.5 |
| microavg. | **72.0** | **79.9** | **79.4** | **82.3** | 84.2 | 85.1 | 85.9 | 86.2 | 85.9 | 86.4 | 86.5 | 86.3 | 86.2 |
| | | | | | combined: **86.0** | | | | | combined: **86.4** | | | |

**Fig. 2.** Precision/recall-breakeven point on the ten most frequent Reuters categories and microaveraged performance over all Reuters categories. $k$-NN, Rocchio, and C4.5 achieve highest performance at 1000 features (with $k = 30$ for $k$-NN and $\beta = 1.0$ for Rocchio). Naive Bayes performs best using all features.

for the $k$-NN classifier were tried. The results for the parameters with the best performance on the test set are reported.

On the Reuters data the $k$-NN classifier performs best among the conventional methods (see figure 2). This replicates the findings of [10]. Compared to the conventional methods all SVMs perform better independent of the choice of parameters. Even for complex hypotheses spaces, like polynomials of degree 5, no overfitting occurs despite using all 9962 features. The numbers printed in bold in figure 2 mark the parameter setting with the lowest VCdim estimate as described in section 3. The results show that this strategy is well-suited to pick a good parameter setting automatically and achieves a microaverage of 86.0 for the polynomial SVM and 86.4 for the RBF SVM. With this parameter selection strategy, the RBF support vector machine is better than $k$-NN on 63 of the 90 categories (19 ties), which is a significant improvement according to the binomial sign test.

The results for the Ohsumed collection are similar. Again $k$-NN is the best conventional method with a microaveraged precision/recall-breakeven point of 59.1. C4.5 fails on this task (50.0) and heavy overfitting is observed when using more than 500 features. Naive Bayes achieves a performance of 57.0 and Rocchio reaches 56.6. Again, with 65.9 (polynomial SVM) and 66.0 (RBF SVM) the SVMs perform substantially better than all conventional methods. The RBF SVM outperforms $k$-NN on all 23 categories, which is again a significant improvement.

Comparing training time, SVMs are roughly comparable to C4.5, but they are more expensive than naive Bayes, Rocchio, and $k$-NN. Nevertheless, current research is likely to improve efficiency of SVM-type quadratic programming problems. SVMs are faster than $k$-NN at classification time. More details can found in [3].

## 6 Conclusions

This paper introduces support vector machines for text categorization. It provides both theoretical and empirical evidence that SVMs are very well suited for text categorization. The theoretical analysis concludes that SVMs acknowledge the particular properties of text: (a) high dimensional feature spaces, (b) few irrelevant features (dense concept vector), and (c) sparse instance vectors.

The experimental results show that SVMs consistently achieve good performance on text categorization tasks, outperforming existing methods substantially and significantly. With their ability to generalize well in high dimensional feature spaces, SVMs eliminate the need for feature selection, making the application of text categorization considerably easier. Another advantage of SVMs over the conventional methods is their robustness. SVMs show good performance in all experiments, avoiding catastrophic failure, as observed with the conventional methods on some tasks. Furthermore, SVMs do not require any parameter tuning, since they can find good parameter settings automatically. All this makes SVMs a very promising and easy-to-use method for learning text classifiers from examples.

## References

1. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, November 1995.
2. T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *International Conference on Machine Learning (ICML)*, 1997.
3. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, Universität Dortmund, LS VIII, 1997.
4. J. Kivinen, M. Warmuth, and P. Auer. The perceptron algorithm vs. winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. In *Conference on Computational Learning Theory*, 1995.
5. T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
6. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
7. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall Inc., 1971.
8. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
9. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.
10. Y. Yang. An evaluation of statistical approaches to text categorization. Technical Report CMU-CS-97-127, Carnegie Mellon University, April 1997.
11. Y. Yang and J. Pedersen. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning (ICML)*, 1997.