Adaptive Term Weighting through Stochastic Optimization

Michael Granitzer[1,2]

[1] Graz University of Technology, Graz, Austria
mgranitzer@tugraz.at,
http://kmi.tugraz.at/
[2] Know-Center Graz, Graz, Austria
mgrani@know-center.at,
http://www.know-center.at/

Abstract. Term weighting strongly influences the performance of text mining and information retrieval approaches. Usually term weights are determined through statistical estimates based on static weighting schemes. Such static approaches lack the capability to generalize to different domains and different data sets. In this paper, we introduce an on-line learning method for adapting term weights in a supervised manner. Via stochastic optimization we determine a linear transformation of the term space to approximate expected similarity values among documents. We evaluate our approach on 18 standard text data sets and show that the performance improvement of a k-NN classifier ranges between 1% and 12% by using adaptive term weighting as preprocessing step. Further, we provide empirical evidence that our approach is efficient to cope with larger problems.

1 Introduction

Term weighting schemes - statistical models for estimating the importance of a term - directly impact the information retrieval ranking and text mining accuracy; a fact shown by information retrieval evaluation initiatives like for example TREC[3], CLEF [4] or INEX[5]. Most of todays weighting techniques rely on parametric approaches, which have a defined functional form and a (usually small) number of parameters. Examples are *tfidf* weighting [1], *BM25* [2], language modeling [3], axiomatic weighting [4] etc. Parametric approaches suffer from the drawback that they do not adapt to particular domains and user needs. For example, searching for research publications requires a different weighting scheme than searching for restaurants.

To overcome the rigidity of parametric weighting schemes Metzler and Zaragoza [5] introduced semi-parametric and non-parametric weighting schemes recently. They extended Anh and Moffats [6] approach to binned ranking as a general form of dimensionality reduction. Binned ranking employs a partially order on terms and groups

[3] Text REtrieval Conference (http://trec.nist.gov/)
[4] Cross-Language Evaluation Forum (http://clef-campaign.org/)
[5] The INitiative for the Evaluation of XML retrieval http://www.inex.otago.ac.nz/

them into k bins. A weight is assigned to every bin and subsequently used for ranking/similarity calculations [6]. Semi-parametric schemes now estimate bin weights similar to parametric schemes, i.e. in a functional form. More interestingly, non-parametric weighting schemes directly modify the bin weights in a supervised manner in order to optimize the ranking. As argued by Metzler and Zaragoza [5], this supervised approach has promising properties under the assumption that enough supervision can be provided. In particular it allows adapting ranking functions to particular domains/user needs, while binning reduces the dimensionality of the vector space and therewith keeps the parameter estimation tractable.

A related approach of adapting weight parameters is presented in [7]. Ernandes et. al. developed a context sensitive, adaptive weighting scheme. Here the weight of a word depends on the words statistical distribution and the words context defined by surrounding words. With resilient parameter adaptation gradient descent the actual influence factors among words and their context are estimated in a batch like manner.

Adapting distance metrics for classification has been described in [8]. Shwartz et. al. optimized the Mahalanobis distance via linear transformations in order to boost the accuracy of a k-NN classification algorithm, which can be seen as implicit application of an weighting scheme.

In this paper, we follow this line of research by learning a linear transformation to optimize a metric space. This linear transformation approximates the corpus dependent component of a term weight in order to optimize the expected similarities among documents. Since optimization is based on pair wise drawn data elements - yielding a quadratic complexity in terms of data set size - we use stochastic on-line gradient descent algorithms. Our approach allows adapting dot product based text mining algorithms to particular user and/or domain needs by providing expected similarities among documents, while keeping the algorithms itself unchanged.

With our work we carry on the ideas provided by Metzler and Zaragoza [5], but do not rely on binning for dimensionality reduction. Instead we use online stochastic gradient descent methods to keep the problem tractable. In contrast to Shwartz et. al. [8], we focus on inner product based similarity measures rather than on distances and, different than Ernandes et. al. [7], we adapt well known retrieval models. Hence, our approach is directly applicable in existing systems. Therefore, we contribute to adaptive, non-parametric weighting schemes by

- presenting a stochastic on-line approach to approximate the corpus dependent component of a term weight in order to optimize the expected similarities among documents
- showing that using pair wise training in such a setting is efficient
- showing that the adapted term weights improve nearest neighbor search as well as k-Nearest Neighbor classification.

[6] With having only one bin, parametric weighting schemes can be seen as special case to binned retrieval.

The remainder of this paper is structured as follows: section 2 shows that linear transformations of a vector space correspond to some well known weighting schemes. This property is exploited in section 3 by approximating the linear transformation through on-line stochastic gradient descent. Experimental results are outlined in section 4, while section 5 concludes our work and points to future topics.

2 Term Weighting as Linear Transformation

Our approach relies on the observation that most weighting schemes are linear transformations $L : \Re^d \rightarrow \Re^d$ of a vector space $D = \{d_1 \dots d_N\}, d_j \in \Re^d$ where d_j denotes the term vector of document v and $|D|$ the dimensionality of the vector space. One example for such an linear transformation is the well known *tfidf* family of weighting schemes. In its most basic form, the weight $w_{k,j}$ of the term k (corresponding to the k^{th} dimension of the vector space D) in document j is calculated as

$$w_{k,j} = tf_{k,j} \cdot idf_k = occ_{k,j} \cdot \log(\frac{N}{n_k}) \tag{1}$$

with $occ_{k,j}$ being the number of occurrences of term k in document j, N being the number of documents and n_k being the number of documents containing term k[7]. Obviously the weighting scheme has a document dependent part, the term frequency *tf*, and a corpus dependent part, the inverse document frequency *idf*. Rewriting the *idf*-part as diagonal matrix $L = diag(idf)$ - with idf being a d-dimensional vector representing the idf values of each dimension as $idf_k = \log(\frac{N}{n_k})$ and assuming that the document vectors contain the document specific part, i.e. $d_{j,k} = occ_{k,j}$ - the *tfidf* weighting scheme can be written as

$$\breve{d_i} = Ld_i \tag{2}$$

Similarly to *tfidf* , the Okapi *BM25* [2] consists of a document and a corpus specific term. Herein, the weight $w_{k,j}$ is given as

$$w_{k,j} = tf_{k,j} \cdot idf_k = \frac{(k_1+1)occ_{k,j}}{k_1(1-b+b\frac{l_d}{l_a})+occ_{k,j}} \cdot \log(\frac{N-n_k+0.5}{n_k+0.5}) \tag{3}$$

with l_d being the length of the documents (i.e. number of total terms), l_a being the average length of documents in a corpus, and b and k_1 being positive constants to tune the impact of document length and/or the term frequency. Again, by separating the document from the corpus specific part *BM25* -weighting can be seen as linear transformation of the according vector space.

The TREC evaluation series[8] as well as several research groups have shown that different weighting schemes, and therefore different linear transformations, have a dramatic impact on the performance of different algorithms. Moreover, differences depend

[7] There are a number of different *tfidf* functions with slightly different calculations of the term frequency *tf*and inverse document frequency *idf*. For details see for example [9].

[8] Text REtrieval Conference (`http://trec.nist.gov/`)

strongly on the underlying corpus as well as on the task at hand such that no weighting scheme outperforms other weighting schemes in general.

Our approach targets this problem by automatically adapting the linear transformation to optimize expected similarities among documents. We focus on cosine based similarity measures since they are commonly used in information retrieval and text mining algorithms like vector space retrieval, clustering or classification and, as we will show, allow the integration of linear transformation for their optimization. In particular, we start with the cosine similarity defined as

$$s_{i,j} = \frac{d_i' \cdot d_j}{\|d_i\|\|d_j\|} \tag{4}$$

which in case that $\|d_i\| = 1$ and $\|d_j\| = 1$ is reduced to a simple dot product ($'$ denotes the transpose of the vector). Hence, applying a weighting scheme as linear transformation, the resulting similarity becomes

$$s_{i,j} = \frac{(Ld_i)' \cdot Ld_j}{\|Ld_i\|\|Ld_j\|} \tag{5}$$

In order to optimize the linear transformation, we assume that the target similarity is given and that we have to adapt the linear transformation to approximate the expected target similarity. Such similarities could be provided for example by relevance judgments, user feedback or information visualization techniques [10]. Assuming that a sufficient number of target similarities are given, we expand the linear transformation in equation 5, rearrange coefficients and, under Euclidean norm we get

$$s_{i,j} = \frac{\sum_k l_k^2 d_{i,k} d_{j,k}}{\sqrt{\sum (l_k \cdot d_{i,k})^2}\sqrt{\sum (l_k \cdot d_{j,k})^2}} \tag{6}$$

The impact of term k on the similarity is defined by the transformation coefficients l_k, which have to be adapted over all terms in order to return the expected similarity. This adaption via on-line learning is shown in the following section.

3 On-line Learning for Term Weighting

Given a target similarity between two documents, denoted as $\hat{s}_{i,j}$, our goal is now to find a linear transformation L to approximate $\hat{s}_{i,j}$ as good as possible. Formally we want

$$\hat{s}_{i,j} \hat{=} \frac{\sum_k l_k^2 d_{i,k} d_{j,k}}{\sqrt{\sum (l_k \cdot d_{i,k})^2}\sqrt{\sum (l_k \cdot d_{j,k})^2}} \tag{7}$$

yielding an equation system consisting of N^2 equations with $|D| - 1$ degrees of freedom.

Solving this equation system exhibits two problems: First, the existence of an exact solution is not guaranteed. This can be easily shown by assuming that two documents share no common term, but must have a similarity larger than 0. Second, the number of equations may be too large for practical problems. In most application scenarios a corpus contains several thousand documents, yielding a large set of equations through the quadratic dependency on the number of documents.

We favor on-line approaches over batch approaches due to several reasons. Typical text data sets have a high number of documents and correspondingly a high dimensionality of the vector space. So batch optimization would become infeasible since it requires to store a $n^2 \times |D|$ sparse matrix to represent all inner document products $d_j \cdot d_i$. On-line approaches allow to randomly draw examples and optimize the cost function on a per-example basis, thereby overcoming the storage problem. Regarding convergence, which directly translates to runtime complexity, stochastic approaches can achieve higher convergence rates than batch algorithms in case of redundant data sets [11], i.e. data sets where examples share similar properties. Redundancies will likely occur in our problem setting since we can assume that document vectors share similar term distributions while having similar target similarities. Hence, stochastic on-line gradient descent seems to be suited best for the task.

For formulating eq. 7 as optimization problem we use a quadratic loss function. Formally we have

$$E(L) = \sum_{\forall i,j,i \neq j} (\hat{s}_{i,j} - s_{i,j})^2 \tag{8}$$

with $s_{i,j}$ begin the similarity as defined in equation 6. By being differentiable, the quadratic loss function allows us to estimate the gradient and to use gradient descent approaches for determining a good solution L^*.

For performing stochastic on-line gradient descent, we randomly draw two documents and their corresponding target similarity. Afterwards, the loss function $E(L)$ and its gradient w.r.t. the current data examples is calculated as

$$\frac{\partial E(L,i,j)}{\partial l_k} = 4\delta l_k d_{i,k} d_{j,k} \tag{9}$$

with $\delta = \hat{s}_{i,j} - s_{i,j}$ and $E(L,i,j)$ being the loss for the document pair i, j.

To avoid negative weights, updates dropping a dimension below zero are ignored. Instead the current weight of this dimension is made smaller by dividing it by two. Hence, the new weight is $l_k = l_k + \Delta_k$ with

$$\Delta_k = \begin{cases} \eta \delta l_k d_{i,k} d_{j,k} & if l_k + \Delta_k > 0 \\ l_k/2 & if l_k + \Delta_k < 0 \end{cases} \tag{10}$$

and $\eta > 0$ being the learning rate resp. step size. Convergence depends strongly on the chosen learning rate. A low learning rate slows down convergence, while a too large learning rate may avoid convergence towards the optimal solution. Therefore, we choose to anneal the learning rate logarithmically via

$$\eta_t = \eta / \log(t+2) \tag{11}$$

with *t* being the current step.

Clearly, this setting allows us to easily add new examples and adapt the linear transformation on the fly. So if new target similarities are provided to the algorithm, the linear transformation can be adapted easily and with rather low computational complexity. Hence, user feedback can be integrated straightforward.

4 Experiments

In order to evaluate our approach to adaptively weight terms we conducted two sets of experiments on 18 standard text data sets. In the first set of experiments we evaluated the correctness of our approach by learning a known *tfidf* scheme (see section 4.2). The second set of experiments targets the practical case of text classification. We show that our approach, used as preprocessing step, improves a k-NN classifier in its classification accuracy (see section 4.3)

4.1 Data Sets

For our experiments we used the Cluto data sets [12]. Those data sets have been explored mostly for document clustering and provide a collection of standard text data sets given as document term matrix. The datasets - compiled from well known benchmark data sets like TREC, Cranfield and the Reuters-21578 corpus - have been already preprocessed by removing stop words and are stemmed using Porter's algorithm. All terms that occur only in one document were eliminated (see [12] for further details)[9]. Table 1 depicts the different data sets and their properties.

Table 1. Data Set Overview

Data Set	# Exa.	# Terms	# Classes	Data Set	# Examples	# Terms	# Classes
Classic	7094	41681	4	Re0	1504	2886	13
Cranmed	2431	41681	2	Re1	1657	3758	25
Fbis	2463	2000	17	Reviews	4069	126373	5
Hitech	2301	126373	6	Sports	8580	126373	7
K1a	2340	21839	20	Tr11	414	6429	9
K1b	2340	21839	6	Tr12	313	5804	8
La1	3204	31472	6	Tr31	927	10128	7
La2	3075	31472	6	Tr41	878	7454	10
New3	9558	83487	44	Wap	1560	8460	20

[9] One exception is the FBIS data set, who also had been pruned to the 2000 most important features according to a forum entry of George Karypis `http://glaros.dtc.umn.edu/gkhome/node/353`.

Table 2. Highest correlation coefficient ρ between the optimized and the original tfidf vector. The number of examples drawn to achieve the correlation are shown for the different learning rates η.

Data Set	η=0.1		η=1.0		dataset	#terms
	ρ	# examples drawn	ρ	# examples drawn	size	
Classic	*0.759*	85,128	0.845	85,126	7,094	41,681
Cranmed	*0.706*	29,172	0.782	29,172	2,431	41,681
Fbis	*0.832*	1,151	0.828	197	2,463	2000
Hitech	*0.767*	27,612	0.824	27,609	2,301	126,373
K1a	*0.784*	28,080	0.847	28,075	2,340	21,839
K1b	*0.784*	28,080	0.847	28,075	2,340	21,839
La1	*0.771*	38,448	0.834	23,067	3,204	31,472
La2	*0.737*	36,900	0.826	32,612	3,075	31,472
New3	*0.812*	114,696	0.829	114,695	9,558	83,487
Re0	*0.872*	18,021	0.876	5,358	1,504	2,886
Re1	*0.824*	19,808	0.814	9,422	1,657	1,657
Reviews	*0.762*	48,828	0.823	48,824	4,069	126,373
Sports	*0.789*	102,954	0.867	102,958	8,580	126,373
Tr11	*0.822*	958	0.855	53	414	6,429
Tr12	*0.820*	354	0.819	17	313	5,804
Tr31	*0.824*	11,109	0.850	1,504	927	10,128
Tr41	*0.832*	10,525	0.839	1,104	878	7,454
Wap	*0.817*	18,720	0.824	17,023	1,560	8,460

4.2 Learning a given Weighing Scheme

In order to evaluate that we can efficiently approximate a weighting scheme based on provided similarities, our first experiment reconstructs the *tfidf* weighting scheme. The target similarity values are estimated by applying a *tfidf* weighting scheme (see equation 1) to each document and by calculating the dot product similarity matrix (see equation 6) accordingly. The linear transformation L is initialized uniformly as diagonal matrix with $1/d$ in each diagonal element. While doing the online gradient descent, we evaluated the Pearsons correlation coefficient ρ between the approximated transformation L and the original *tfidf* transformation after every 25 examples drawn. The correlation coefficient determines how good both transformations match independent of their actual scaling. Hence, we measure whether our approach correctly approximates the differences of term importance. Table 2 depicts the results for all data sets.

Accuracy: Two important properties regarding accuracy can be observed: Firstly, we achieve high correlation coefficients for every data sets, indicating that our approach is capable to identify the linear transformation accompanying a given similarity distribution. Second, the solution could be achieved within a small numbers of iterations. While it depends on the learning rate used, the highest correlation coefficients could be achieved in the range of around 2-10 times the number of term vectors in the data set. Hence, every run did only see a fraction of all possible N^2 examples. Intuitively this seems to be valid since the number of parameters is much smaller than the number of examples. Thus, it can be assumed that *the data set is highly redundant* which is *optimal for stochastic approaches and allows to find good transformations efficiently*.

Convergence: Another important aspect to discuss is convergence. Figure 1 shows the convergence for $\eta = 1.0$ and $\eta = 0.1$ respectively. As expected, the different learning rates influence convergence stability and $\eta = 0.1$ achieves a more stable accuracy over the different data sets. Exceptions are the fbis and the Tr11 data sets. They rapidly achieve high correlation values but drop afterwards. We see the reason therefore in the rather different numbers of dimensions (Fbis) and examples (Tr11). However, for the majority of data sets the solution is improved with every iteration.

We conclude that all data sets achieve high correlation values rather fast and continue to improve them slowly. *Hence, on-line gradient descent allows to efficiently approximate a given weighting scheme.*

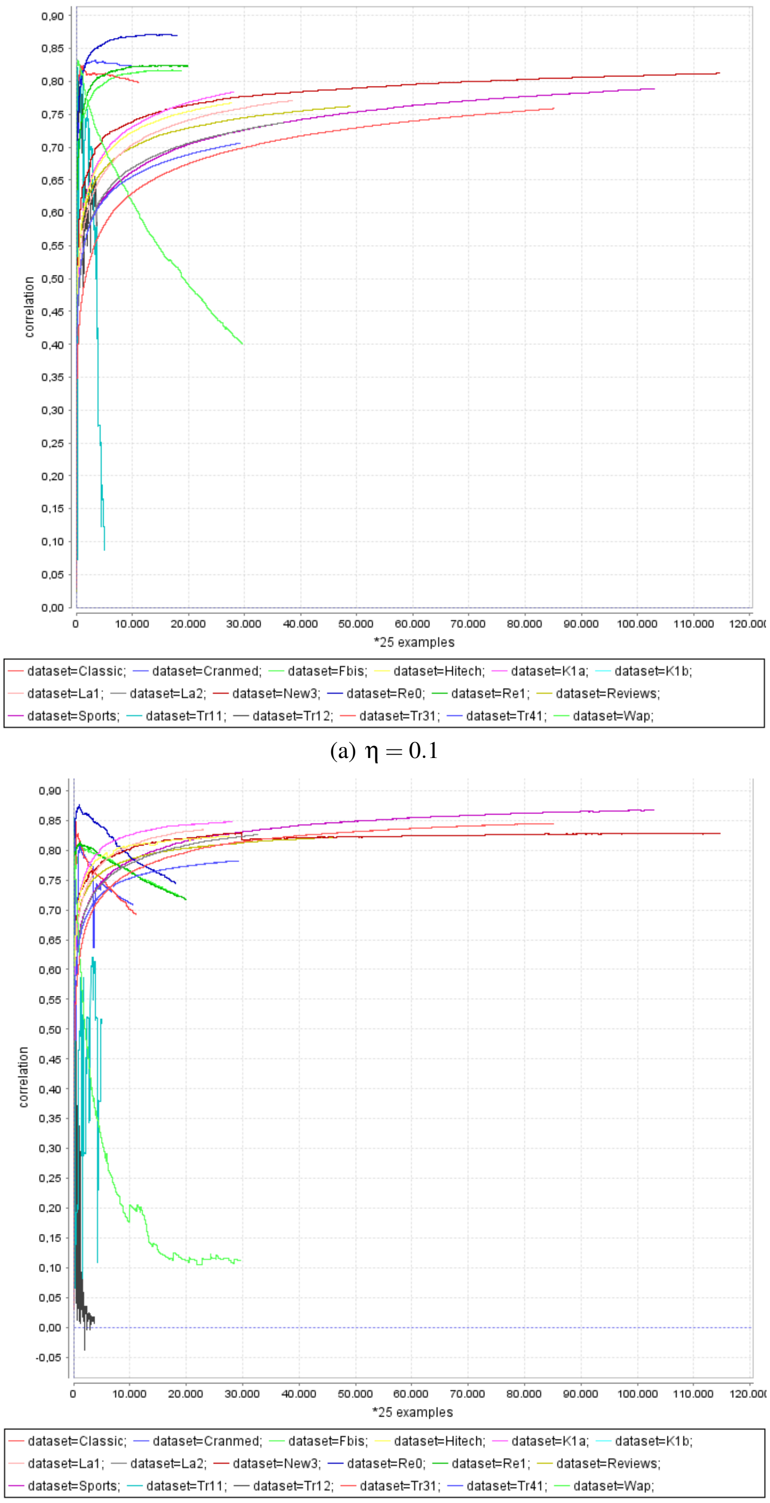
4.3 Weight Adaption for k-NN Classification

To estimate the impact on nearest neighbor search and classification, we considered the improvement of k-NN classifiers via adapting the term weighting scheme based on class information in the training set. k-NN classifiers assign a document to the class where the k nearest neighbors of this document belong to. In adapting the weighting scheme such that documents of the same class become more similar while documents belonging to different classes become more dissimilar, we assume that k-NN classification performance has to increase. Hence, for this experiment the target similarity among two documents is set to 1.0 if they belong to the same class and 0.0 otherwise. Note that we are moving from a continuous spectrum of target values to a discrete one. Besides the actual improvement in classification accuracy, we also evaluate whether stochastic gradient descent can improve a particular parametric weighting scheme. Therefore, we distinguish whether documents are already weighted with a corpus specific part (denoted *tfidf* & *BM25* in the result tables) or not (denoted with *tf* in the result tables). In both cases the found linear transformation adds a multiplicative scaling term to each dimension of the vector space.

In our experiments, three folded cross-validation splits the data set into a training and test data set for each fold[10]. During the learning phase, which usually involves no calculations in classical k-NN classifiers, stochastic gradient descent updates the weighting scheme iteratively. Every time after randomly drawing half of the total number of term vectors n, the F_1 measure is evaluated on the test set. Results for all data sets are depicted in table 3 showing the improvements for the different weighting schemes. The learning rate η has been set to 2.0.

Improvement: While the quadratic loss function may not be optimal for the discrete classification task in this use case, results show that a significant improvement can be

[10] We choose three folded cross-validation over ten-folded cross validation in order to reduce the required computation time for our experiments.



(a) $\eta = 0.1$

(b) $\eta = 1.0$

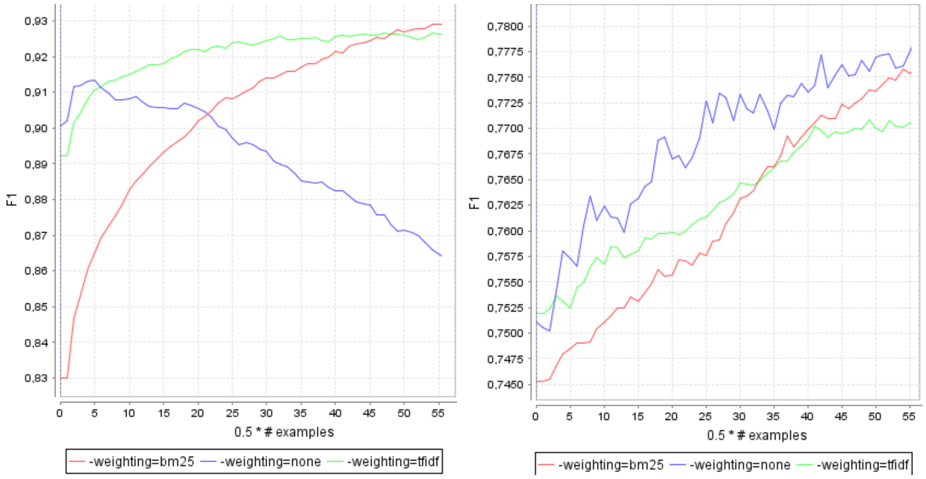
Fig. 1. Correlation per 25 examples for different learn rates η

Table 3. Improvement & achieved F_1 value for 3-folded k-NN Classification. F_1^{max} depicts the maximal achieved F_1 value, F_1^{ori}, the original F_1 value without adaptive weighting and $\frac{F_1^{max}}{F_1^{ori}}$ the corresponding improvement

	BM25			*tf*			*tfidf*		
	F_1^{max}	F_1^{ori}	F_1^{max}/F_1^{ori}	F_1^{max}	F_1^{ori}	$\frac{F_1^{max}}{F_1^{ori}}$	F_1^{max}	F_1^{ori}	F_1^{max}/F_1^{ori}
Classic	0.9305	0.8293	*1.122*	0.9150	0.9004	1.016	0.9287	0.8914	1.042
Cranmed	0.9962	0.9933	1.003	0.9703	0.9617	1.009	0.9892	0.9802	1.009
Fbis	0.7421	0.7262	1.022	0.7146	0.7033	1.016	0.7053	0.6891	1.023
Hitech	0.5879	0.5827	1.009	0.6360	0.6100	1.043	0.6020	0.5764	*1.045*
K1a	0.3620	0.3574	1.013	0.6502	0.6121	1.062	0.5728	0.5581	1.026
K1b	0.4467	0.4124	1.083	0.8516	0.8226	1.035	0.7886	0.7822	1.008
La1	0.7804	0.7437	1.049	0.7801	0.7510	1.039	0.7716	0.7518	1.026
La2	0.7513	0.7178	1.047	0.7900	0.7663	1.031	0.8006	0.7829	1.023
New3	0.6813	0.6726	1.013	0.7095	0.6971	1.018	0.6819	0.6697	1.018
Re0	0.6666	0.6193	1.076	0.7233	0.6903	1.048	0.7290	0.6658	1.095
Re1	0.6742	0.6199	1.088	0.6306	0.5855	1.077	0.7266	0.6745	1.077
Reviews	0.8904	0.8772	1.015	0.9064	0.8815	1.028	0.9050	0.8673	1.043
Sports	0.9438	0.9304	1.014	0.8937	0.8932	1.001	0.9144	0.8975	1.019
Tr11	0.6762	0.6642	1.018	0.7685	0.7403	1.038	0.7544	0.7039	1.072
Tr12	0.6405	0.6313	1.015	0.7938	0.7495	1.059	0.8197	0.7944	1.032
Tr31	0.8837	0.8757	1.009	0.8865	0.8626	1.028	0.8999	0.8880	1.013
Tr41	0.9031	0.8819	1.024	0.8925	0.8722	1.023	0.9159	0.9048	1.012
Wap	0.4347	0.4299	1.011	0.6686	0.6183	*1.081*	0.6255	0.6041	1.035
Average			1.035			1.036			1.034
Std. Dev.			0.035			0.022			0.025

achieved. In general, every run seems to improve the F_1 measure. The improvements range from around 12% to around 1%, depending on the data set and weighting scheme. For example, the Cranmed data set already achieved a F_1 measure of 0.9933 without adapting the weighting scheme. In this case only marginal improvements are possible. The average improvement over all data sets is about 3.6%, whereas improvement rates for *BM25* vary more than for *tfidf* and *tf* in terms of standard deviation. *Overall, our approach significantly improves classification accuracy on average.*

Convergence: Besides the actual improvement, convergence behavior plays an important role. For all data sets and experiments, we analyzed the convergence behavior of the improvement. It turned out that improvements in F_1 measure differ strongly among the different weighting schemes. For the *BM25* and *tfidf* weighting scheme nearly every iteration increased the F_1 measure for all data sets (see figure 2(b) as an example on the LA1 data set). Since convergence depends strongly on the chosen learning rate, one can argue that *BM25* and *tfidf* weighted document vectors yield to more robust convergence behavior. *tf* weighted examples show a different pictures (see figure 2(a) as an example on the Classic data set). Around half of the data sets show an increase of the F_1 measure after the first few iterations, but decreases then. The behavior could be interpreted as overfitting on the training set. A further explanation would be a too slow annealing of the learning rate, so that training does not converge to a good solution. However, since improvements do not steadily increase, it would be required to select a particular solution. Without using a validation set, this remains problematic since selecting a solution in stochastic on-line optimization is hard to solve in general. Particularly, we could not identify a good heuristic therefore in our task. In terms of number of drawn examples we observe that improvements can be achieved after only doing a few steps. This is important since the number of total examples is quadratic. In particular it seems that similarity calculations among documents yield to a redundant set of training examples for approximating similarities. One reason therefore may be seen in the fact that the number of possible training examples is much larger than the degree of freedom of the optimization problem, i.e. $n^2 \gg |D|$, such that not all examples have to be seen. Hence, stochastic online approaches have an advantage over batch techniques. *Overall, our approach achieves stable convergence at least for* BM25 *and* tfidf *weighted vector spaces.*



(a) Classic dataset

(b) La1 data set

Fig. 2. Convergence rates after every $0.5 * |D|$ randomly drawn examples.

Influence of the weighting scheme : One interesting observation made in our experiments is the difference in accuracy among the different weighting schemes. Surprisingly, *tf* outperforms *idf* on 10 data sets and *BM25* on 12 data sets; *tfidf* outperforms *BM25* also on 12 data sets. In terms of absolute difference, *tf* and *tfidf* outperform *BM25* by a large degree. Hence, ignoring the corpus specific weighting component achieves the best performance on average. Partially stemming and stop word filtering during preprocessing may be responsible for this effect, but does not explain it fully.

However, since our optimization approach improves every weighting scheme, the relative differences among weighting schemes stay the same. Also, those differences do not influence our results and thus can be ignored in our work.

5 Conclusion

In this paper we introduced a new approach to automatically estimate the corpus related component for weighting schemes. Our experiments on 18 data sets showed that our approach (i) can approximate a given weighting scheme rather efficiently using online stochastic gradient descent, (ii) improves classification accuracy in a range between 1% and 12% with an average of 3.6%, (iii) steadily increases improvements on F_1 measures for *BM25* and *tfidf* weighted document vectors and (iv) can achieve significant improvements after drawing only a fraction of all possible $O(n^2)$ examples.

While our results are promising, several issues remain open. The loss function is not optimally suited for the classification case. Here, cross entropy would define a better loss function [13]. Second, stopping criterion's and solution selection are potential problems in our approach. Another interesting extension would be to consider nonlinear transformations by adding weights between different terms that co-occur together in a similarity calculation. Hence, the diagonal matrix L would become a symmetric matrix reflecting term-term relationships.

Future work will address those issues, as well as the application to more complex scenarios like for example learning to rank retrieval results or to integrate a-priori knowledge into content based ranking functions. Moreover, since we can optimize the representation of documents towards an expected similarity measures, it would be possible to map between data set of different characteristics or to adapt data set similarity through visualization.

Acknowledgments

The Know-Center GmbH Graz is funded within the Austrian COMET Program - Competence Centers for Excellent Technologies - under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency FFG.

References

1. Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.
2. S. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *In Proceedings of the Third Text REtrieval Conference (TREC 1994)*, pages 109–126, 1996.

3. Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM.
4. Hui Fang and Chengxiang Zhai. An exploration of axiomatic approaches to information retrieval. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 480–487, New York, NY, USA, 2005. ACM.
5. Donald Metzler and Hugo Zaragoza. Semi-parametric and non-parametric term weighting for information retrieval. In *Proceedings of the 2nd International Conference on the Theory of Information Retrieval (ICTIR 2009)*, 2009.
6. Vo N. Anh and Alistair Moffat. Simplified similarity scoring using term ranks. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 226–233, New York, NY, USA, 2005. ACM.
7. Marco Ernandes, Giovanni Angelini, Marco Gori, Leonardo Rigutini, and Franco Scarselli. Adaptive context-based term (re)weightingan experiment on single-word question answering. *Frontiers in Artificial Intelligence and Applications; Vol. 141*, page 1, 2006.
8. Shai S. Shwartz, Yoram Singer, and Andrew Y. Ng. Online and batch learning of pseudo-metrics. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, pages 94+, New York, NY, USA, 2004. ACM.
9. Christopher D. Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*, volume 8. MIT Press, 2000, 2002.
10. M Granitzer, T Neidhart, and M Lux. Learning term spaces based on visual feedback. In *Proc. 17th International Conference on Database and Expert Systems Applications DEXA '06*, pages 176–180. Ieee, 2006.
11. Léon Bottou. Stochastic learning. In *Advanced Lectures on Machine Learning, Lecture Notes in Artificial Intelligence, LNAI 3176*, pages 146–168, Berlin, 2004. Springer Verlag.
12. Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proc. of CIKM '02*, pages 515–524, McLean, Virginia, 2002.
13. Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK, 1996.