Effective Kernelized Online Learning in Language Processing Tasks

Simone Filice[1], Giuseppe Castellucci[2], Danilo Croce[3] and Roberto Basili[3]

[1]DICII, [2]DIE, [3]DII
University of Roma, Tor Vergata
00133 Roma, Italy
{filice,castellucci}@ing.uniroma2.it
{croce,basili}@info.uniroma2.it

Abstract. Kernel-based methods for NLP tasks have been shown to enable robust and effective learning, although their inherent complexity is manifest also in Online Learning (OL) scenarios, where time and memory usage grows along with the arrival of new examples. A state-of-the-art *budgeted* OL algorithm is here extended to efficiently integrate complex kernels by constraining the overall complexity. Principles of Fairness and Weight Adjustment are applied to mitigate imbalance in data and improve the model stability. Results in Sentiment Analysis in Twitter and Question Classification show that performances very close to the state-of-the-art achieved by batch algorithms can be obtained.

1 Introduction

Given the growing interactivity needed by Web applications, Information Retrieval challenges such as Question Answering (QA) [17], or Sentiment Analysis (SA) [26] over Web or microblog sources [34] are increasingly interesting. In these tasks, as well as in real-time marketing, semantic web-search or exploratory data analysis, the application of Natural Language Processing (NLP) techniques is crucial. Natural Language Learning (NLL) systems deal with the acquisition of models in order to turn texts into meaningful structures. In NLL, traditional and effective paradigms, such as Support Vector Machines or Maximum Entropy models [13], are effectively applied in a *batch* fashion: they require all examples to be available while inducing the model. However, daily interactions are more natural in the Web and suggest to explore dynamic techniques such as Online Learning (OL) [20, 6]. These algorithms, that evolve from the Rosenblatt's Perceptron [27], induce models that can be updated when negative feedback is available, in order to correctly account for new instances and improve accuracy, without complete re-training. In this way, the resulting OL model *follows* its target problem and adapts dynamically. This makes online schemas very appealing in Web scenarios, although performance drops can be observed with respect to the corresponding batch learning algorithms.

The robustness in NLL systems depends also on the suitability of the adopted linguistic features whereas manual encoding is usually carried out by experts.

Kernel methods [32] enable an implicit acquisition of such information. They have been largely employed in NLP, such as in [5, 22, 36, 35, 8], in order to provide statistical models able to separate the problem representation from the learning algorithm. However, kernel methods are not suitable for large datasets, due to time and space complexity. In large margin learning algorithms, classification complexity may prevent kernel adoption in real world applications, as the required time for a single classification depends on the number of *Support Vectors* (SVs) [30]. This led to an unbounded complexity that is in contrast with the online paradigm, as it should provide a (potentially) never-ending learning. An effective way to cope with this problem introduces budgeted-version of such algorithms [4, 24, 33, 10], binding the maximum number of SVs.

In this paper, we adapt and improve the Budgeted Passive Aggressive algorithm [33] in order to provide an effective and efficient way to design NLL systems. We first investigate the idea of *Fairness* in order to balance the importance of different involved classes. Then, we introduce the notion of *Weight Adjustment*, consisting in a consolidation of SVs. It improves the usability of off-the-shelf kernel functions within OL algorithms. From one hand, it allows a fast system design, by avoiding a manual feature engineering. On other hand, we combine the robustness of state-of-the-art kernels with the efficiency of Budgeted OL algorithms. The complexity of the resulting system is thus bounded, allowing to control the computational cost at the best achievable quality. We evaluate this method in two NLP tasks: Sentiment Analysis in Twitter and Question Classification. In both settings, results close to the state-of-the-art are achievable and they are comparable with one of the most efficient SVM implementation. Results are straightforward considering that no manually coded resource (e.g. WordNet or a Polarity Lexicon) has been used. We mainly exploited distributional analysis of unlabeled corpora.

In the rest of the paper, Section 2 describes the Budgeted Online Learning algorithm. In Section 3 employed kernel functions for robust NLL are discussed. In Section 4 experimental evaluation is discussed.

2 Efficient Kernel-based Passive Aggressive Algorithm

The Passive Aggressive (PA) learning algorithm [6] is one of the most popular online approaches and it is generally referred as the state-of-art online method. When an example is misclassified, the model is updated with the hypothesis most similar to the current one, among the set of classification hypotheses that correctly classify the example.

Let $(\mathbf{x}_t, y_t)$ be the t-th example where $\mathbf{x}_t \in \mathbb{R}^d$ is a feature vector in a d-dimensional space and $y_t \in \{+1, -1\}$ is the corresponding label. Let $\mathbf{w}_t \in \mathbb{R}^d$ be the current classification hypothesis. The PA classification function is $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. The learning procedure starts setting $\mathbf{w}_1 = (0, \ldots, 0)$, and after receiving $\mathbf{x}_t$, the new classification function $\mathbf{w}_{t+1}$ is the one that minimizes the following objective function[1] $Q(\mathbf{w})$:

[1] We are referring to the PA-I version in [6]

$$Q(\mathbf{w}) = \frac{1}{2}\left\|\mathbf{w} - \mathbf{w}_t\right\|^2 + C \cdot l(\mathbf{w}; (\mathbf{x}_t, y_t)) \tag{1}$$

where the first term $\|\mathbf{w} - \mathbf{w}_t\|$ is a measure of how much the new hypothesis differs from the old one, while the second term $l(\mathbf{w}, (\mathbf{x}_t, y_t))$ is a proper loss function[2] assigning a penalty cost to an incorrect classification. C is the aggressiveness parameter that balances the two competing terms in Equation 1. Minimizing $Q(\mathbf{w})$ corresponds to solving a constrained optimization problem, whose closed form solution is the following:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{x}_t, \;\; \alpha_t = y_t \cdot \min\left\{C, \frac{H(\mathbf{w}_t; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2}\right\} \tag{2}$$

After a wrong prediction of an example $\mathbf{x}_t$, a new classification function $\mathbf{w}_{t+1}$ is computed. It is the result of a linear combination between the old $\mathbf{w}_t$ and the feature vector $\mathbf{x}_t$. The linear PA is extremely attractive as the classification and the updating steps have a computational complexity of $\mathcal{O}(d)$, i.e. a single dot product in the d-dimensional space. However, these algorithms cannot directly learn non-linear classification functions.

The kernelized version of the PA algorithm overcomes this limitation, and enables to use structured data, like syntactic trees using tree kernel functions [5]. Generic data representations x can be exploited using an implicit mapping $\phi(x)$ into a Reproducing Kernel Hilbert Space $\mathcal{H}$ operated by a proper kernel function $k(\cdot, \cdot)$. When a misclassification occurs, the model is updated to embed the "problematic" example. At step t, the classification function is $f_t(x) = \sum_{i \in SV_t} \alpha_i k(x_i, x)$, while SV_t is the set of the support vector indices. The updating step corresponds to adding a new support vector:

$$f_{t+1}(x) = f_t(x) + \alpha_t k(x_t, \cdot), \;\; \alpha_t = y_t \cdot \min\left\{C, \frac{H(f_t; (x_t, y_t))}{\|x_t\|^2_{\mathcal{H}}}\right\}$$

where $\|\cdot\|_{\mathcal{H}}$ is the norm.

Unfortunately, the kernelized PA algorithm (as any other OL schemas) cannot be used against large datasets. The growth of SVs has a significant drawback in terms of computational complexity and memory usage. All the SVs and their weights must be stored, and each new prediction requires the computation of the kernel function between the example and all current SVs. Several works dealt with such issue, as [4, 33, 10]. In particular, [33] proposed a budgeted version of the PA algorithm that strictly binds the number of support vectors to a predefined budget B. In their proposal, the updating step of the PA algorithm changes when B is reached: the optimal solution of the PA problem f^* cannot be employed as it would require $B + 1$ support vectors; thus the best estimation of f^* in the space spanned by B support vectors must be chosen. Before adding a new support vector, an old r one must be removed, obtaining a new function f^r_{t+1}. Thus, a new constraint to the minimization problem $Q(\mathbf{w})$ is considered, resulting in:

[2] In this work we will consider the hinge loss $H(\mathbf{w}; (\mathbf{x}_t, y_t)) = max(0, 1 - y_t \mathbf{w}^T \mathbf{x}_t)$

$$
\begin{aligned}
& f^r_{t+1} = \underset{f^r}{\operatorname{argmin}} \tfrac{1}{2} \left\| f^r - f_t \right\|^2_{\mathcal{H}} + C \cdot \xi \\
& s.t. \;\; 1 - y_t \mathbf{w}^T \mathbf{x}_t \leq \xi, \; \xi \geq 0 \\
& \qquad f^r = f_t - \underbrace{\alpha_r k(x_r, \cdot)}_{SV\ elimination} + \underbrace{\sum_{i \in V} \beta_i k(x_i, \cdot)}_{weights\ modification}
\end{aligned} \tag{3}
$$

where r is the index of the support vector to be removed, t is the index of the new support vector to be added and $V \subseteq SV_t \cup \{t\} - \{r\}$ is the set of the indices of SVs whose weights can be modified. Let f^r_{t+1} be the solution of Equation 3 for a given r; using a brute force approach, the support vector to be removed r^* and the corresponding new classification function f_{t+1} are selected as the ones that minimize Equation 1: $r^* = \underset{r \in SV_t \cup \{t\}}{\operatorname{argmin}} Q(f^r_{t+1})$. In [33] f^r_{t+1} has been solved in closed-form and the overall updating complexity is shown[3] to be $\mathcal{O}(B|V|^2)$. The choice of V (i.e. the set of SVs whose weights can be modified) has a deep impact on the computational complexity of this step. Three different Budgeted Passive Aggressive (BPA) policies are proposed:

- BPA-Simple (BPA-S): $V = \{t\}$
- BPA-Projecting (BPA-P): $V = SV_t \cup \{t\} - \{r\}$
- BPA-Nearest-Neighbor (BPA-NN): $V = \{t\} \cup NN(r)$ where $NN(r)$ is the index of the nearest neighbor of x_r.

The third one is shown to be a good trade-off between the accuracy and computational complexity. The main idea is to preserve the information provided by the support vector r to be removed, by projecting it into the space spanned by the support vectors in V. The space spanned by r is similar to the one spanned by its nearest neighbor, so the BPA-NN is supposed to provide good performances.

2.1 Improving robustness through Fairness and Weight Adjustment

The original formulation of the PA algorithm makes no distinction between positive and negative examples. In many real-world classification domains, examples are not equally distributed among the classes. Thus, very imbalanced datasets can penalize less frequent classes. A common solution is data sampling [31] in order to obtain a more balanced distribution. However, these methods discard some informative examples. A different approach directly modifies the learning algorithm and emphasizes the contribution of the less frequent examples. In [21] the original SVM formulation is slightly modified splitting the empirical risk term into a positive part and a negative one. Each is weighted by parameters C_+ and C_- that substitute the original regularization parameter C: in this way it is possible to adjust the cost of false positives vs. false negatives. Accordingly, we reformulated the original objective function:

$$
\begin{aligned}
& \underset{\mathbf{w}}{\operatorname{argmin}} \tfrac{1}{2} \left\| \mathbf{w} - \mathbf{w}_t \right\|^2 + C(y_t) \cdot \xi \\
& s.t.\ 1 - y_t \mathbf{w}^T \mathbf{x}_t \leq \xi, \; \xi \geq 0, \text{where } C(y_t) = \begin{cases} C_+ \text{ if } y_t = +1 \\ C_- \text{ if } y_t = -1 \end{cases}
\end{aligned}
$$

[3] It applies when kernel computations are cached.

The parameters C_+ and C_- can be chosen so that the potential total cost of the false positives equals the potential total cost of the false negatives, i.e. the ratio C_+/C_- is equal to the ratio between the overall number of negative and positive training examples. It allows to introduce a sort of **fairness** in the learning phase. During the experimental evaluation we will refer to the experiment settings as $fair$ BPA (F-BPA) if the aggressiveness parameters are chosen as the above ratio, and as $unfair$ BPA if $C_+ = C_-$.

Another drawback of OL algorithms is that generally they do not perform as good as batch learning ones, due to their inherent simplicity. For instance, the updating step of the PA algorithm performs a local optimization that considers a single example at a time. Batch learning algorithms, e.g. SVM, perform a global optimization over the whole training set. Changing the model at each misclassification has two consequences in OL: from one side, it enables the algorithm to track a shifting concept; on the other side, it can produce instability in the learning process. Outliers or mislabeled examples can affect the current model and different orders of the same training examples can produce quite different solutions. The PA approach mitigates this instability modifying the current classification hypothesis as less as possible, but it does not completely solve this issue. A common solution is performing multiple iterations on the training data. A new iteration (i.e. epoch) involves the computation of new kernel operations that have not been evaluated during the first epoch, thus losing the OL advantages. In fact, let SV_t be the set of the indices of the current SVs and let x_t be the current training example such that $t \notin SV_t$ (i.e. x_t is not a support vector); then, during the second epoch, for each support vector x_i such that $i > t$, a new kernel computation $k(x_i, x_t)$ must be computed. In order to overcome these drawbacks, we propose an effective variation to the standard multiple iteration method. Instead of revisiting the whole training set, only the support vector set is exploited again. We will refer to this approach as **weight adjustment** to emphasize the fact that the support vector set does not change, and only the weights α_i of the support vectors can be adjusted changing their contribution to the classification function. From a computational perspective, caching the kernel operation of the first epoch, any additional computation must be performed. We will indicate as A-BPA a standard Budgeted Passive Aggressive Algorithms which performs a weight adjustment, and AF-BPA is the corresponding fair version.

3 Combining Semantic Kernels

Combination of classifiers is commonly used to improve the performances joining the strengths of many methods. Another strategy is to combine similarity functions among different representations in a kernel combination. It is still a valid kernel and can thus be integrated in learning algorithms [30]. We decided to apply three kernel functions, each emphasizing a specific aspect.

Bag of Word Kernel ($BOWK$) A first kernel function exploits pure lexical information, expressed as the word overlap between texts. It is very common in Information Retrieval, since [29], where documents are represented as vectors

whose dimensions correspond to different terms. A boolean weighting is applied: each dimension represents an indicator of the presence or not of a word in the text. The kernel function is the cosine similarity between vector pairs.
Lexical Semantic Kernel (LSK) A kernel function generalizes the lexical information, without exploiting any manually coded resource. Lexical information is obtained by a co-occurrence Word Space built accordingly to the methodology described in [28]. First, a word-by-context matrix M is computed through a large scale corpus analysis. Then, *Latent Semantic Analysis* [18] technique is applied as follows. The matrix M is decomposed through Singular Value Decomposition (SVD). The original statistical information about M is captured by the new k-dimensional space, which preserves the global structure while removing low-variant dimensions, i.e. distribution noise. Thus, every word is projected in the reduced space and a sentence is represented by applying a *linear combination*. The resulting kernel is the cosine similarity between vector pairs, as in [7].
Smoothed Partial Tree Kernel (*SPTK*) Tree kernels exploit syntactic similarity through the idea of convolutions among substructures. Any tree kernel evaluates the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space [5]. Its general equation is $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$ where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes respectively, and $\Delta(n_1, n_2)$ is equal to the number of common fragments rooted in the n_1 and n_2 nodes. In the SPTK formulation [8] the function Δ emphasizes lexical nodes. SPTK main characteristic is its ability to measure the similarity between syntactic tree structures, which are partially similar and whose nodes can differ but are semantically related. One of the most important outcomes is that SPTK allows "embedding" external lexical information in the kernel function only through a similarity function σ among lexical nodes, namely words. Such lexical information can be automatically acquired though a distributional analysis of texts. The $\sigma(n_1, n_2)$ function between lexical item is measured within a Word Space. As in [8] the Grammatical Relation Centered Tree[4] (GRCT) is used to represent the syntactic information. Grammatical Relations are central nodes from which dependencies are drawn and all the other features of the central node, i.e. lexical surface form and its POS-Tag, are added as additional children.

A kernel combination αBOWK + βLSK + γSPTK combines the lexical properties of BOWK (generalized by LSK) and the syntactic information captured by the SPTK.

4 Experimental Evaluations

Experimental evaluations over the Twitter Sentiment Analysis [34] and Question Classification [36] tasks are described. We aim at verifying that: (i) the combination of the proposed online learning schema with *Fairness* and *Weight*

[4] Notice that in [8] other trees are presented. In this work we selected the GRCT as it provides the most explicit syntactic information.

Adjustment techniques achieves results comparable with batch ones; (ii) the number of kernel computation can be reduced with reasonable budgets.

4.1 Sentiment Analysis in Twitter

Web 2.0 and Social Network technologies allow users to generate contents on blogs, forums and new forms of communication (such as micro-blogging) writing their opinion about facts, things, events. Twitter[5] represents an intriguing source of information as it is used to share opinions and sentiments about brands, products, or situations [14]. Tweet analysis represents a challenging task for Data Mining and Natural Language Processing systems: tweets are short, informal and characterized by their own particular language.

When applied to tweets, traditional approaches to Sentiment Analysis [26] show significant performance drops as they use to focus on larger and well written texts, e.g. product reviews. Some recent works tried to model the sentiment in tweets [25, 16, 9, 1]. Specific approaches are used, such as hand coded resources and artificial feature modeling, in order to achieve good accuracy levels. To assure a fast deployment of robust systems, our approach exploits kernel formulations to enable learning algorithms to extract the useful information. We applied a Multi-Kernel approach considering the BOWK and LSK above described. The Word Space used within the LSK is acquired through the analysis of a generic corpus made of 3 million of tweets. We built a matrix M, whose rows are vectors of pairs $\langle$lemma, POS$\rangle$ of the downloaded tweets. Columns of M are the contexts of such words in a short window ($[-3, +3]$) to capture paradigmatic relations. The most frequent 10,000 items are selected along with their left and right 20k contexts. Point-wise-mutual information is used to score M entries. Finally, the SVD reduction is applied, with a dimensionality cut of $k = 250$. We used only lexical information, as the low quality of parse trees of tweets [11] would compromise the Tree Kernel contribution.

A pre-processing stage is applied to reduce data sparseness thus improving the generalization capability of learning algorithms. In particular, a normalization step is applied on the text: fully capitalized words are converted in lowercase; reply marks are replaced with the pseudo-token `USER`, hyperlinks by `LINK`, *hashtags* by `HASHTAG` and emoticons by special tokens[6]; any character repeated more than three times are normalized (e.g. "*nooo!!!!!*" is converted into "*noo!!*"). Then, an almost standard NLP syntactic processing chain [3] is applied. Evaluation is carried out on the SemEval-2013 Task 2 corpus [34]. In particular, we focus on the *Message Polarity Classification*: it deals with the classification of a tweet with respect to three classes *positive*, *negative* and *neutral*. Training dataset is composed by $10,205$ annotated examples, while test dataset is made of $3,813$ examples[7]. Classifier parameters are tuned with a Repeated Random

[5] `http://www.twitter.com`

[6] We normalized 113 well-known emoticons in 13 classes.

[7] Training dataset is made of $3,786$ positive, $1,592$ negative and $4,827$ neutral examples; test dataset is made of $1,572$ positive, 601 negative and $1,640$ neutral.

Sub-sampling Validation, consisting in a 10-fold validation strategy on a subset of the training data split according to a 70%-30% proportion. As results may depend on the order of training data, 10 different models are acquired on 10 shuffled training data.

Table 1: Sentiment Analysis in Twitter Results. *Saving* is the percentage of kernel computations avoided with respect to a SVM classifier implemented in SvmLight [15]. SVM requires 141 million of kernel computations, achieving 0.654 F1. Notice that BPA/AF-BPA *Saving* is the same, as Adjustment does not add any new kernel computation.

Results									Saving	
Budg.	BPA				AF-BPA				BPA	AF-BPA
	BOW_{lin}	LSK_{lin}	BOW_{lin} + LSK_{lin}	BOW_{lin} + LSK_{rbf}	BOW_{lin}	LSK_{lin}	BOW_{lin} + LSK_{lin}	BOW_{lin} + LSK_{rbf}	BOW_{lin} + LSK_{rbf}	
100	.396±.04	.451±.04	.444±.08	.430±.06	.394±.03	.346±.07	.456±.06	**.471±.03**	98%	98%
250	.408±.03	.469±.03	.462±.06	.444±.06	.418±.03	.454±.06	**.505±.02**	.438±.06	95%	95%
500	.443±.02	.480±.02	.513±.03	**.516±.04**	.412±.04	.459±.06	.503±.05	**.516±.04**	89%	89%
750	.448±.03	.495±.03	.500±.04	.503±.06	.426±.04	.451±.06	**.542±.04**	.527±.03	84%	84%
1000	.446±.02	.488±.02	.517±.03	.525±.03	.475±.02	.511±.04	.542±.03	**.552±.02**	80%	80%
1500	.448±.04	.496±.04	.516±.04	.506±.05	.471±.02	.517±.04	**.555±.02**	.591±.03	71%	70%
2000	.463±.03	.494±.03	.530±.04	.520±.05	.490±.03	.564±.02	**.585±.02**	.572±.02	62%	62%
2500	.478±.03	.501±.03	.525±.04	.532±.04	.498±.03	.585±.01	**.599±.01**	.595±.03	55%	54%
3000	.484±.02	.496±.02	.548±.05	.544±.05	.541±.02	.582±.01	.600±.02	**.605±.03**	49%	47%
3500	.496±.01	.521±.01	.546±.05	.568±.04	.548±.01	.574±.01	.609±.01	**.619±.01**	43%	41%
4000	.496±.03	.513±.03	.559±.05	.572±.04	.556±.02	.573±.01	.621±.01	**.623±.01**	38%	35%
4500	.504±.01	.538±.01	.564±.05	.574±.05	.569±.02	.573±.02	.618±.02	**.628±.01**	35%	30%
5000	.505±.02	.535±.02	.572±.04	.579±.04	.576±.01	.574±.01	**.610±.01**	.604±.02	32%	24%
6000	.505±.02	.543±.02	.581±.04	.593±.04	.545±.01	.573±.01	.626±.01	**.633±.01**	28%	18%
7000	.510±.02	.548±.02	.580±.04	.595±.04	.576±.01	.572±.01	.634±.01	**.640±.01**	28%	14%
8000	.516±.02	.550±.02	.580±.04	.597±.04	.575±.01	.574±.01	.635±.01	**.643±.01**	28%	14%
9000	.516±.02	.550±.02	.580±.04	.597±.04	.578±.01	.574±.01	.636±.01	**.643±.01**	28%	14%
10000	.516±.02	.550±.02	.580±.04	.597±.04	.577±.01	.575±.01	.636±.01	**.642±.01**	28%	14%
-	.516±.02	.550±.02	.580±.04	.597±.04	.577±.01	.575±.01	.635±.01	**.643±.01**	28%	14%

In Table 1 results of different kernels and different budget values are reported. We investigated the contribution of a linear kernel on top of BOWK (BOW_{lin}), and latent representation (LSK_{lin}) as well as their linear combination ($BOW_{lin} + LSK_{lin}$). Moreover, a combination considering a gaussian kernel over the latent semantic representation is investigated ($BOW_{lin} + LSK_{rbf}$). Results are reported in terms of mean F1-Measure between the positive and negative classes, consistently with results reported in [34]: the same metric used in SemEval challenge. 10 different models have been trained by shuffling the training set and the result mean is reported. We report also the standard deviation of the F1 to verify the stability of the proposed solution with respect to training order. The standard BPA and the AF-BPA learning are here considered. All kernels benefits from the adoption of AF-BPA in terms of pure performance, i.e. the mean F1-Measure improves of 5 F1 points. For almost all budget values the standard deviation is lower, so reducing the dependence on training order. Moreover, AF-BPA overtakes best results of BPA at low budget levels, i.e. 3000. Due to lack of space, A-BPA and F-BPA results are not reported; however, their performance is between the BPA and the AF-BPA, demonstrating that both Fairness and Weight Adjustment individually have a valuable contribution. As a comparison, we trained a SVM classifier: it achieves 0.654 F1 after about 141

millions of kernel computations[8]. The AF-BPA achieves 0.643 F1, with a performance drop of only 1.6%. It is straightforward considering that all advantages of online learning are preserved. At $B = 4000$ computational cost is reduced, saving about 35% kernel computations with respect to SVM, achieving 0,623 F1.

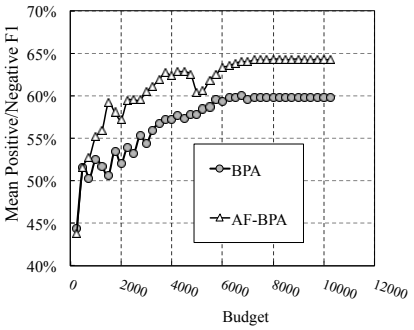

Fig. 1: Mean Positive/Negative F1

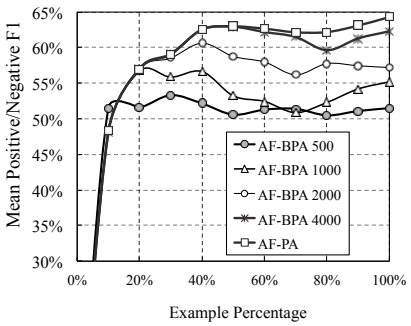

Fig. 2: Learning curve

Performances of AF-BPA and BPA are shown in Figure 1 where the improvement of the proposed method is noticeable since low budgets. We also report the generalization capability in poor training conditions, summarized by the learning curve in Figure 2. Performance at $B = 4000$ is close to the unbudgeted version at different training set size. Results are promising even with smaller budgets, e.g. $B = 1000$. The lack of performance drop (when the budget is "full") is important here, as it suggests that the model would be stable through time, adapting itself to shifting concepts. Finally, the AF-BPA and the used kernels are competitive with respect to state-of-the art systems that participated in SemEval-2013 Task [34]. The AF-BPA would have ranked 4^{th} over 36 systems, with respect to systems trained with the same conditions, i.e. not using external annotated material. It is straightforward, considering that better systems in the challenge used hand-made lexicon [34]. Limiting the number the support vectors to $B = 4000$ the AF-BPA would have ranked 7^th.

4.2 Question Classification

Question Classification (QC) is usually applied in Question Answering systems to map the question into one of k classes of answers, thus constraining the search. In these experiments, we used the UIUC dataset [19]. It is composed by a training set of 5,452 and a test set of 500 questions[9], organized in 6 classes (like `ENTITY` or `HUMAN`). It has been already shown the contribution of (structured)

[8] We compare SVM and OL by the number of kernel computations, as the computational cost of the learning algorithm itself is negligible if compared to novel kernel computations. More details on specific kernels in [7, 8].

[9] http://cogcomp.cs.illinois.edu/Data/QA/QC/

kernel based learning within batch algorithms for this task, since [36, 23]. Here, we want to study an online setting with a kernel combination αBOWK + βLSK + γSPTK to robustly combine lexical and syntactic information[10]. Lexical similarity within LSK and SPTK is derived through the distributional analysis on the UkWaC [2] corpus. The same classifier and Word Space settings of the Sentiment Analysis scenario is here applied. In Table 2 results are reported in terms of Accuracy, i.e. the percentage of test examples obtaining a correct labeling; it is the most common metric adopted in QC. Again, the mean Accuracy is reported for 10 different training shuffles. Standard deviation as indicator of the robustness of our solution is reported. Here, traditional BPA, the contribution of Weight Adjustment (A-BPA), Fairness (F-BPA) and both (AF-BPA) are the models compared.

Table 2: Question Classification Results. *Saving* is again relative to SvmLight [15]. SVM requires 40 million of kernel computations, achieving 93,6% accuracy. Notice that BPA/A-BPA and F-BPA/AF-BPA *Savings* are the same, as Adjustment does not add any new kernel computation.

Budget	Standard			withFairness		
	BPA	A-BPA	Saving	F-BPA	AF-BPA	Saving
100	**79,2%±3,9%**	73,3%±4,9%	92%	75,5%±5,6%	64,1%±5,7%	92%
250	**82,1%±2,4%**	74,4%±8,6%	81%	80,3%±3,0%	79,2%±5,9%	81%
500	86,9%±1,9%	82,5%±4,0%	66%	**87,5%±1,7%**	83,4%±4,4%	66%
750	87,3%±2,3%	84,8%±3,8%	53%	**88,7%±1,7%**	87,9%±1,3%	53%
1000	86,2%±3,8%	86,9%±2,4%	44%	88,3%±1,5%	**88,5%±1,1%**	42%
1500	87,4%±2,9%	87,5%±1,4%	32%	89,6%±0,5%	**90,8%±0,6%**	30%
2000	88,1%±2,4%	88,1%±1,7%	29%	90,1%±0,6%	**91,1%±0,5%**	24%
3000	87,9%±2,1%	88,2%±1,7%	28%	90,1%±0,5%	**90,8%±0,4%**	21%
4250	87,9%±2,1%	88,2%±1,7%	28%	90,1%±0,5%	**90,8%±0,5%**	21%
-	87,9%±2,1%	88,2%±1,7%	28%	90,1%±0,5%	**90,9%±0,4%**	21%

Results confirm that Fairness improves the discrimination between the 6 classes by an absolute 3%, even if a significant imbalance is faced[11]. Accuracy also benefits from the adoption of Weight Adjustment, as shown by the improvements of A-BPA and AF-BPA. Adjustment is also useful to reduce standard deviation, especially with respect to standard BPA. Moreover, AF-BPA reaches better results at low budget levels, i.e. 750. As a comparison, we trained a SVM classifier, which achieves a 93,6% after about 40 millions of kernel computations. The AF-BPA shows only a 3% accuracy drop with a 30% reduction of kernel computations, and a 6% with 53% reduction, i.e. only $B = 750$. When Adjustment and Fairness are applied within extremely low budgets, performance drops. Fairness tends to assign higher weights to less represented classes: when only few examples can be stored, the classifier tends to prefer such classes, increasing the overall error probability. Weight Adjustment instead leads to data over-fitting acting on few Support Vectors and to worse results.

[10] In our experiments, α, β and γ are set to 1

[11] Examples in `Entity` class are 1250 while `Abbreviation` are 86.

5 Conclusion

In this paper, an extension of the Budgeted Passive Aggressive Algorithm is proposed to enable robust and efficient Natural Language Learning processes based on semantic kernels. The proposed principles of Fairness and Weight Adjustment obtain results that are close to the state-of-the-art achieved by batch versions of the same algorithm. This confirms that the proposed OL learner can be applied to real world linguistic tasks. Opinions and sentiments in fact change over time and are strongly language-driven resulting as highly volatile and dynamic. Batch learning is unsuited in these cases as full re-training would be infeasible most of the times. The ability to continuously learn binding the overall complexity, coupled with the Weight Adjustment stage, allows to preserve accuracy by saving in computational costs: in Question Classification we observed only a 6% drop with a 53% saving with respect to SVM. Future work will address concept shifts against ad-hoc datasets. Moreover, extension to other Kernels, or Kernel Combinations, is foreseen [12].

References

1. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., Passonneau, R.: Sentiment analysis of twitter data. In: Proceedings of LASM. pp. 30–38 (2011)
2. Baroni, M., Bernardini, S., Ferraresi, A., Zanchetta, E.: The wacky wide web: a collection of very large linguistically processed web-crawled corpora. Language Resources and Evaluation 43(3), 209–226 (2009)
3. Basili, R., Zanzotto, F.M.: Parsing engineering and empirical robustness. Nat. Lang. Eng. 8(3), 97–120 (Jun 2002)
4. Cesa-Bianchi, N., Gentile, C.: Tracking the best hyperplane with a simple budget perceptron. In: In proc. of the nineteenth annual conference on Computational Learning Theory. pp. 483–498. Springer-Verlag (2006)
5. Collins, M., Duffy, N.: Convolution kernels for natural language. In: Proceedings of Neural Information Processing Systems (NIPS'2001). pp. 625–632 (2001)
6. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research 7, 551–585 (Dec 2006)
7. Cristianini, N., Shawe-Taylor, J., Lodhi, H.: Latent semantic kernels. J. Intell. Inf. Syst. 18(2-3), 127–152 (2002)
8. Croce, D., Moschitti, A., Basili, R.: Structured lexical similarity via convolution kernels on dependency trees. In: Proceedings of EMNLP. Scotland, UK. (2011)
9. Davidov, D., Tsur, O., Rappoport, A.: Enhanced sentiment learning using twitter hashtags and smileys. In: COLING. pp. 241–249 (2010)
10. Dekel, O., Shalev-Shwartz, S., Singer, Y.: The forgetron: A kernel-based perceptron on a budget. SIAM J. Comput. 37(5), 1342–1372 (Jan 2008)
11. Foster, J., Çetinoglu, Ö., Wagner, J., Roux, J.L., Hogan, S., Nivre, J., Hogan, D., van Genabith, J.: #hardtoparse: Pos tagging and parsing the twitterverse. In: Analyzing Microtext (2011)
12. Gönen, M., Alpaydin, E.: Multiple kernel learning algorithms. Journal of Machine Learning Research 12, 2211–2268 (2011)
13. Jaakkola, T., Meila, M., Jebara, T.: Maximum entropy discrimination. In: Solla, S.A., Leen, T.K., Müller, K.R. (eds.) NIPS. pp. 470–476. The MIT Press (1999)

14. Jansen, B.J., Zhang, M., Sobel, K., Chowdury, A.: Twitter power: Tweets as electronic word of mouth. J. Am. Soc. Inf. Sci. Technol. 60(11), 2169–2188 (Nov 2009)
15. Joachims, T.: Learning to Classify Text Using Support Vector Machines. Kluwer Academic Publishers (2002)
16. Kouloumpis, E., Wilson, T., Moore, J.: Twitter sentiment analysis: The good the bad and the omg! In: ICWSM (2011)
17. Kwok, C.C., Etzioni, O., Weld, D.S.: Scaling question answering to the web. In: World Wide Web. pp. 150–161 (2001)
18. Landauer, T., Dumais, S.: A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. Psychological Review 104 (1997)
19. Li, X., Roth, D.: Learning question classifiers: the role of semantic information. Natural Language Engineering 12(3), 229–249 (2006)
20. Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. In: Machine Learning. pp. 285–318 (1988)
21. Morik, K., Brockhausen, P., Joachims, T.: Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In: ICML. pp. 268–277. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
22. Moschitti, A., Pighin, D., Basili, R.: Tree kernels for semantic role labeling. Computational Linguistics 34 (2008)
23. Moschitti, A., Quarteroni, S., Basili, R., Manandhar, S.: Exploiting syntactic and shallow semantic kernels for question/answer classification. In: Proceedings of ACL'07 (2007)
24. Orabona, F., Keshet, J., Caputo, B.: The projectron: a bounded kernel-based perceptron. In: Proceedings of ICML '08. pp. 720–727. ACM, USA (2008)
25. Pak, A., Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining. In: LREC (2010)
26. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retr. 2(1-2), 1–135 (Jan 2008)
27. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65(6), 386–408 (1958)
28. Sahlgren, M.: The Word-Space Model. Ph.D. thesis, Stockholm University (2006)
29. Salton, G., Wong, A., Yang, C.: A vector space model for automatic indexing. Communications of the ACM 18 (1975)
30. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA (2004)
31. Van Hulse, J., Khoshgoftaar, T.M., Napolitano, A.: Experimental perspectives on learning from imbalanced data. In: Proceedings of the ICML. ACM, USA (2007)
32. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience (1998)
33. Wang, Z., Vucetic, S.: Online passive-aggressive algorithms on a budget. Journal of Machine Learning Research - Proceedings Track 9, 908–915 (2010)
34. Wilson, T., Kozareva, Z., Nakov, P., Ritter, A., Rosenthal, S., Stoyonov, V.: Semeval-2013 task 2: Sentiment analysis in twitter. In: Proceedings of the 7th International Workshop on Semantic Evaluation (2013)
35. Zanzotto, F.M., Pennacchiotti, M., Moschitti, A.: A machine learning approach to textual entailment recognition. Natural Language Engineering 15-04 (2009)
36. Zhang, D., Lee, W.S.: Question classification using support vector machines. In: Proceedings of SIGIR '03. pp. 26–32. ACM, New York, NY, USA (2003)