

A Comparison of Online learning Naïve Bayes Classifier on RSS Feeds using SPARK

Authors:

Ahmet Anil PALA & Franziska ADLER

TU BERLIN

Advanced Information Management III
Scalable Data Analytics and Data Mining

Motivation & Problem Statement

Many practical applications rely on immediate data. The need for solutions to continuously conduct analyses as mathematical computations on large data amounts led to new technologies in the past years. Stream processing operates on real-time data for example through stream windowing and analytical operations within those. Since data distributions might not remain static over time a precomputed (batch) model for analysis

can produce poorly results after some time. The reconstruction or updating of the underlying analytical model to receive accurate results is required. This problem is known as Concept Drift. In this project we are using unstructured streaming data from BBC RSS feeds in order to classify them to their corresponding category. We focuss on the evaluation of a Naïve Bayes Classifier with different model-update approaches to compare their analytical performance according to Concept Drift.

Data & Setup

For our evaluation purposes of classifying streamed textdata we are using RSS feeds from BCC. A RSS feed is a collection of tags in xml structure which contains tags for title, a description and an url among other tags. Those listed are used in our application. Our data points are constructed from title and description of the feed, the url gives us the respective label. Furthermore we are using SPARK for parallel execution of streaming and data classification. The available library for Machine Learning Algorithms MLlib provides us with a Naive Bayes Classifier.

Streaming

In order to deal with the contonious nature of the data, traditional programming primitives are not of much help. In order to make programmers's job easier, libraries providing higher lever abstractions are introduced. Knowing that, for a news feed classifier we need to split the 'main' stream into multiple streams to treat different portions of stream differently. For example, there needs to be at least two channels of streams branching out from the main one having the test items and training items. The main flow of the streaming logic we used is depicted in the figure below.

Methodology

Batch and On-line

The batch model implements a classic Training-Testing-Phase setup. A subset training points are pre-collected from a stream and used to build a final model on which three testsets are applied. The batch model functions as a reference model to observe the performance of the initial training over time.

Bruteforce

On the other hand On-line Learning will change the model with the arriving of new data points. The brute-force approach updates the model after a period of time through retraining. Based on a sliding window over the stream with a constant number of data points the model is rebuilt.

Threshold-triggered

As a variation of the brute-force model-update the threshold triggered one will rebuild the model on a sliding window as soon as the performance of our model is beneath a certain threshold.

Incremental

In the incremental model, unlike the other update models, following an initial training-only phase, all the stream items are used for training right after

the prediction. This way the we aimed to achieve a real-time response to the changes in the data stream. Model is updated by 'dampening' the feature vectors by a function of the learning rate used. Initial training size of 600 and $\text{abs}(\arctan(x)/(\pi/2))$ as the learning function ,guaranteeing learning rate converges to 0 or 1 as the error gets smaller or infinitely high respectively, are used in our implementation.

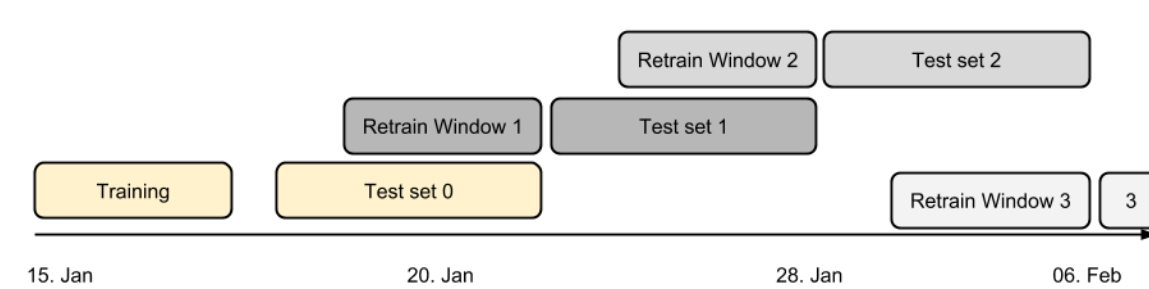
Batch

For Offline learning the model was built within the first three days out of 600 data points. The following four days with 1248 data points created the first test set for this model. A second and a third test set were created after the previous test phase and contained 1102 and 1172 data points, respectively RSS feeds.

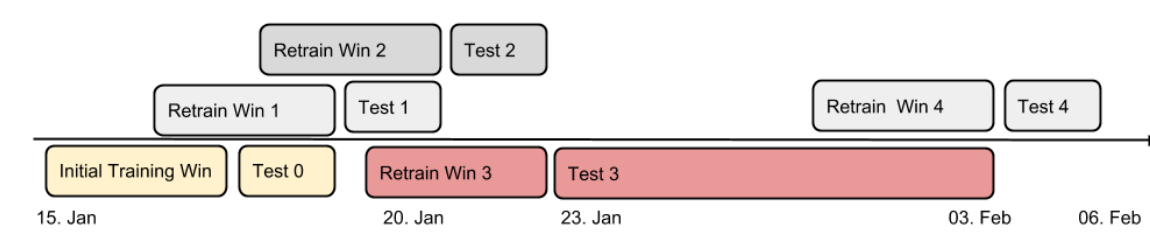


Bruteforce and Error-triggered

The window size for the bruteforce model contains 600 data points. After 400 test points we led it retrain with the last 600 data points. We then waited for 1000 data points and second retrain phase followed which was tested also with 1000 new data points.



The window size of the error-triggered model is constructed like the brute-force update out of 600 datapoints. Due to our observations an acceptable accuracy threshold of 63% seemed reasonable. A sanity window of testpoints ensures that at least 300 new data points arrive and based on their performance the model is rebuilt or not.

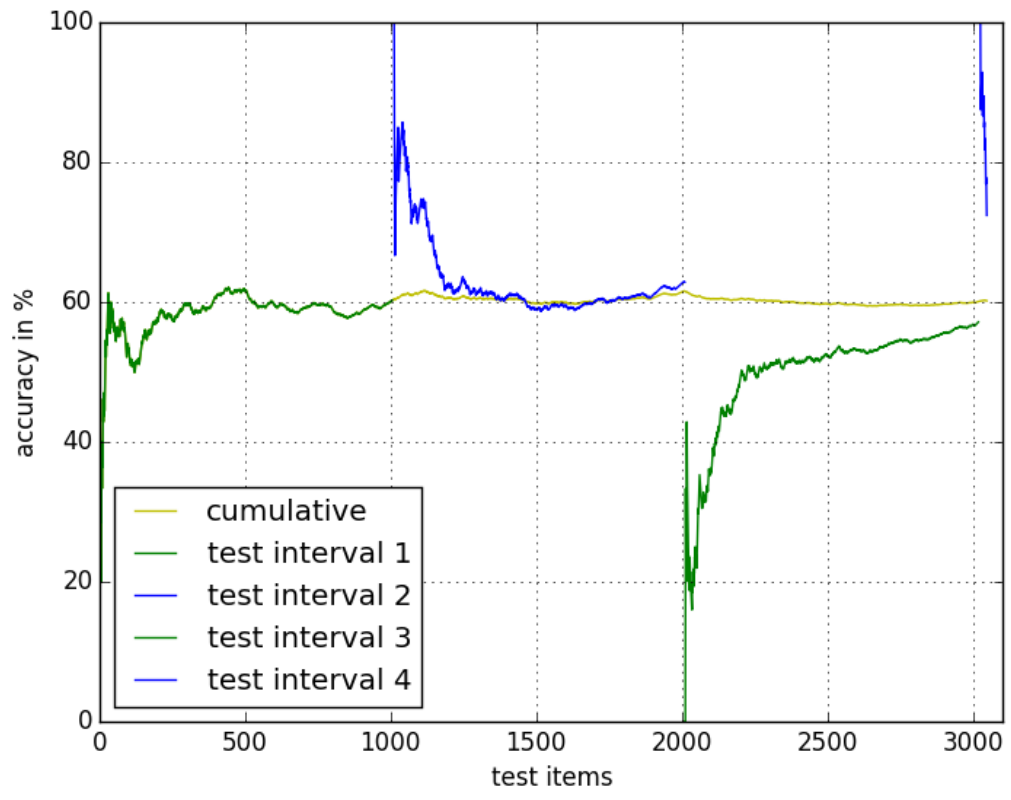


Incremental up- dates

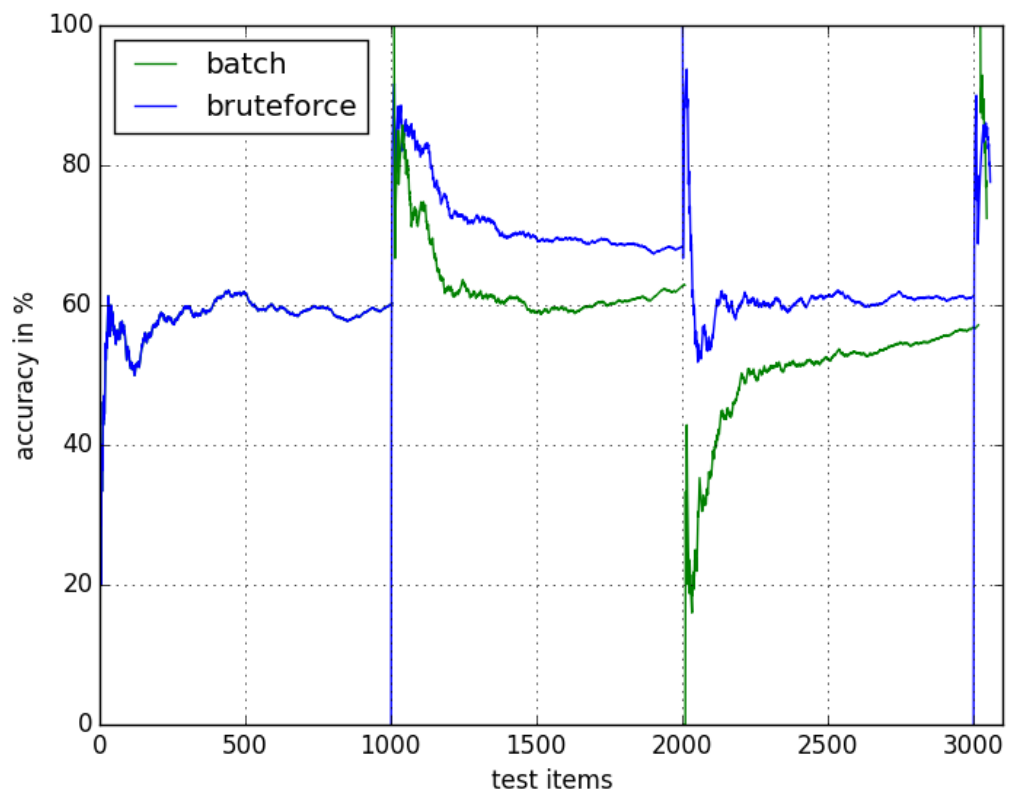
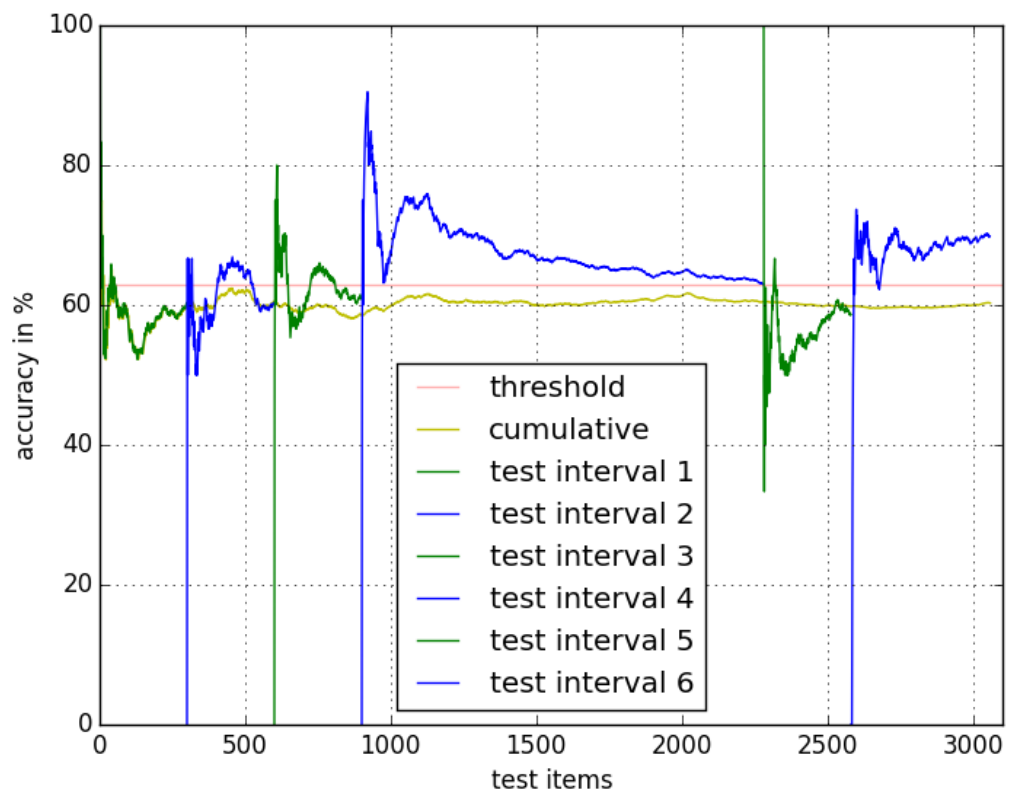
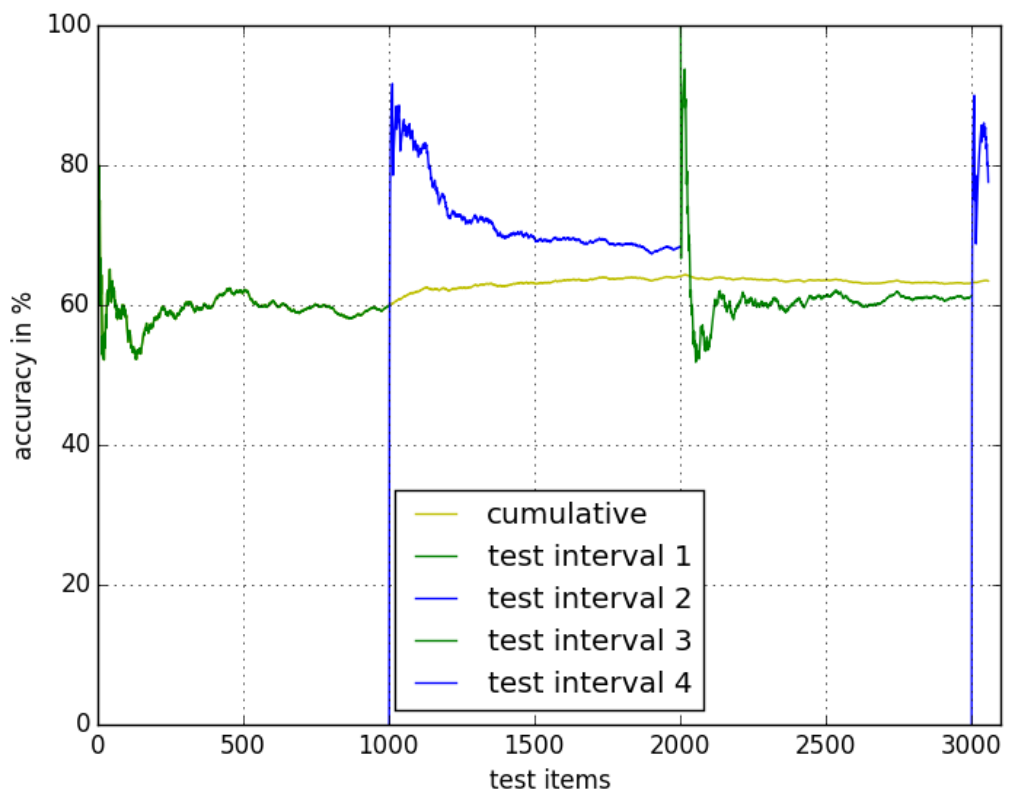
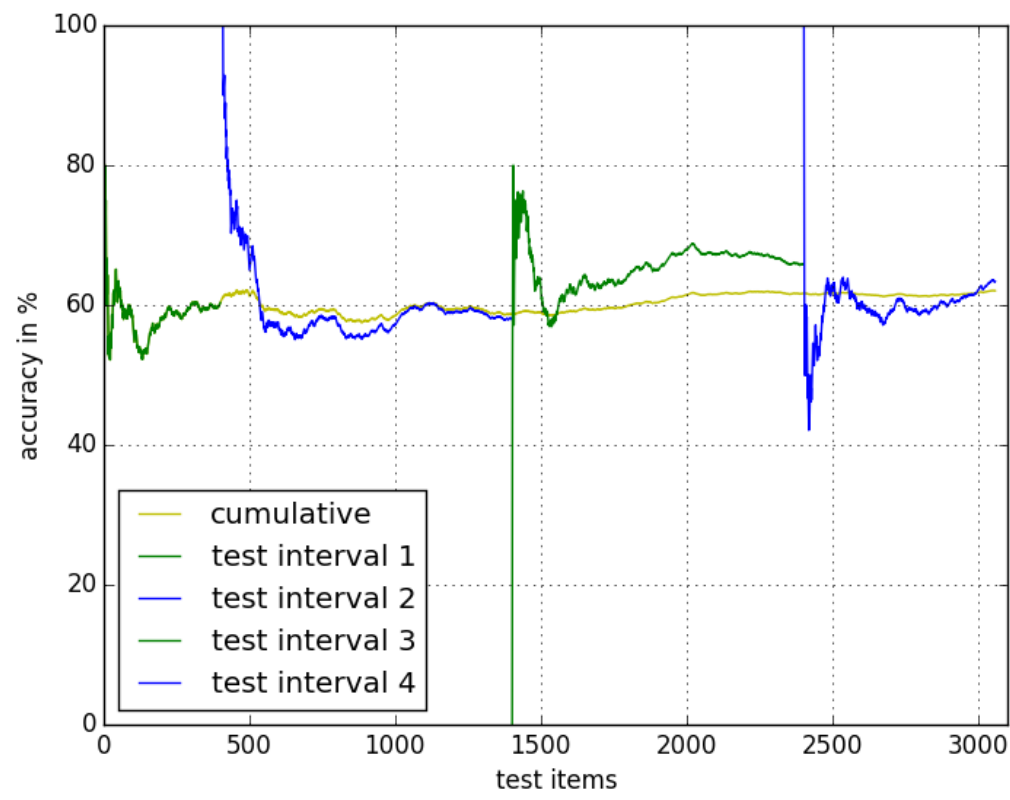
The initial training size of 600 is used for this model update method. For the incremental model, The initial training size of 600 is used for this model update method. Dampening factor is simply the value of the learning rate function subtracted by 1. Dampening factor is sim-

ply the value of the learning rate function subtracted by 1. As for the learning rate function, we used $\text{abs}(\arctan(x))/(\pi)$. This is because we want learning rate values between 1 and 0 which converges to 0 as the error gets smaller and converges to 1 as the error gets infinitely high.

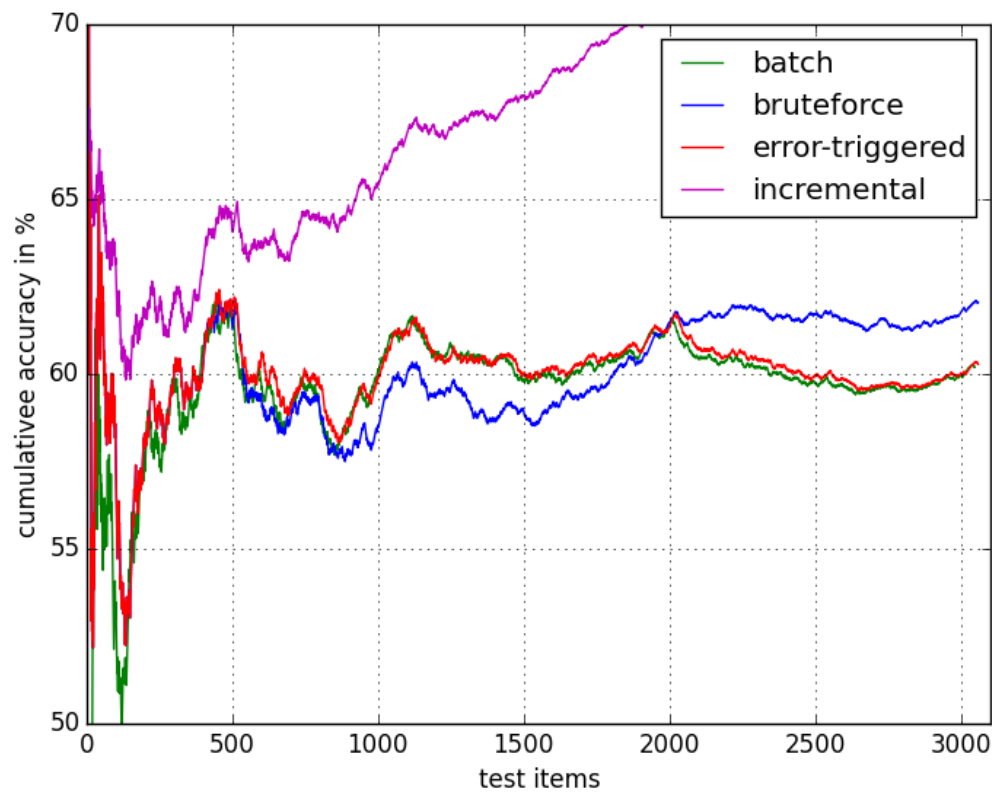
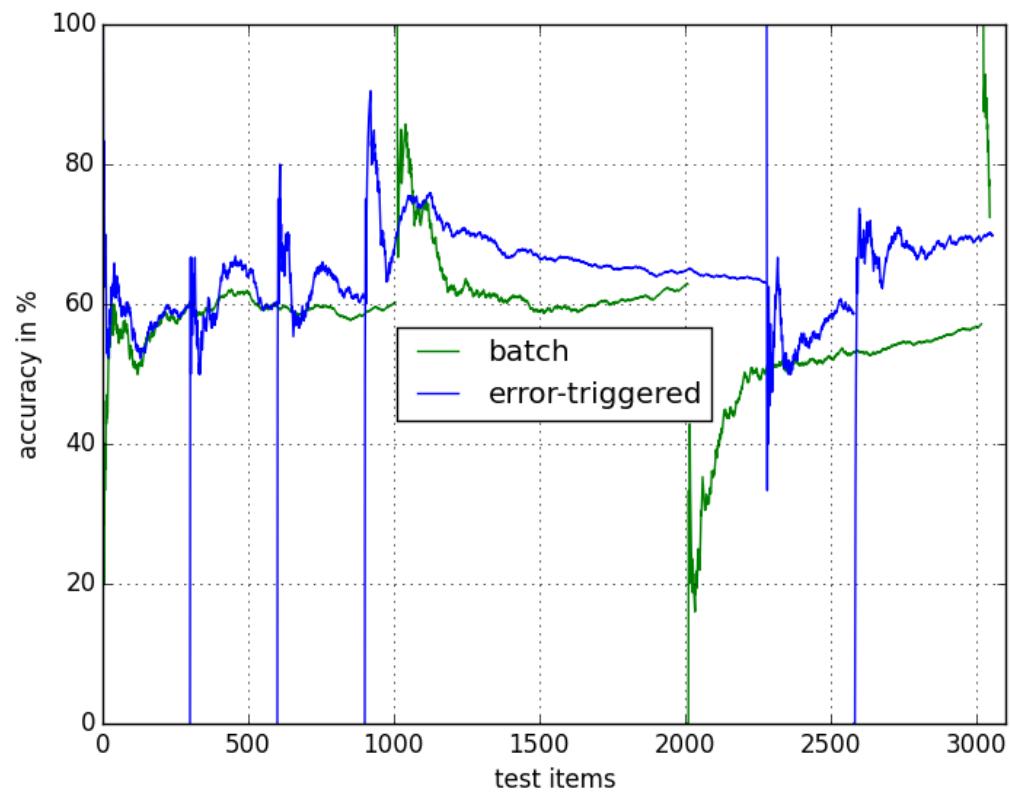
The offline versions result with 1000 test points per interval is always around 60% even though it decreased at the last testing phase. The offline versions result with 1000 test points per interval is always around 60% even though it decreased at the last testing phase. The offline versions result with 1000 test points per interval is always around 60% even though it decreased at the last testing phase. The offline versions result with 1000 test points per interval is always around 60% even though it decreased at the last testing phase. The offline versions result with 1000 test points per interval is always around 60% even though it decreased at the last testing phase. The offline versions result with 1000 test points per interval is always around 60% even though it decreased at the last testing phase.



though it decreased at the last testing phase. The offline versions result with 1000 test points per interval is always around 60% even though it decreased at the last testing phase.



The bruteforce model which updates after



Literatur

- [1] M. HALLO & L. DADF,*Depressing literature title*, page 33, NoPressQL, 1883.
- [2] K. SULLIVAN, *Monotonously down draggin words*, page 1, Academic Press, 2001.