Applying Multiclass Bandit algorithms to call-type classification

Liva Ralaivola[1], Benoit Favre[1], Pierre Gotab[2], Frederic Bechet[1], Geraldine Damnati[3]

[1]*Aix Marseille Universite - LIF/CNRS - Marseille, France*
{liva.ralaivola,benoit.favre,frederic.bechet}@lif.univ-mrs.fr
[2]*Universite d'Avignon - LIA/CERI - Avignon, France*
pierre.gotab@univ-avignon.fr
[3]*France Telecom - Orange Labs, Lannion, France*
geraldine.damnati@orange-ftgroup.com

***Abstract*—We analyze the problem of call-type classification using data that is *weakly labelled*. The training data is not systematically annotated, but we consider we have a *weak* or *lazy oracle* able to answer the question "Is sample x of class q?" by a simple 'yes' or 'no' answer. This situation of learning might be encountered in many real-world problems where the cost of labelling data is very high. We prove that it is possible to learn linear classifiers in this setting, by estimating adequate expectations inspired by the Multiclass Bandit paradgim. We propose a learning strategy that builds on Kessler's construction to learn multiclass perceptrons. We test our learning procedure against two real-world datasets from spoken langage understanding and provide compelling results.**

I. Introduction

Multiclass Bandit algorithms correspond to online classification algorithms with *bandit* or partial feedback: only a binary feedback (positive or negative) is given at each trial of online learning. The true class label for a given instance remains unknown if the feedback is negative. This problem is related to the multi-armed bandit problem originally proposed by [1]: a gambler plays with a slot-machine with K arms. At each time step, he has to choose one arm to pull and he receives a reward corresponding to the arm chosen. The gambler's purpose is to maximize his return over a sequence of pulls. Each arm delivers rewards that are independently drawn from a fixed and unknown distribution so the goal of the gambler is to find the arm with the highest expected reward as early as possible. At each step in the game the player must decide if he wants to explore a new arm or exploit the knowledge acquired to choose the best arm known so far. This problem is an illustration of the exploration vs. exploitation trade-off studied in reinforcement learning algorithms.

Multi-armed bandit algorithms have been widely studied in the Machine Learning community and recently applied to multiclass online learning (see, e.g., [2], [3], [4], [5], [6]). In multiclass bandit algorithms the feature vectors of the training examples are considered as *side information* given to the gambler to help him choose the next arm to pull — these bandit algorithms using side information may as well be termed *contextual bandits*.

The goal of this paper is to apply the multiclass bandit paradigm to Spoken Dialog Systems (SDS) through the online adaptation of Call-Routing classification models. In this setting the whole dialog system is considered as a slot machine with K arms, each arm corresponding to a call-type to predict. For each spoken request expressed by a caller (the *side information*), the system (considered here as the *gambler*) will pull one arm by proposing a call-type to the caller and asking him for a confirmation. The answer to this confirmation can be positive or negative and corresponds here to the *bandit* or partial feedback. For each request the system can choose a random call-type (*exploration*) or the most probable call-type estimated by the current classification models on the caller's request (*exploitation*).

Two algorithms are compared in this study on two call-routing tasks developed at Orange Labs: firstly the online learning *Banditron* algorithm proposed by [4]; secondly a novel batch learning algorithm, closely related to the Banditron, that we believe is more adequate for the rapid adaptation of Spoken Dialog systems.

More formally, this new algorithm is a linear multiclass perceptron-based classification algorithm that is capable of learning from *lazily labelled* data. Lazy labels are obtained thanks to a *lazy oracle* which, given a query pair of the form $(\boldsymbol{x}, q)$ where $\boldsymbol{x}$ is an input data and q is from a set of predefined labels $\mathcal{Q} = \{1, \ldots, Q\}$ merely answers *yes* or *no* depending on whether q is the correct label for $\boldsymbol{x}$ or not. In other words, when hypothesizing a label q for the example $\boldsymbol{x}$, it is possible to query the *lazy oracle* $\tilde{y}_q(\boldsymbol{x}) \in \{0, 1\}$ which will say yes (1) if q is indeed the correct label or no (0) if it is not. In the case its answer is *no*, the real label of $\boldsymbol{x}$ is still unknown, the only thing we learn is that the correct label is not q. In this work, each example can only be queried once but one could imagine multiple queries on the same example leading eventually to the full labeling, at the cost of calling the oracle several times.

In the following, we will take advantage of a *uniform random query scheme* that, given $\boldsymbol{x}$, issues the query $(\boldsymbol{x}, q)$ with probability $1/Q$, for all $q \in \mathcal{Q}$. The learning algorithms that we propose extend multiclass Perceptron algorithms that

have been analyzed by [7]. The strategy that we develop exploits the peculiarity of the random query scheme that is implemented and it relies on computing empirical estimates of *update vectors* whose expectations are known to correspond to misclassified data.

The paper is organized as follows. In the following section, we describe some related work done on reducing the need for human supervision in the training of SDS models. Section III presents more formally the problem and introduce the notations that are going to be used to present our approach in section IV. Finally section V presents experiments and results performed on two call-type classification corpora collected at Orange Labs.

II. Related Work

Reducing the need for manual transcription and annotation data is the key for the rapid deployment of statistical SDS on a large scale. From a machine learning point of view, this can be seen as studying *weakly* or *partially* supervised methods for training and updating ASR and SLU models. Most of the weakly supervised methods applied to SLU are based on the *active learning* paradigm [8].

Regardless of the method used, the active learning paradigm always needs a *Full Oracle* to obtain the true labels on the portion of the raw data selected. In the context of SDS a *weak* supervision can be provided by the caller using the system: by analyzing the dialog structure and detecting implicit or explicit confirmations and corrections, by analyzing system logs, we can collect clues about the correctness of the semantic hypotheses proposed by the system. Such a process has been proposed in [9] and was named *Implicitly-supervised Learning* by the authors. It was used in order to automatically train a confidence annotation process. In [10] the confirmation prompts of the users of a call-routing spoken dialog system are used as a *partial Oracle* for directly improving the classification accuracy of a set of n binary classifiers (one for each call-type).

In all these previous studies a bootstrap corpus, fully annotated, containing examples of all the expected labels is needed in order to train a first classifier producing the hypotheses given to the partial oracle. The online adaptation process only performs *exploitation* using the current classification models. The goal of this study is to adapt the Multiclass Bandit paradigm to call-type classification in order to perform *exploration* as well as *exploitation* during the adaptation process. We will first use the *Banditron* algorithm proposed by [4]. This algorithm is based on a perceptron adapted to handle the case of partial feedback where it is possible to manually set the tradeoff between exploration and exploitation.

In this study, we want to push forward this paradigm, both from a theoretical and practical point of view, by implementing an exploration strategy based on a uniform random query scheme that does not need any bootstrap corpus and which can learn only from this weak supervision.

III. Learning from Lazy Labels

This section provides a formal description of the problem we are interested in.

A. General

From here on, the notation we introduce holds throughout the paper. As already mentioned, we are addressing a multiclass learning task. This task is defined over the product space $\mathbb{R}^d \times \mathcal{Q}$, where $d < \infty$ is the number of features that describe our data $\boldsymbol{x} \in \mathcal{X}$ and $\mathcal{Q} = \{1, \ldots, Q\}$ is the set of Q categories or classes of interest. We assume that there exists a (deterministic) labelling function $y : \mathbb{R}^d \to \mathcal{Q}$ such that $y(\boldsymbol{x})$ denotes the class of $\boldsymbol{x}$. We also assume that there exists a *fixed* but *unknown* distribution D over $\mathbb{R}^d$ such that every vector $\boldsymbol{x}$ or $\boldsymbol{x}^n$ is an independent random vector with distribution D.

We consider three different settings: the usual multiclass learning setting (full oracle), the online bandit setting in which each prediction of a classifier is revealed as correct or incorrect in sequence by a lazy oracle and a batch derivative of the bandit setting in which a batch of examples is randomly queried to the lazy oracle. In the following, we explain more formally how this batch bandit setting is derived.

B. Partial or Lazy Label Information

Our goal is to propose a learning procedure capable of building a classifier from weakly or *lazily* labelled data Contrarily to what is commonly encountered in supervised learning scenarios, lazily labelled data are not of the form $\mathcal{S} = \{(\boldsymbol{x}^n, y(\boldsymbol{x}^n))\}_{n=1}^N$ but of the form $\widetilde{\mathcal{S}} = \{(\boldsymbol{x}^n, \tilde{\boldsymbol{y}}^n)\}_{n=1}^N$, where $\tilde{\boldsymbol{y}}^n = (\tilde{y}_q^n)_{q \in \mathcal{Q}}$ is a Q-dimensional vector with $\tilde{y}_q^n \in \{1, 0, \perp\}$, for $q \in \mathcal{Q}$; $\tilde{\boldsymbol{y}}^n$ carries the following information:

- if $\tilde{y}_q^n = 1$, then $\boldsymbol{x}^n$ is of class q;
- if $\tilde{y}_q^n = 0$, then $\boldsymbol{x}^n$ is *not* of class q;
- if $\tilde{y}_q^n = \perp$, then we *do not know* whether $\boldsymbol{x}^n$ is of class q or not.

The label information carried by $\tilde{\boldsymbol{y}}^n$ is only partial whenever none of the $\tilde{y}_q^n$'s is equal to 1. In the following, we show how such partial or *lazy* label $\tilde{\boldsymbol{y}}^n$ can be obtained from what we call a *lazy oracle*.

Remark 1 (Positive and negative data). In the following, we will refer to *positive* data for all pairs $(\boldsymbol{x}^n, \tilde{\boldsymbol{y}}^n)$ such that there is some $q \in \mathcal{Q}$ such that $\tilde{y}_q^n = 1$. All other pairs, i.e. those $(\boldsymbol{x}^n, \tilde{\boldsymbol{y}}^n)$ such that $\tilde{y}_q^n \neq 1,\ \forall q \in \mathcal{Q}$, are termed *negative* data.

C. Building $\widetilde{\mathcal{S}}$ with Uniform Queries

We now describe a scenario that labels the unlabelled training sequence $\{\boldsymbol{x}^n\}_{n=1}^N$ of vectors to give rise to the lazily labelled set $\widetilde{\mathcal{S}} = \{(\boldsymbol{x}^n, \tilde{\boldsymbol{y}}^n)\}_{n=1}^N$. As already pointed out in the introduction, we do not have access to an oracle that may provide us with the correct labels $\{y(\boldsymbol{x}^n)\}_{n=1}^N$. Instead, we consider a *lazy oracle* that, when queried whether q is the correct class for some example $\boldsymbol{x}$, only answers *yes* or *no*.

To build $\widetilde{\mathcal{S}}$, we rely on this lazy oracle by generating a specific type of queries, namely *uniform queries*: for each example $\boldsymbol{x}^n$, we uniformly pick at random one class q among

the Q possible categories from $\mathcal{Q}$ and ask the oracle whether q is the correct label of $\boldsymbol{x}^n$. The lazy label $\tilde{\boldsymbol{y}}^n$ is obtained as follows: if q is indeed the correct label of $\boldsymbol{x}^n$, then $\tilde{y}_q^n$ is set to 1 and the $\tilde{y}_r^n$'s for $r \neq q$ are all set to 0; if q is not the correct label of $\boldsymbol{x}^n$ then $\tilde{y}_q^n$ is set to 0 and the $\tilde{y}_r^n$'s for $r \neq q$ are all set to $\perp$. The following property characterizes the random labels $\tilde{\boldsymbol{y}}^n$ produced by the uniform querying scheme.

Property 1. *Given the uniform querying scheme, the following holds (we drop the superscript n).*

$$\forall q \in \mathcal{Q},\ \mathbb{P}(\tilde{y}_q = 1 | y(\boldsymbol{x}) = q) = \frac{1}{Q} \tag{1}$$

$$\forall q \in \mathcal{Q},\ \mathbb{P}(\tilde{y}_q = 0 | y(\boldsymbol{x}) = q) = 0, \tag{2}$$

and

$$\forall q, r \in \mathcal{Q},\ r \neq q,\ \mathbb{P}(\tilde{y}_r = 1 | y(\boldsymbol{x}) = q) = 0 \tag{3}$$

$$\forall q, r \in \mathcal{Q},\ r \neq q,\ \mathbb{P}(\tilde{y}_r = 0 | y(\boldsymbol{x}) = q) = \frac{2}{Q}. \tag{4}$$

In a more compact way, we may write:

$$\forall q \in \mathcal{Q},\ \mathbb{P}(\tilde{y}_q = 1 | y(\boldsymbol{x})) = \frac{1}{Q}\mathbb{I}\{y(\boldsymbol{x}) = q\} \tag{5}$$

$$\forall q \in \mathcal{Q},\ \mathbb{P}(\tilde{y}_q = 0 | y(\boldsymbol{x})) = \frac{2}{Q}\mathbb{I}\{y(\boldsymbol{x}) \neq q\}. \tag{6}$$

Proof: (2) and (3) are straightforward. For (1) it suffices to see that $\tilde{y}_q = 0$ only if the queried label is indeed q, which happens with probability $1/Q$.

For (4) we use the fact that we are in a single-label scenario. The situations that imply $\tilde{y}_r = 0$ are the following: either the queried label is r or the queried label is q, in which case $\tilde{y}_q = 1$ *and* $\tilde{y}_r = 0$ for all $r \neq q$. Both events have a probability of $1/Q$ to occur, which gives the expected result. ■

D. Goal: Learning a Linear Classifier from $\tilde{\mathcal{S}}$

Our goal is to propose a strategy for learning a classifier from the set $\tilde{\mathcal{S}}$ obtained using the uniform querying procedure. Note that the majority of the results we provide may be extended to the case of more elaborate query schemes.

More precisely, we are interested in producing multiclass linear predictors $f_W : \mathcal{S} \to \mathcal{Q}$ parameterized by a family of vectors $W = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_Q\}$ such that f_W predicts the class of $\boldsymbol{x}$ according to:

$$f_W(\boldsymbol{x}) = \arg\max_{q \in \mathcal{Q}} \langle \boldsymbol{w}_q, \boldsymbol{x} \rangle. \tag{7}$$

To achieve our goal, we build upon the multiclass perceptron learning procedures described by [7] and extend their learning strategy so as to take advantage of the lazily labelled data.

IV. Proposed Approach

This section presents the learning algorithms capable of dealing with lazily labelled data. As a first step, we recall the multiclass perceptron learning schemes proposed by [7].

TABLE I
GENERIC MULTICLASS PERCEPTRON LEARNING SCHEME.

input: $\mathcal{S} = \{(\boldsymbol{x}^n, y^n)\}_{n=1}^N$
output: $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_Q\}$

initialization: $\boldsymbol{w}_1 = \ldots = \boldsymbol{w}_Q = \boldsymbol{0}$
repeat T **times**

- get a labelled pair $(\boldsymbol{x}, y)$ from $\mathcal{S}$
- set $E := \{q : q \neq y \wedge \langle \boldsymbol{w}_q, \boldsymbol{x} \rangle \geq \langle \boldsymbol{w}_y, \boldsymbol{x} \rangle\}$
- **if** $E \neq \emptyset$, **then** choose $\tau_1, \ldots, \tau_Q$ such that
  1) $\tau_q \leq 0$ for $q \neq y$,
  2) $\sum_{q \in \mathcal{Q}} \tau_q = 0$,
  3) $\tau_q = 0$ for $q \notin E \cup \{y\}$,
  4) $\tau_y = 1$,

  and, for $q = 1, \ldots, Q$: $\boldsymbol{w}_q \leftarrow \boldsymbol{w}_q + \tau_q \boldsymbol{x}$, (update)
  endif

endrepeat

TABLE II
BANDITRON LEARNING SCHEME.

input: $\gamma \in [0, 0.5]$
output: $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_Q\}$

initialization: $\boldsymbol{w}_1 = \ldots = \boldsymbol{w}_Q = \boldsymbol{0}$
repeat T **times**

- get instance $\boldsymbol{x} \in \mathcal{X}$
- set $\hat{p} = \arg\max_{q \in \mathcal{Q}} \langle \boldsymbol{w}_q, \boldsymbol{x} \rangle$
- $\forall q \in \mathcal{Q}$, set $P(q) = (1 - \gamma)\mathbb{I}\{q = \hat{p}\} + \frac{\gamma}{Q}$
- randomly sample $\tilde{p}$ from P
- predict $\tilde{p}$ and recieve $\tilde{y}_{\tilde{p}} \in \{0, 1\}$
- let $\tau_1 = \ldots = \tau_Q = 0$,
  $\tau_{\hat{p}} \leftarrow \tau_{\hat{p}} - 1$,
  $\tau_{\tilde{p}} \leftarrow \tau_{\tilde{p}} + \frac{\tilde{y}_{\tilde{p}}}{P(\tilde{p})}$
- and, for $q = 1, \ldots, Q$: $\boldsymbol{w}_q \leftarrow \boldsymbol{w}_q + \tau_q \boldsymbol{x}$, (update)

endrepeat

A. Full-information Multiclass Perceptron

For a multiclass linear predictor f_W, a generic learning scheme suggested by [7] to learn W from a (regularly) labelled training set $\mathcal{S} = \{(\boldsymbol{x}^n, y^n)\}_{n=1}^N$, with $y^n \in \mathcal{Q}$, is provided in Table I. It is interesting to note that [7] have proved the convergence properties of the generic learning algorithm for any choice of $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_Q)$ fulfilling conditions 1, 2, 3 and 4 of Table I whenever the training set $\mathcal{S}$ is actually linearly separable, i.e. whenever there exists $W^* = [\boldsymbol{w}_1^*, \ldots, \boldsymbol{w}_Q^*]$ such that

$$y^n = \arg\max_{q \in \mathcal{Q}} \langle \boldsymbol{w}_q^*, \boldsymbol{x} \rangle, \ \text{ for } n = 1, \ldots, N.$$

This generic algorithm is the cornerstone of the learning algorithms we develop and, in order to fully apprehend their philosophy, it is critical to understand the following fact. The perceptron *does not* use examples for which it performs correct predictions: it needs misclassified examples to build its model, examples for which we do not necessarily know the actual label in the bandit setting.

B. The online bandit perceptron (Banditron)

The *Banditron* is an online algorithm derived from the perceptron for learning in the bandit setting [4]. It operates in

two modes: exploration and exploitation. In exploitation mode, the model is used as in a regular perceptron. In exploration mode, predictions are randomly performed regardless of the model in order to gather misclassified examples for which the label is known. At each round, the banditron randomly outputs a label, by drawing from a distribution biased towards the predicted label. It then uses the information from the oracle to update its model. The γ parameter is used to trade off between exploration and exploitation. Details of the algorithm are given in Table II. Contrarily to the algorithms that we will present in the following section, the banditron works in a fully online setting and does not memorize examples for later use. Interested readers are advised to refer to the original paper for the spectific properties of the Banditron.

C. Misclassified Examples from Positive Data Only

We then consider the batch bandit setting where a training set $\widetilde{S}$ is first gathered using a uniform query and then a classifier is trained using this (lazily-labelled) data. The most naive approach is to learn a regular perceptron on the positive data only and completely ignore examples for which we do not know the label. In this setting, it might be observed that the probability for a vector $\boldsymbol{x}$ to get a positive label is given by

$$\begin{aligned}\sum_{q\in\mathcal{Q}}\mathbb{P}_{\tilde{y}_q}(\tilde{y}_q=1) &= \sum_{q,r\in\mathcal{Q}}\mathbb{P}_{\tilde{y}_q,\boldsymbol{x}}(\tilde{y}_q=1,y(\boldsymbol{x})=r)\\ &= \sum_{q,r\in\mathcal{Q}}\mathbb{P}_{\tilde{y}_q}(\tilde{y}_q=1|y(\boldsymbol{x})=r)\mathbb{P}_{\boldsymbol{x}}(y(\boldsymbol{x})=r)\\ &= \sum_{q}\frac{1}{Q}\mathbb{P}_{\boldsymbol{x}}(y(\boldsymbol{x})=q)=\frac{1}{Q}, \qquad \text{(using (1))}\end{aligned}$$

This means that if learning is based on positive data, only a $1/Q$ fraction of the whole dataset is actually considered, which obviously is suboptimal.

In what follows, we show how to make a more efficient use of the lazily labelled data, and, in particular, how to make use of the negative data.

D. Misclassified Examples from Positive and Negative Data

As mentioned earlier, the perceptron procedure only requires examples of known class that are misclassified by the current model $W=\{\boldsymbol{w}_1,\dots,\boldsymbol{w}_Q\}$. Our learning approach is precisely to *iteratively* create such examples and the originality of our method is that *both* positive and negative data may be used to this end. Namely, given a lazy training set $\widetilde{S}=\{(\boldsymbol{x}^n,\tilde{y}^n)\}_{n=1}^N$, a current model $W=\{\boldsymbol{w}_1,\dots,\boldsymbol{w}_Q\}$ and two labels p and q, the following proposition provides statistical estimates $\hat{\boldsymbol{\mu}}^1_{pq}$ and $\hat{\boldsymbol{\mu}}^2_{pq}$ of a vector $\boldsymbol{\mu}_{pq}$ that i) is of class q and ii) is such that $\langle\boldsymbol{w}_p,\boldsymbol{\mu}_{pq}\rangle\geq\langle\boldsymbol{w}_q,\boldsymbol{\mu}_{pq}\rangle$. We make use of these vectors $\hat{\boldsymbol{\mu}}^1_{pq}$ and $\hat{\boldsymbol{\mu}}^2_{pq}$ to learn from $\widetilde{S}$; the resulting multiclass Lazy Perceptron algorithm is depicted in Table III.

Proposition 1 (Expected Misclassified Data)**.** *Let $\widetilde{S}=\{(\boldsymbol{x}^n,\tilde{\boldsymbol{y}}^n)\}_{n=1}^N$, $W=\{\boldsymbol{w}_1,\dots,\boldsymbol{w}_Q\}$ and $p,q\in\mathcal{Q}$. In addition, let A_{pq} be the subspace*

$$A_{pq}:=\left\{\boldsymbol{x}:\langle\boldsymbol{w}_p,\boldsymbol{x}\rangle\geq\langle\boldsymbol{w}_q,\boldsymbol{x}\rangle,\ \boldsymbol{x}\in\mathbb{R}^d\right\}.$$

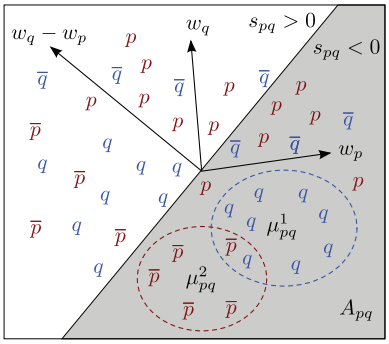

Fig. 1. Illustration of the generation of misclassified examples $\hat{\boldsymbol{\mu}}^1_{pq}$ and $\hat{\boldsymbol{\mu}}^2_{pq}$ from lazily labelled examples; $\bar{p}$ (resp. $\bar{q}$) means that the lazy label associated with the location of the point is such that $\tilde{y}_p=0$ (resp. $\tilde{y}_q=0$). Here $s_{pq}:=\langle\boldsymbol{w}_q-\boldsymbol{w}_p,\boldsymbol{x}\rangle$.

Let $\alpha\in\mathbb{R}$ and let $\hat{\boldsymbol{\mu}}^1_{pq}$ and $\hat{\boldsymbol{\mu}}^2_{pq}$ be the random vectors

$$\hat{\boldsymbol{\mu}}^1_{pq}:=\frac{Q}{N}\sum_{n=1}^N\mathbb{I}\{\tilde{y}^n_q=1\}\mathbb{I}\{\boldsymbol{x}^n\in A_{pq}\}\boldsymbol{x}^n \tag{8}$$

$$\hat{\boldsymbol{\mu}}^2_{pq}:=\frac{1}{2N}\sum_{n=1}^N\left(2-Q\mathbb{I}\{\tilde{y}^n_q=0\}\right)\mathbb{I}\{\boldsymbol{x}^n\in A_{pq}\}\boldsymbol{x}^n \tag{9}$$

$$\hat{\boldsymbol{\mu}}^\alpha_{pq}:=(1-\alpha)\hat{\boldsymbol{\mu}}^1_{pq}+\alpha\hat{\boldsymbol{\mu}}^2_{pq}. \tag{10}$$

The following holds

$$\mathbb{E}_{\widetilde{S}}\hat{\boldsymbol{\mu}}^1_{pq}=\mathbb{E}_{\widetilde{S}}\hat{\boldsymbol{\mu}}^2_{pq}=\mathbb{E}_{\widetilde{S}}\hat{\boldsymbol{\mu}}^\alpha_{pq}=:\boldsymbol{\mu}_{pq} \tag{11}$$

$$q=\arg\max_{r\in\mathcal{Q}}\langle\boldsymbol{w}^*_r,\boldsymbol{\mu}_{pq}\rangle \tag{12}$$

$$\langle\boldsymbol{w}_p,\boldsymbol{\mu}_{pq}\rangle\geq\langle\boldsymbol{w}_q,\boldsymbol{\mu}_{pq}\rangle. \tag{13}$$

Proof: Calculating the expectation of $\hat{\boldsymbol{\mu}}^1_{pq}$. Using the linearity of the expectation, it is sufficient to study the following expectation:

$$\mathbb{E}_{\boldsymbol{x}\tilde{y}_q}\mathbb{I}\{\tilde{y}_q=1\}\mathbb{I}\{\boldsymbol{x}\in A_{pq}\}\boldsymbol{x},$$

where we have dropped the superscript n. We may rewrite the latter expectation as

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{x}}\left[\mathbb{E}_{\tilde{y}_q|\boldsymbol{x}}\left[\mathbb{I}\{\tilde{y}_q=1\}\mathbb{I}\{\boldsymbol{x}\in A_{pq}\}\boldsymbol{x}\right]\right],\\ &\quad=\mathbb{E}_{\boldsymbol{x}}\left[\mathbb{I}\{\boldsymbol{x}\in A_{pq}\}\boldsymbol{x}\mathbb{E}_{\tilde{y}_q|\boldsymbol{x}}\mathbb{I}\{\tilde{y}_q=1\}\right],\end{aligned}$$

and use (5) to get

$$\begin{aligned}\mathbb{E}_{\tilde{y}_q|\boldsymbol{x}}\mathbb{I}\{\tilde{y}_q=1\}&=\sum_{v\in\{1,0,\perp\}}\mathbb{I}\{v=1\}\mathbb{P}(\tilde{y}_q=v|y(\boldsymbol{x}))\\ &=\frac{1}{Q}\mathbb{I}\{y(\boldsymbol{x})=q\}.\end{aligned}$$

As the $\boldsymbol{x}^n$'s are distributed identically (and independently) as $\boldsymbol{x}$ (see section III), we obtain that

$$\begin{aligned}\mathbb{E}_{\widetilde{S}}\hat{\boldsymbol{\mu}}^1_{pq}&=\mathbb{E}_{\boldsymbol{x}}\left[\mathbb{I}\{y(\boldsymbol{x})=q\}\mathbb{I}\{\boldsymbol{x}\in A_{pq}\}\boldsymbol{x}\right]\\ &=\mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}|\boldsymbol{x}\in H_{pq}]\mathbb{P}_{\boldsymbol{x}}(\boldsymbol{x}\in H_{pq})\end{aligned}$$

where H_{pq} is defined as

$$H_{pq}:=\left\{\boldsymbol{x}:y(\boldsymbol{x})=q\wedge\boldsymbol{x}\in A_{pq},\ \boldsymbol{x}\in\mathbb{R}^d\right\}.$$

TABLE III
MULTICLASS LAZY PERCEPTRON.

input: $\widetilde{\mathcal{S}} = \{(\boldsymbol{x}^n, \tilde{\boldsymbol{y}}^n)\}_{n=1}^N$
output: $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_Q\}$

initialization: $\boldsymbol{w}_1 = \ldots = \boldsymbol{w}_Q = \mathbf{0}$
repeat T **times**

- randomly pick a label q (correct label)
- for each label $p \neq q$ (incorrect prediction)

compute a misclassified example $\boldsymbol{\mu}_{pq}^{\alpha}$ using either:

- Lazy I: equation (8)
- Lazy II: equation (9)
- Lazy III: equation (10) with α empirically fixed on the training corpus

and, update the model:

$$\boldsymbol{w}_q \leftarrow \boldsymbol{w}_q + \boldsymbol{\mu}_{pq}^{\alpha},$$
$$\boldsymbol{w}_p \leftarrow \boldsymbol{w}_p - \boldsymbol{\mu}_{pq}^{\alpha},$$

endrepeat

Calculating the expectation of $\hat{\boldsymbol{\mu}}_{pq}^2$ is achieved by following a similar reasoning. It suffices to focus on $\mathbb{E}_{\tilde{y}_q|\boldsymbol{x}}\mathbb{I}\{\tilde{y}_q = 0\}$ to get (using (6)):

$$\mathbb{E}_{\tilde{y}_q|\boldsymbol{x}}\mathbb{I}\{\tilde{y}_q = 0\} = \frac{2}{Q}\mathbb{I}\{y(\boldsymbol{x}) \neq q\},$$

and to use $1 - \mathbb{I}\{y(\boldsymbol{x}) \neq q\} = \mathbb{I}\{y(\boldsymbol{x}) = q\}$ to show that $\hat{\boldsymbol{\mu}}_{pq}^1$ and $\hat{\boldsymbol{\mu}}_{pq}^2$ have the same expectation.

Let us now consider

$$\boldsymbol{\mu}_{pq} := \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x}|\boldsymbol{x} \in H_{pq}]\mathbb{P}_{\boldsymbol{x}}(\boldsymbol{x} \in H_{pq}).$$

By construction, the expectation is computed on vectors of class q (see the definition of H_{pq}): this directly gives (12). At the same time, H_{pq} only contains vectors $\boldsymbol{x}$ such that $\langle \boldsymbol{w}_p, \boldsymbol{x} \rangle \geq \langle \boldsymbol{w}_q, \boldsymbol{x} \rangle$: this directly gives (13). ■

Again, this result provides us with a way to approximately construct misclassified data (where 'approximately' refers to the fact that we compute estimates on a finite sample). Figure 1 graphically represents (a scaled version of) $\hat{\boldsymbol{\mu}}_{pq}^1$ and $\hat{\boldsymbol{\mu}}_{pq}^2$.

The learning algorithm that we have implemented and that makes use of the various estimates presented above is given in Table III. We call our algorithm Lazy Perceptron.

V. EXPERIMENTS ON CALL-CLASSIFICATION TASKS

We carry out a set of experiments on two real-world datasets collected from deployed spoken language understanding services at Orange Labs. We chose two datasets corresponding to two different call-routing services in order to check the relevance of our methods within different experimental conditions. However, in these experiments, the lazy oracle process can be simulated since both datasets contain the true labels for each example. The parameter α (see Table III) is computed at each update so that it minimizes the variance of the estimate $\boldsymbol{\mu}_{pq}^{\alpha}$.

A. The call-routing corpora

Two corpora have been extracted from two different customer-care services exploiting a natural language call-routing system. The first level in the Spoken Language Understanding (SLU) module of these systems is a call-routing classification module leading to around 10 targets for both applications. All the experiments reported in this paper are made at the call-routing classification step.

The first application (*A1*) has been designed for a general public customer-sale service while the second one (*A2*) is a technical assistance application. The corpora have different complexities and class distributions. *A1* contains 8 targets. It has a dominant target which covers nearly 67% of the utterances. The second most frequent target represents 16% of the utterances. *A2* is composed of 10 targets and has a more balanced target distribution, with the dominant target covering 49% of the utterances and the second most represented covering 14% of the data. All these experiments have been carried out on the Automatic Speech Recognition transcriptions of the customers requests. On overall the word error rate for the *A1* application is around 45% and around 30% for the *A2* application. Each corpus has been split between a training and a test set, sorted by the date of collection. The training corpus of *A1* contains 17,000 requests collected between 03/2008 and 05/2009. The *A1* test corpus contains 3,800 requests collected after the traininig set in 05/2009. Similarly the training corpus of *A2* contains 23,968 requests collected between 10/2010 and 03/2011 and the test corpus contains 1,000 requests collected in 03/2011.

B. Evaluation protocol

Each experiment is run with an increasing number N of training examples, and for each size 20 runs are performed to be able to plot means and standard deviations. Six algorithms are compared:

- *Banditron*: this corresponds to the online Bandit algorithm with an exploration vs. exploitation ratio fixed to 0.15
- *Regular*: only the examples for which the class is known — i.e. the positive examples from the oracle — are used to train a *Classical Perceptron* (as described in Table I)
- *Lazy I, II, III*: given an unlabeled training set $\tilde{\mathcal{S}}$ of N examples, each example is submitted to the *Lazy Oracle* with one of the Q random classes according to the uniform query scheme. The resulting *Lazy Labels* are used to train a *Lazy Perceptron* in modes I, II and III
- *Full*: acting as a reference, the *Full* experiment leverages the hypothetic data set $\mathcal{S}$ to learn a *Classical Perceptron* on all fully-labeled examples. This is a topmost result we strive to reach.

C. Results

The results are presented in Table IV.

As expected, the classifier of the *Full* experiment outperforms all the other classifiers, and is a topline for them. The *Regular* experiment uses roughly $\frac{1}{Q}$ examples and therefore behaves like the *Full* perceptron at that amount of data. *Regular* and *Lazy I* start with similar results for corpora up to 2,000 examples. After 2,000 examples *Lazy I* gives better results than *Regular* while using the same quantity of positive examples. Even if the gain is small, this shows that *Lazy I*

TABLE IV
CLASSIFICATION ERROR RATE FOR THE *A1* AND *A2* CORPORA

Corpus *A1*						
#added examples	Banditron	Regular	Lazy I	Lazy II	Lazy III	Full
200	29.2	31,5	31.0	47.7	29.3	26.7
500	28.3	31.0	29.5	47.5	29.2	22.8
1000	26.9	28.0	28.0	46.4	27.7	23.4
2000	24.5	25.1	24.7	40.1	24.4	23.1
5000	23.5	22.4	21.6	35.1	21.0	15.4
10000	22.1	19.6	18.4	32.4	18.2	14.4
Corpus *A2*						
#added examples	Banditron	Regular	Lazy I	Lazy II	Lazy III	Full
200	27.4	45.8	48.0	71.8	46.7	25.4
500	23.0	38.7	38.3	58.0	37.5	17.9
1000	20.8	30.2	28.9	51.6	29.2	16.7
2000	20.9	24.6	24.9	48.4	23.9	13.6
5000	19.2	19.7	18.1	41.1	18.2	12.7
10000	18.8	16.4	14.8	37.0	14.7	9.7

can take advantage of the generated examples to improve its performance. *Lazy II* which uses only negative examples (the oracle always said "no") starts poorly but manage to reduce significantly its error rates on large corpora. This shows that with sufficient data, our *Lazy Perceptron* can learn from the $\frac{Q-1}{Q}N$ negative examples without information on their real class label. The *Lazy III* experiment combines positively and negatively labeled examples to create virtual examples as a linear interpolation between the *Lazy I* and *Lazy II* examples. The *Banditron* algorithm performs very well for small size of data but provides worse results than the Bandit batch algorithms *Lazy I* and *Lazy III* when using the whole training corpus.

All these results have been obtained with automatic transcriptions with a relatively high Word Error Rate (over 30%), both for training the classifiers and testing them. This means that our learning algorithms are robust to the noise generated by Automatic Recognition Systems.

VI. CONCLUSION

We have explored in this paper a *lazy labeling* method as a cheap way to train classification models in the context of call-routing spoken dialog systems. In such a labeling process, examples are presented to annotators with a label which they either validate or invalidate. Such lazy labels can be obtained through customer feedback, agent annotations or call log analysis. We have proven that it is possible to learn linear classifiers in this setting, by estimating adequate expectations. A set of experiment was run on two call center routing tasks. We have shown that under a uniform query, the *Lazy Perceptron* algorithm performs better than a regular perceptron only using the small proportion of positively labeled data. An interesting outcome is that even when relying solely on negative instances, for which we know that they are not of a given label, the perceptron can learn a model. The uniform query setting is an interesting exploration strategy that doesn't need any bootstrap corpus and which can learn new events only from this weak supervision.

As for future work, we plan to extend this setting with more query types, including data truly lazily labeled by humans. The idea of generating surrogates of misclassifed examples is also appealing and might be beneficial to other learning algorithms than the perceptron. We plan on trying more elaborate example generation techniques and eventually we hope to be able to extend this setting to structured predictions which power most of machine learning applications in natural language processing.

VII. ACKNOWLEDGMENT

This work is supported by the French agency ANR, Project DECODA, contract no 2009-CORD-005-01.

REFERENCES

[1] H. Robbins, "Some aspects of the sequential design of experiments," *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2003.

[3] L. Li, W. Chu, J. Langford, and R. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on WWW*. ACM, 2010, pp. 661–670.

[4] S. M. Kakade, S. S. Shwartz, and A. Tewari, "Efficient bandit algorithms for online multiclass prediction," in *ICML '08: Proc. of the 25th Int. Conference on Machine learning*, 2008, pp. 440–447.

[5] H. Valizadegan, R. Jin, and S. Wang, "Learning to trade off between exploration and exploitation in multiclass bandit prediction," in *ACM Conference on Knowledge Discovery and Data Mining (KDD 2011)*, 2011.

[6] J. Langford and T. Zhang, "The epoch-greedy algorithm for contextual multi-armed bandits," in *Adv. in Neural Information Processing Systems 20 (NIPS 2008)*, 2008.

[7] K. Crammer and Y. Singer, "Ultraconservative Online Algorithms for Multiclass Problems," *Journal of Machine Learning Research*, vol. 3, pp. 951–991, January 2003.

[8] G. Riccardi and D. Hakkani-Tur, "Active learning: theory and applications to automatic speech recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 504–511, July 2005.

[9] D. Bohus and A. Rudnicky, "Implicitly-supervised learning in spoken language interfaces: an application to the confidence annotation problem," in *Proceedings of SigDial*, 2007, pp. 256–264.

[10] P. Gotab, F. Béchet, G. Damnati, and L. Delphinpoulat, "Online SLU model adaptation with a partial Oracle," in *Proceedings of Interspeech'10*, Makuhari, Japan, 2010.