Crowdsourcing Thousands of Specialized Labels: A Bayesian Active Training Approach

Maximilien Servajean, Alexis Joly, Dennis Shasha, Julien Champ, Esther Pacitti

Crowdsourcing Thousands of Specialized Labels: a Bayesian active training approach

Maximilien Servajean, Alexis Joly, Dennis Sasha, Julien Champ, Esther Pacitti

Abstract

Large-scale annotated corpora have yielded impressive performance improvements in computer vision and multimedia content analysis. However, such datasets depend on an enormous amount of human labeling effort.

When the labels correspond to well known concepts, it is straightforward to train the annotators by giving a few examples with known answers. It is also straightforward to judge the quality of their labels. Neither is true when there are thousands of complex domain specific labels. Training on all labels is infeasible and the quality of an annotator's judgements may be vastly different for some subsets of labels than for others.

This paper proposes a set of data-driven algorithms to (i) train image annotators on how to disambiguate among automatically generated candidate labels, (ii) evaluate the quality of annotators' label suggestions and (iii) weight predictions. The algorithms adapt to the skills of each annotator both in the questions asked and the weights given to their answers. The underlying judgements are Bayesian, based on adaptive priors.

We measure the benefits of these algorithms on a live user experiment related to image-based plant identification involving around 1,000 people. The proposed methods are shown to enable huge gains in annotation accuracy. A standard user can correctly label around 2% of our data. This goes up to 80% with machine learning assisted training and assignment and up to almost 90% when doing a weighted combination of several annotators' labels.

M. Servajean (servajean@lirmm.fr), A. Joly (alexis.joly@inria.fr), J. Champ (champ@lirmm.fr) and E. Pacitti (pacitti@lirmm.fr) are with the Zenith Team from INRIA at LIRMM, +33467149772, 860, rue de St Priest, 34095 Montpellier, France.

Dennis Shasha is with NYU in the Dpt of Computer Science and has an international chair from INRIA, shasha@courant.nyu.edu, 251 Mercer Street New York, NY 10012, U.S.A.

I. Introduction

Recent years have seen the emergence of considerable progress in computer vision and multimedia analysis techniques. In addition to fundamental algorithmic contributions, the availability of large-scale annotated datasets has played a central role in this performance improvement. Popular datasets such as LabelMe [26], ImageNet [11], or more recently YFCC100M [33] have formed the basis of thousands of research publications and have greatly influenced the research directions taken by the multimedia and machine learning communities. In particular, the recent increase in attention on convolutional neural networks is largely due to the impressive performance gains these methods have enabled in the context of the ImageNet classification challenge [19]. However, building such large datasets is often challenging since it requires the accurate tagging of huge quantities of media items. Therefore, the use of crowdsourced and more generally user-generated annotations became the *de facto* methodology for building training data in a variety of multimedia tasks [3], [11], [20], [28]. For instance, with Amazon Mechanical Turk, each image is presented to a set of users whose task is to provide relevant tags [3], [11]. Platforms such as Zooniverse [5] have focused their action on citizen science data such as astronomical pictures of galaxies. Citizen scientists have processed large amounts of scientific data that could not have been processed otherwise.

When the labels correspond to well known or easy-to-learn concepts, it is straightforward to train the annotators by giving a few examples with known answers. Neither is true when there are thousands of complex domain specific labels. In this paper, we focus on the particular case of crowdsourcing domain-specific annotations that usually require hard expert knowledge (such as plant species names, architectural styles, medical diagnostic tags, etc.). We consider that common knowledge is not sufficient to perform the task but any people can be taught to recognize a small subset of domain-specific concepts. In such a context, it is best to take advantage of the various capabilities of each annotator through teaching (annotators can enhance their knowledge), assignment (annotators can be focused on tasks they have the knowledge to complete) and inference (different annotator propositions can be aggregated to enhance labeling quality).

In typical crowdsourcing workflows, annotators receive tasks such as labeling a picture or media item. Because the quality of each individual response can be low, most systems make use of redundancy. Thus, the same task is usually assigned to several annotators. Then, by aggregating

the results, the system can achieve higher quality. Our system focuses on classification tasks. That is, we are given a dataset to label (images in our case) and a set of classes (plant names in our case). Each item within the dataset is associated to an unknown true class which needs to be determined. State-of-the-art crowdsourcing algorithms generally aim at resolving two main issues:

1) **task assignment:** Each task requires some skills to be performed and each worker is associated with some of those skills. The goal is to assign tasks to qualified users, giving the right amount of redundancy to ensure quality and maximize the number of correctly completed tasks.
2) **classification inference:** Each annotator, proposes a class for a given item. The goal is to aggregate the class votes in order to find the correct one for each item.

Task assignment is often considered to be an optimization problem: the goal is to maximize the number of correctly completed tasks. This is done by estimating the skills required to resolve the tasks and the skills associated with each user [2], [24], [34]. Then, one can derive the likelihood of each task to be correctly performed given the set of annotators assigned to it and maximize the number of correctly completed tasks.

Crowdsourcing algorithms for classification inference [10], [18], [25], [31], [36] are typically based on a Bayesian inference of the most probable labels according to the confusion matrix of each worker. The confusion matrix of each annotator is usually derived from that annotator's previous contributions (*i.e.* labels given to other items) and thus represents the annotator's established classification ability. New classification tasks can then be assigned according to these abilities [2], [34] and the most likely class can be inferred from the provided labels. Traditional Bayesian models [10], [18] are initialized by taking into account the known confusion of each class with respect to all other classes for all given annotators.

There have been approaches that have successfully treated thousands of classes (e.g. cars, trees, people) such as ImageNet [11] but the few specialized classes of such datasets are very noisy and therefore unusable [16]. Other works [21] propose to define a balanced and structured taxonomy that can be used to ask few questions in order to quickly identify the correct class even when confronted with thousands of potential classes. Unfortunately defining such a balanced taxonomy is not trivial for many expert problems including plant classification [7] for which the natural taxonomy is strongly imbalanced (*e.g.* among 197 families of the French flora, one family accounts for 800 species, while 120 of these families account for less than 10 species

each). None of these works have dealt successfully with thousands of specialized classes. Indeed, the very large number of classes, *e.g.* thousands of plant species or of any other named entities, makes it infeasible to train a complete confusion matrix for each participant as it would require them to answer a huge number of queries (typically quadratic in the number of classes). We therefore propose new crowdsourcing models and algorithms for complex classification tasks that take into account the high dimensionality of the problem and the need to dynamically adapt the hypothesis space (*i.e.* the set of classes presented to each user).

As Simpson and Robert [30] suggest, each potential annotator ideally would need to be trained [1] to disambiguate among the plausible candidate classes of each item. In the context of complex classification tasks involving very large number of classes, we propose an active user training framework where the training hypothesis space is adaptively and dynamically chosen for each user and observation through machine learning techniques. For instance, if at some point a data item has some probability distributions $p(t_i = j)$ over the classes, then it is useful to train one or more annotators to disambiguate among the most probable classes. Further, since a given annotator is trained only on a subset of classes, the system should ask him or her to classify data items corresponding to the classes with which he or she has some familiarity.

This paper presents the following original contributions in order to address the problem of thousands-of-classes domain-specific image annotations crowdsourcing:

- We address the cold-start problem (*i.e.* find an initial probability distribution over the classes) by using the predictions of a machine learning algorithm (in the event, a convolutional neural network). This will enable us to assign tasks appropriately and to build the appropriate training quizzes.
- We then introduce *Task Skills-Aware Assignment* that assigns tasks to annotators based on their confusion matrices.
- Third, we present a new method called *Active User Training*. Here, the goal is to train annotators to disambiguate a specific set of classes. To do so, we form a quiz composed of the most likely classes for a given item according to the previously hypothesized classes for that item (whether human or machine learning based).
- Finally, we present a variant of the Bayesian inference framework devised by Simpson and Robert [30] which we improve by taking into account the uncertainty of the confusion matrix prior resulting from the system's partial knowledge of the annotators.

To evaluate the proposed solutions, we implemented a publicly available game at theplantgame.com[1]. It is part of the Pl@ntNet system [15]. Pl@ntNet is an innovative participatory sensing platform relying on image-based plant identification as a mean to enlist non-expert annotators and facilitate the production of botanical observation data. It relies on a mobile application available on iOS and Android which enables the users to take images of plants and to receive in return the most likely species. The data stream generated through the app consists in a set of plant observations. This dataset can be used to monitor bio-diversity, invasive species and general plant population structure. However, although machine learning might successfully identify some observations, the data stream is highly noisy and therefore would benefit from large-scale human validation.

This paper is organized as follows. Section II introduces the related work upon which we have built our method. Section III presents the problem definition as well as our framework architecture. In Section IV we describe *active user training*. Section V presents our *skills-aware batch assignment* approach. Section VI describes our variant of the Bayesian *inference* model. Section VII presents our experimental evaluation and we conclude the paper in Section VIII.

II. Related Work

In our work, we address three main questions: how to train users, how to assign tasks to users, and how to perform inference based on the results. We therefore propose an overview of the previous work addressing these questions.

A. Classification Inference

Although crowdsourcing is still an emerging domain, contributions related to classifier combinations, error-rate evaluations and other issues started in the $1970s$. One of the first works on the topic of multiple classifiers was done by Dawid and Skene [10]. It focused on estimating the error-rate of several expert classifiers from a noisy ground truth. This work forms the basis of a lot of the current work that we will present in this section.

In [35], Tulyakov *et al.* propose an overview of several classifier combination methods. From the simplest, such as a majority voting or Borda – the rank of a result depends on its rank

[1]http://theplantgame.com

proposed by all classifiers – to more complex methods such as the ones that use the Dempster-Shafer theory of evidence.

Recently, Bayesian models have been shown to outperform other combination methods [18], [25], [31]. In [25], Raykar *et al.* show how to learn a classifier using data features and classification votes. Their approach is only partially Bayesian as the model is used only as a "point estimate". Additionally, the main goal was not to perform classification but to learn a classifier on more realistic labels: each item in the training dataset is not associated to a single class but to a probability distribution over all the classes. The goal is to determine these probability distributions and to train a classifier on it.

Kim and Ghahramani [18] propose a complete Bayesian model to aggregate the classifications votes of multiple imperfect classifiers, be they humans or machines. They develop two main approaches, one taking into account the dependency among classifiers and one ignoring it. They show that complex models that take into account the dependencies among classifiers do not actually enhance the classification quality. In addition, they use Gibbs sampling to evaluate the model – Gibbs sampling is however expensive to run and therefore unusable when thousands of labels are present (as in our work).

In [36], Venanzi *et al.* propose to exploit correlations among the annotators to increase the classification quality. Thus, each user is part of a community whose members are likely to have the same confusion matrix. The main advantage of this work is to estimate the classification capabilities of the annotators more precisely even though very few classification examples are available. In contrast to our setting, they consider each annotator to be already skilled and capable of performing every task.

Simpson *et al.* [31] propose a Bayesian approach similar to the one introduced by Kim and Ghahramani. However, they use a Variational Bayesian approach to estimate the model. Additionally, they model evolving skills using the notion of dynamic classifier combination. Our inference procedure builds upon their variational model.

Welinder and Perona [38] also provide a Bayesian approach for image classification. We differ from their approach in two main ways. First, they consider the annotators to be already skilled. Second, they iteratively try to assign tasks to experts. In our case, experts are a rare resource so assignment is considered to be a global optimisation problem.

Parameswaran *et al.* [21] propose to identify the correct class by following a set of questions structured in a graph. The goal there is to find the minimum expected number of questions in

order to find the true classes of all items. However, defining a taxonomy to reduce the problem difficulty when confronted with thousands of classes is not trivial. As already mentioned earlier, the plant taxonomy tree is also completely unbalanced, so each question would just remove few species at each step (rather than, say, half). Defining a plant taxonomy based on visual facets has also been shown to produce unbalanced results [7].

Bragg *et al.* [6] propose a method where the confidence in an annotator's ability is computed based on his/her number of correct classification (from a ground truth). In our work, we use a Bayesian network to infer the annotators' abilities.

B. Task Assignment

Task assignment refers to the action of assigning a task to a user depending on his own skills.

Tran-Thanh *et al.* [34], propose a strategy where up to B tasks can be assigned in a way that minimizes the expected number of misclassified items. That is, each user is supposed to give an answer that follows a Bernoulli distribution. Then, by applying a fusion method, the correct label of each task may be estimated. The goal is to assign tasks to annotators without exceeding the budget while maximizing the number of correctly estimated labels. However, they consider all annotators capable of accomplishing all tasks. This is not true in our scenario nor in any scenario in which classification requires specialized knowledge (basically any field such as biology, ecology, or epistemologically similar application).

Basu *et al.* propose [2] an approach that, given an objective function, efficiently allocates tasks to workers. Their approach supposes that each user provides a constant quality gain (*i.e.* the quality gain does not depend on the task), an assumption we cannot make as some classes are more difficult to disambiguate.

As some tasks require expert knowledge, Gottlieb *et al.* [13] propose a method to discover experts in the crowd. Their approach consists of asking complex question where the ground truth is known and only when correct answers have been given by the user, ask him to perform the task.

Simpson and Robert [30] use the Kullback-Leibler information gain formula to determine the most relevant annotator to whom a specific task should be assigned. To do so they consider a Hire and Fire policy. They assume that tasks are allocated one by one and if an annotator is not allocated a task immediately, he/she will leave. In our scenario, we have found that annotators

may stay for a sufficiently long time. Thus we consider task assignment as an optimization problem where several annotators may be assigned to several tasks and vice versa.

Sheng *et al.* [29] and Parameswaran *et al.* [22] propose a probabilistic model to determine which item will most likely benefit from being assigned. In our model, as inference generates probabilities, one can easily be used to fix a threshold for instance (*e.g.* all items having probabilities over 99% of belonging to one class are considered correctly classified).

All these works consider each worker to be already sufficiently skilled. Simpson and Robert [30] admit the possibility that a worker's skills may evolve through time and suggest training as one possibility but do not address the problem of training.

C. System Oriented Works

In this subsection, we consider works that combine several techniques in a system oriented way.

In [23], Quinn *et al.* combine machine learning with crowdsourcing to address three objectives: maximize the processing speed, minimize the cost and maximize the quality. These objectives are part of the *input* and the system automatically allocates the tasks to humans and machines accordingly. They consider the crowdsourced tasks as sufficiently easy to be performed by everyone and don't address profile-based task assignment.

In [17], Kamar *et al.* develop an architecture called *Crowd Synth* that provides a trade-off between cost and accuracy under budget constraints. The platform is based on a Markov Decision Process. They consider each worker as already skilled.

Welinder and Perona [38] propose to progressively select items to be processed depending on a budget constraint, thus offering an accuracy-cost trade-off. We follow their approach of using Bayesian Inference to estimate the true labels. We also follow them in estimating the expertise of each user based on the posterior probabilities of the Bayesian model. In their context, all users have the same skills, so users are interchangeable. In our case, many plants can be identified by many people, but a few require special expertise, so we reserve those for our expert informants.

All these papers assume that human classifiers have sufficient skills from the beginning to perform the tasks. Therefore, they presume no training is necessary (or they don't address the training problem). In our case, we cannot make these assumptions because most annotators are initially not experts in most labels. Our goal is thus to educate them first and to assign them tasks they can solve accordingly.

D. Annotator Training

In [12], Fang *et al.* study an active learning approach. Like us, they assume that annotators can be trained and their performance improved. To do so, they select items that need to be classified and for each of them, they construct a tuple of the most reliable annotator and the least reliable one. Thus, the knowledge from the best annotator helps to improve the performance of the worst one. We differ from them in three ways. First, they suppose that no training data is available to perform the training. In our case, we use training data both to train a convolutional neural network and to train the crowd (the annotators). Second, in complex classification problem as ours, all annotators may not be capable of learning all the sub-problems. Thus, finding the best and the worst annotator for training may end up with no performance gain. For instance, in our plant classification problem, species with the biggest potential gain actually correspond to species almost impossible to learn for beginners. Finally, we study the whole workflow from training, assignment to inference.

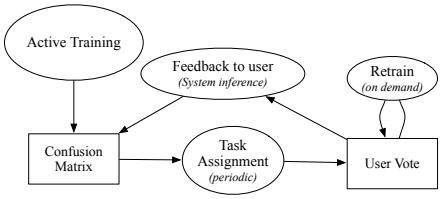

Fig. 1. Framework Architecture: User View.

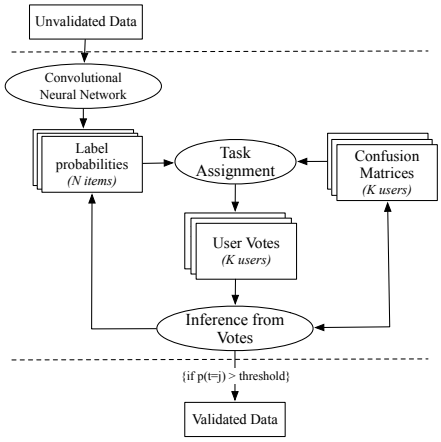

Fig. 2. Framework Architecture: System View.

III. Problem Definition and Framework Architecture

In this section, we present the problem definition as well as the framework architecture introduced in this paper.

Problem Definition: we are given a dataset D of items that need to be classified and a set of classes C. Each item in D is associated to an unknown true class in C that needs to be discovered. We are given a set of annotators K. Each annotator can suggest a possible class in C for an item in D. The goal is to maximize the number of true class discovered for all items in D. To achieve such results, we focus on three intermediate goals:

1) how to train human annotators efficiently,
2) how to assign items in D to human classifiers in K who have a good chance of proposing a correct classification,
3) how to infer the most probable classes based on the multiple classifiers' votes.

To address these problems, we rely on the architecture presented in Figure 1 for the users'

view and in Figure 2 for the system's view. It is composed of four modules:

1) **Convolutional Neural Network**: when a new (unvalidated) data item is added to the system, a machine learning module will first predict the set of label-probability pairs. In our experiments we use the convolutional neural network architecture described in [32] pre-trained on ImageNet and fine tuned on LifeClef's training data [8] which gives state-of-the-art performance for image-based plant identification. However, our system could be generalized to any other data and machine learning module. Each new item is thus systematically associated to a probability distribution $p(t_i = j)$ indicating the likelihood of a class for an item. When the probability of one class exceeds a system defined threshold (*e.g.* $t = 99\%$ in our experiments), the item is automatically tagged as validated. Otherwise, the item is considered as unvalidated and is issued to the iterative crowdsourced human validation processes (*i.e.* active training, task assignment and classification inference).
2) **Active training**: each item is associated with a probability distribution indicating the likelihood of a class for an item. Based on the likelihood of all items, the system creates quizzes containing the candidate classes so that the annotators learn how to disambiguate them.

   Each quiz is associated to its difficulty which is the probability that the annotators will successfully answer the questions in the quiz. Then, when some annotator starts a training session, the system selects a quiz whose difficulty is adapted to his current skills. That way, annotators are trained to disambiguate classes that actually correspond to the problem at hand. In parallel, the system starts to learn the annotators' confusion matrices to use the during the inference process (as explained in paragraph 4) as well as to estimate the skills and success probabilities of each user. A confusion matrix for a user maps a label j to a set of label-probability pairs P, noted $\pi_j^{(k)}$ for user k, denoting the likelihood that the annotator will mistake a plant having label j with each of the labels in P.
3) **Task Assignment**: the system assigns an unvalidated item to the annotators who are likely to assign a correct class to them, based on the annotators' confusion matrices. Here, we consider annotators' success to be more important than the overall information gain as the annotators must be rewarded to obtain better cognitive skills [14]. Most related works allow the opposite: an annotator who always gives the same wrong answer would be considered to be a high quality classifier.

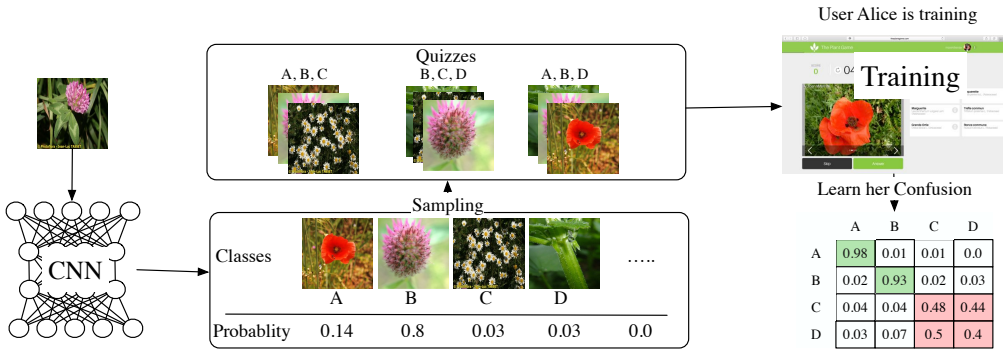

Fig. 3. Active training example. An item is first processed by the CNN which produces the list of probabilities associated to each class. This probability distribution is then sampled several times to create training quizzes. Once some user (*e.g.* Alice) has completed a training session, her confusion matrix is updated..

4) **Inference**: as labels (votes) are given by the annotators to a set of items, an inference model will update their probability distributions as well as the annotators' confusion matrices. Once the probability that an item belongs to a class exceeds the threshold t, it is considered validated and the annotators who were assigned to it are informed of the result (feedback to the annotator, *cf.* Figure 1). Otherwise, it enters a new cycle of validation but with its updated probability distribution $p(t_i = j)$ which reflects reduced uncertainty.

IV. Annotator Active Training

In this section we present our method for training the annotators. We restrict ourselves to the case of training solely based on quizzes with feedbacks (*i.e* quizzes in which we indicate to the annotator the correct answer after each question so that he or she can learn through *trial and error*).

Unlike machine learning-based classifiers, humans cannot face and learn a thousands of class problem directly. For instance, in plant classification, a non-expert human would be completely lost if he had to identify an observation among thousands of species.

Thus, to train the annotators in an effective way, the idea is to generate quizzes of small size (*e.g.* 6, 8, 10 or 12 among thousands of possible classes). A quiz corresponds to a set of classes $Q = \langle c_1, ..., c_k \rangle$ that the annotator will be trained on and a number of questions m. Then, running the quiz entails sampling plant images uniformly from each class in Q and then presenting those to the annotator. Each time a user indicates his class (or classes if he is hesitating) assignment

to a training image v, the system gives him/her the correct answer. To make this a game, the more often a user gives correct answers, the more points he/she receives.

Given all $\binom{n}{s}$ possible quizzes of size s among the total number of classes n, we want to select those "useful" for our classification purpose and assign them to each annotator. A way of doing so would consist of finding the optimal set of quiz/annotator pairs that maximizes the expected number of correct classifications given the likelihood of all unvalidated items and the annotators' confusion matrices $\pi^{(k)}$. However, such ideal optimization is non trivial because it would require to compute *a priori* the posterior probability of success of a specific annotator on a quiz (*e.g.* the probability $p(\pi'^{(k)}_{l,j}|\pi^{(k)}_{l,j}; q)$ of the annotator's confusion $\pi'^{(k)}_{l,j}$ before he/she has been trained on a quiz q). This is problematic in several ways. First, it is not trivial to estimate how well an annotator will assimilate the given lesson: some annotators may learn quickly and durably, while some others may not. More generally, estimating $p(\pi'^{(k)}_{l,j}|\pi^{(k)}_{l,j}; q)$ depends on many neuropsychological factors that are very complex to model.

Second, even considering a simpler heuristic such as training an annotator on his/her current most confused classes can be problematic. Such classes usually correspond to the most difficult ones that only experts can deal with. Thus, he/she might not be able to disambiguate them even after dozens of training quizzes.

As an alternative, we propose a two-step approach. Figure 3 presents such training through an example that will be discussed throughout this paper. First, a large set of quizzes are created independently of the annotators' confusion matrices. Second, the quizzes are assigned to the annotators based on the expected difficulty of each quiz and each annotator's confusion matrix.

1) **Quiz creation via Monte-Carlo sampling**: the goal here is to create quizzes based on the likely labels of the unvalidated items independently of the annotators' confusion matrices. Given an unvalidated item i and its labels' likelihood $p(t_i)$, a set of quizzes is generated through Monte-Carlo sampling. In other words, the probability of appearance of a class j in a quiz is proportional to its likelihood $p(t_i = j)$ – based on the outputs of the machine learning module or on the previous inference results if some classifications propositions have already been retrieved. Using Monte-Carlo sampling rather than creating the quiz based on the top-k most probable classes is beneficial for several reasons. First, it increases the probability that the correct class appears in at least one of the quizzes, even if it is very low at the beginning. Whatever the probability p of the correct class for an item, the probability that it appears in at least one of q quizzes of k classes is $1-(1-f(p,k))^q$ where

$f(p, k)$ is the probability mass function of the non-central hyper-geometric distribution. Whatever the values of p and k, there is always a sufficiently large number m of quizzes guaranteeing that the correct species appears in at least one quiz and thus that at least one user has been trained on that class. Second, as the quizzes are created through random sampling, annotators are trained in complementary ways and their classification votes tend to be more statistically independent (which is a nice property as we want to aggregate their votes). In the example of Figure 3, the machine learning module (CNN) returned a probability distribution for a new item. Only four classes have positive probabilities. These four classes are sampled several times with respect to their probabilities in order to create several subsets of all the classes. Thus, in the end, several groups of classes currently considered likely for an item are created and are directly used as quizzes.

2) **Quiz assignment via difficulty estimation:** Given some user k, the goal here is to assign him/her quizzes of increasing difficulty. Based on the confusion matrices of all annotators, one is able to compute the difficulty of a quiz thanks to its expected success probability:

$$p(success) = \sum_{j \in J} \pi_{jj} \times p(t_i = j) \tag{1}$$

where J refers to all the classes, π_{jj} is the expected probability that some user will give the correct answer when confronted with class j and $p(t_i = j)$ is the probability that item i belongs to class j.

Each user is associated with a threshold indicating the maximum allowed difficulty. Then, the more a user plays, the lower will be the threshold and the more difficult the quiz assigned to him will be. The threshold is first initialized based on the user's declared expertise.

The system, given some user, will automatically and randomly select a quiz which meets the threshold criteria.

As an example, a beginner who just starts to play should be trained on quizzes whose expected success rate is superior to 99% across all quizzes. Note that the more a user plays, the more difficult the quizzes will be and also the more classes he/she will be asked to remember because quizzes are selected randomly.

Fighting Cognitive Bias. Another limitation of human being is cognitive bias. In our preliminary experiments, we have observed three main biases the annotators are subject to. First, when confronted several times with the same answers, the annotators tend to remember the position

of the answer in the list instead of its label. Second, we also noticed that annotators tend to remember the shape of the words constituting the labels instead of the words themselves. Both of these biases prevent the annotator from remembering the labels names durably. The third bias we noticed concerns the images used to illustrate each class. When only presented with a small number of different classes, the annotator tends to remember simplistic discriminative features for each class. These features may work well for the small number of classes in the quiz but can't be generalized.

To address these issues, we proposed the design of training sessions to incorporate redundancy. Instead of training the user on a quiz composed of p questions, we train him on k different quizzes of p/k questions. All the k quizzes are sampled from the same likelihood but with an increasing number of classes. They are therefore likely to intersect in terms of the classes they contain but will force the user to learn new discriminative features and force him to read and remember the labels. The position of each question is also randomized between each quiz so that the annotator does not learn the position of the answers. Such an approach is inspired from experiments on mice and has been applied on Amazon Mechanical Turk[2] where the system penalizes motor skills and gives rewards (points) to cognitive skills [14].

Fighting Forgetfulness. As another limitation of human beings is memory, we observed that sometimes annotators performed very well on a given species during training but forgot its name when an image of that plant was assigned to them later on. To address this problem, we introduced the notion of *on demand retraining* (*cf.* Figure 1). When an annotator is assigned to a new task, he/she may run a new *ad-hoc* training quiz which is directly sampled from the distribution $p(t_i = j)$ of the item that needs to be annotated.

Starting to Learn User Confusions. As soon as users start to train, it is possible to detect classes among which they are confused. In the example of Figure 3, Alice had a 0.98 success rate for class A but tended to confuse classes C and D. This information gives a prior idea of each user's ability to classify the items among the classes. This confusion matrix will be further enriched and re-inferred during the inference step (see Section VI).

Note that having all this data to initialise the confusions of all users is very important in the context of thousands of classes where the total number of possible confusions is quadratic in the number of classes.

[2]www.MTurk.com

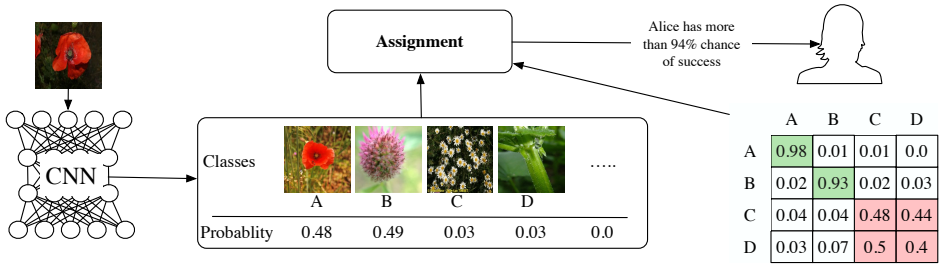

Fig. 4. Assignment example. An item is first processed by the convolutional neural network (CNN) which produces the list of probabilities associated to each class. This probability distribution is then used together with the user confusion matrices to estimate the user's chance of success. Here Alice is very good at classifying items in classes A or B which are the most likely classes of the new item. By contrast, she is not at all good at classifying among other classes. Therefore, the system assigns her to this item, because she is a good candidate.

V. Skills-Aware Task Assignment

We propose here a task assignment strategy that takes the user's inferred skills into account when assigning unvalidated items.

The objective function we want to maximize here is the number of objects for which at least one user has proposed a correct classification. Pragmatically, however, there are some constraints. First, based on each user k's willingness to work, user k should be assigned no more than n_k objects. Also, no object i should be assigned to more than m_i users to avoid an unbalanced tagging of the items. Given these considerations, the objective function can be described as follows:

$$\begin{aligned} \underset{x_{ik}}{\text{maximize}} \quad & \sum_{i=0}^{N}\sum_{k=0}^{K} x_{ik} \times f(p(\text{success}_i^{(k)})) \\ \text{subject to} \quad & \sum_{i=0}^{N} x_{ik} < n_k \ \forall k, \sum_{k=0}^{K} x_{ik} < m_i \ \forall i \\ & x_{ik} \in \{0,1\} \ \forall i,k. \end{aligned}$$

Where $x_{i,k}$ takes the value 1 if item i is assigned to annotator k and 0 otherwise, and N and K are the total number of items and annotators respectively. Additionally, if the success probability is too low, a user may not want to answer at all – most annotators prefer not to answer when they feel very unsure. Thus, we introduce a function $f(p(\text{success}_i^{(k)}))$ which tends to 0 when the probability of success decreases and to 1 when the probability of success increases. Thus, the function f enables more accurate prediction of the success of a user. Notice that this objective function is very similar to the one proposed by Simpson *et al.* [30] based on entropy. We mainly

differ from them in the overall goal: in our playful approach, we do not maximize the number of global classifications but the number of successful assignments. We want the user to feel satsified and enjoy the experience by successfully classifying their items.

Figure 4 follows the example from the previous section for the assignment tasks. Here, Alice showed her ability to distinguish classes A and B with more than a 90% success rate. A new item is processed by the neural network and the most likely classes are those for which Alice is good. By contrast, Alice is not good for all for other classes. Thus, she is likely to be assigned to this item.

When $n_k = 2$ the maximization problem is NP-Hard [27]. When $n_k > 2$ there is no proof that it is NP-Hard nor is there an exact known polynomial solution. Algorithm 1 presents a heuristic providing an approximate solution (inspired from Dantzig's greedy approximation of the knapsack problem [9]). The main idea of the heuristic is to associate a cost for each annotator equal to the number of classes where $\pi_{jj}^{(k)} > t$. Here, t represents a threshold which indicates if an annotator will be likely to correctly classify an item. Thus, each item will be assigned to the annotators who can classify that item and as few other items as possible. Experts, who can identify many different classes and are considered "expensive", are reserved for the difficult tasks.

Algorithm 1: Task Assignment Heuristic

Data: annotators U, items N, an acceptance threshold t
Result: the items assigned to the annotators

```
to_assign ← all items in N;
for all annotators u do
    compute cost of u;
end
while to_assign is not empty do
    for all item i do
        u ← select cheapest user for i with pi_ik > t, x_ik = 0 and that does not break a constraint;
        if u is not null then
            x_ik = 1;
        else
            remove i from to_assign;
        end
    end
end
```

Our experiments have shown that this heuristic achieves 90% of the score of the optimal assignment – computed using open sources optimizers – while being much faster. Indeed, the

heuristic does not even need to process all the data to perform task assignments while a huge cost of the optimal strategy is to create the optimisation problem (*e.g.* the objective function must consider all annotator/item pairs). In addition, the optimizers we used rely on a *Branch and Bound* (B&B) algorithm which is known to have an exponential worst case complexity while the complexity of our heuristic is linear in the number of users and items. Note however that the performance of our algorithm depends on the previously introduced function f and on the constraint t. If f tends to be uniform and/or t is low, then the algorithm will tend to assign very difficult tasks as they appear to users that will likely not be able to perform correct classification. By contrast, if f favors strong probabilities of success and/or t is high, then the algorithm may not be able to assign items to some users as their probability of success will be too low. However, any non-extreme values show to have good performance. In our experiments, t was set to 0.9, *i.e.* only tasks with a success probability superior than 90% can be assigned.

VI. Inference

In this section, we show how inference is performed using both the results of the training quiz and the annotators' classifications votes. Note that inference both estimates each user's confusion matrix in addition to estimating the label probabilities of each item (as well as the prior probability of each class). The effectiveness of our active training and assignment methods is thus closely related to the effectiveness of the inference method.

A. Bayesian Model

In crowdsourcing, a crucial issue is to combine the annotators' classification votes in a way that maximizes the likelihood that the system will find the correct classifications. In the example of Figure 5 which follows the previous examples, an item is associated to a probability distribution that shows that it likely belongs to class A or B. Alice proposed class A (that she is good at disambiguating). After the inference process takes into account the current probabilities, the users' confusion matrices and the votes, a new probability distribution is proposed where class A has a 0.98 chance of being the correct one.

In addition to the posterior probability of each item, the user confusion matrices are also updated to maximize the likelihood.

The inference process is run periodically (every 5 minutes in our implementation), thus user confusion matrices and label probabilities are continuously updated.

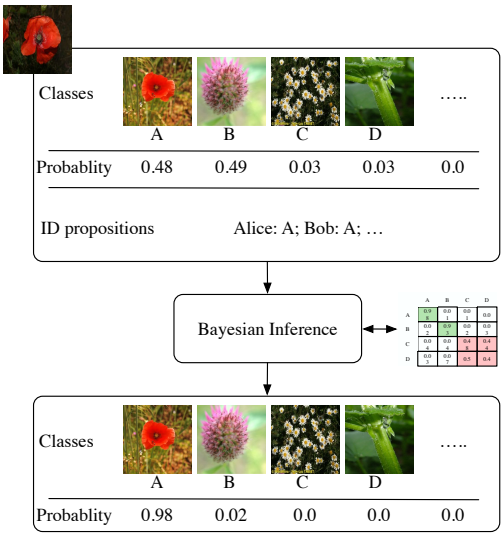

Fig. 5. Inference example. An item is associated to some probability distribution thanks to the convolutional neural network (CNN) and previous inferences. This probability distribution is then used together with the user confusion matrices and the user's assignment of class labels to infer a posterior probability.

Formally, we are given a set of classifiers K (annotators), a set of items D and a set of classes C. Each item in D is associated to an unknown true class in C and each classifier in K can propose a class in C for an item in D.

One popular way to infer the true class of each item is to use Bayesian probabilities and networks [10], [18], [25], [31].

Figure 6 presents the Bayesian network we use. It is based on the one proposed by Simpson *et al.* [31] and is similar to the one proposed by Kim *et al.* [18]. In such Bayesian networks, the goal is to find the true label t_i of all data items $1, ..., N$. The true label is supposed to be generated through a multinomial distribution where the probability of each class j is denoted κ_j. We are given a set of annotators who produce a set of classification votes $c_i^{(k)}$, where i is the data item index and k is the annotator's index. These classification votes are supposed to be generated through a multinomial distribution of parameters $\pi_j^{(k)}$ coming from the confusion matrix. That is,

$$\pi_{j,l}^{(k)} = p(c_i^{(k)} = l | t_i = j)$$

or, in other words, it is the probability that user k will give answer l while the true answer was j. Thus, even though some users may give incorrect classification votes, the system can still infer the true label. Finally, both $\pi_j^{(k)}$ and κ are conceptually generated through a Dirichlet distribution of parameters $\theta_j^{(k)}$ and ν respectively. Equation 2 presents the joint probability.

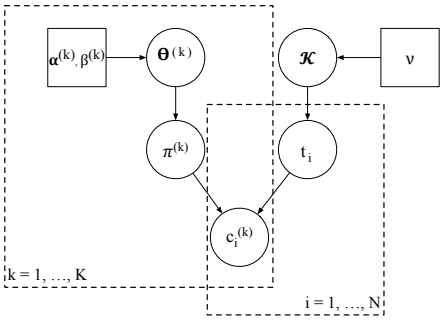

Fig. 6. Graphical model of the Bayesian network used to infer the correct classification with partial knowledge.

$$p(\kappa, \Pi, t, c|\theta_0, \nu) = \prod_{i=1}^{N}\{\kappa_{t_i} \prod_{k=1}^{K} \pi^{(k)}_{t_i, c_i^{(k)}}\} p(\kappa|\nu) p(\Pi|\theta_0) \tag{2}$$

Unfortunately, in large-scale classification, it is very likely that only partial knowledge is available. For instance, based on our active training quizzes concerning only a few classes for any given annotator/user, we may know that a user u does not confuse classes D and E but we may not know anything about that user's ability to distinguish D from F.

To represent this incompleteness, we propose the model shown in Figure 6 where we suppose that each $\theta^{(k)}_{jl}$ is generated through an independent gamma distributions of parameters $\alpha^{(k)}_{jl0} \in A$ and $\beta^{(k)}_{jl0} \in B$.

The joint probability thus becomes:

$$\begin{aligned} p(\kappa, \Pi, t, c, \theta|A, B, \nu) = &\prod_{i=1}^{N}\{\kappa_{t_i} \prod_{k=1}^{K} \pi^{(k)}_{t_i, c_i^{(k)}}\} p(\kappa|\nu) \\ &p(\Pi|\theta) p(\theta|A, B) \end{aligned} \tag{3}$$

The next subsection shows how we infer the parameters of the new model efficiently.

B. Solving the Inference with Thousands of Labels

The large dimensionality of the problem makes it difficult to run Bayesian inference. For instance, suppose that we are given $N = 1,000$ classes and $K = 1,000$ classifiers. Then, we may represent the confusion of all classifiers with a tensor composed of one billion cells, making it infeasible to run a full Bayesian inference in reasonable time. We focused on two approaches to address this problem.

1) Variational Bayes: First, we use a variational approach (inspired from [31]) because it is known to be efficient in computation time compared to Gibbs sampling and offers better results than the popular *EM* method [31].

Unfortunately, in our model, the matrices $\pi^{(k)}$ are generated through Dirichlet distributions while the matrices $\theta^{(k)}$ are generated through independent Gamma distributions which are not conjugate with the Dirichlets. This renders the variational equations no longer tractable. To address this issue, our proposed solution is to approximate the posterior probability of the non conjugate branch of the Bayesian network using the Laplace approximation Method [37] (Laplace's method approximates integrals of exponentials using Taylor's theorem). Once all variational equations have been derived, they form a circular dependency allowing to solve the inference through the procedure described in Algorithm 2. More details are given in appendix A.

Algorithm 2: Variational Bayesian Algorithm

1 Initialize $\mathbb{E}[\log \pi_{j,l}^{(k)}]$;
2 Initialize $\mathbb{E}[\log \kappa_j]$;
3 **repeat**
4 Compute $q^\star(t)$ from Equation 11;
5 Compute $q^\star(\kappa)$ from Equations 12 and 15;
6 Find $\widehat{\theta}$ by minimizing Equation 26;
7 Compute $q^\star(\pi)$ from Equations 13 and 30;
8 **until** *Convergence*;

In Algorithm 2, q refers to the variational approximation of the probability function p. Hence, $q^\star(t)$ is the approximation of the posterior probability of the variable t. Unlike the approach of Simpson *et al.* [31], we update the posterior probability of θ (line 6) before updating π while directly using the initialization values. In particular, one can notice that when the variance of our gamma priors tends toward 0 our model is the same as that proposed by Simpson *et al.* – if the variance tends toward 0, then θ is fixed. Thus our model is a generalization of their method having the advantage that it can be used for several sources of uncertainty in the classification (in our case, the partial knowledge of the possible classes during training).

2) Exploiting sparsity: As we work on thousands of classes, we are actually dealing with sparse data: two classes j and l may never be confused by any classifier (because they have never been presented together) and $\pi_{jl}^{(k)}$ may be considered equal to 0 or $\epsilon \to 0$. In our scenario, with over $1,000$ classes and more than $1,000$ annotators, we have observed that less than 2% of the possible confusions appear at all. The main benefit of that sparsity is to use

a sparse representation factorization as empty cells will end up with the same value and can thus be processed as a single variable. Therefore, instead of computing sparse cells separately, one just has to compute them once according to their factorization. The details are available in appendix A. This approach significantly reduces the memory demands of the model as well as its computation time.

VII. Experiments

In this section we present the experimental evaluation of our approach. Subsection VII-A introduces the setup of our experiments and subsection VII-B presents the results.

A. Setup

To evaluate our contributions, we implemented the framework architecture of Figures 1 & 2 within a game platform called *The Plant Game*[3]. We used the dataset released for the LIFE-CLEF2015 [4] contest. We kept all plant observations where at least one picture of the flower was available. We thus had a total of $6,300$ plant observations to classify. In addition we had a training set composed of $13,900$ observations. Note that this training dataset is the same as the one used to train the Convolutional Neural Network. Each single observation is composed of at least one picture but could consist of several images of different organs (*e.g.* entire view, leaf, branch, flower, fruit) of the same plant. The test and the training sets correspond to $1,000$ different plant species – *classes* in the paper. These species correspond to the most popular ones from the Pl@ntnet data stream associated to the Western European flora. $1,107$ annotators participated to the experiment.

To play, participants had to establish an account where he/she had to self-evaluate his/her current expertise (*i.e.* beginner, intermediate, expert); this information is notably used to fit the initial threshold of the difficulty of the training quizzes recommended to him/her. Figure 7 represents the distribution of self-declared expertise. Most of our players self-identified as beginners.

The Plant Game offers several game modes:

1) **Training:** allow a user to be trained as described in Section IV,
2) **Assignment:** this is the user interface where each annotator can see the tasks/observations that are assigned to him using the different strategies,

[3]http://theplantgame.com

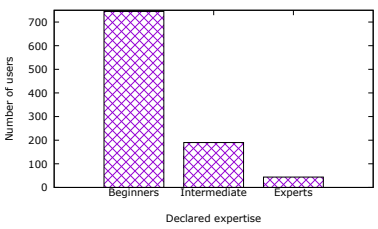

Fig. 7. Plant Game users' statistics.

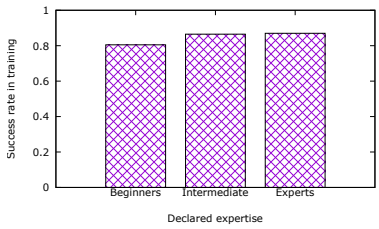

Fig. 8. Users' average success rate in training quiz per declared expertise. The success rate does not vary very much, because the assigned quizzes are simpler for less expert users.

3) **Ranking:** to boost user motivation, all users can see how they perform with respect to other users in the *Ranking* view.

We compared three strategies to obtain plant identifications:

1) observations are randomly assigned and users have not been trained on them beforehand,
2) Users have been trained (cf. Section IV) but observations are randomly assigned,
3) Users have been trained (cf. Section IV) and observations are assigned as discussed in Section V.

In addition to training and assignment we also analyse the two following aspects:

1) **Inference effectiveness**: we compare three classification inference methods. First, *majority voting* (**MV**) which is the baseline. It is the easiest to implement and offers good results. Then, *Variational Independent Bayesian classifier combination* named (**VIBCC**) which provides state-of-the-art performance [31]. Finally, our updated model (**VIBCC2**) taking into account the partial knowledge of the annotators' confusion.
2) **Overall classification performance**: we show the quality gain compared to the fully automated machine learning-based algorithm used as input of our crowdsourcing system.

In addition, we computed the p-value to evaluate the statistical significance of training and adaptive assignment. When doing so, the null hypothesis corresponds to "the contribution in question (training or adaptive assignment) does not change the results" (*e.g.* random assignment and skills-aware assignment performs similarly). A high p-value means that it is not possible to reject the null hypothesis.

Our results are structured in three sections, each describing one of the three contributions: active training, assignment (and the effect of retraining) and inference. Also, a final analysis of

the overall workflow is provided.

B. Results

We present here the results of our experiments. Figure 9 shows a global overview of the benefits of our contributions. The figure will be discussed in detail at the end, but shows that the benefits are practically and statistically significant.

1) Active Training: Here we evaluate the impact of the users' training. Figure 9 clearly shows that plant identification is essentially an impossible task for untrained annotators. Additionally, the number of correct answers is so small that no classification method actually outperforms the others. Notice, however, that although the 2.5% accuracy for untrained users is bad, it is still significantly better than a random classifier (*i.e.* 0.001% accuracy). On the other hand, when the users have the option of being trained, their accuracy reaches 37% (with random assignment) and 80% (with optimized assignment).

Figures 10 and 11 show the benefit of training by comparing the evolution of the success rate and of the response time per user depending on the number of the question asked in a quiz – between 1 and 30. These figures clearly show that the users become more and more confident after acquiring training. Indeed, they answer faster and faster and increase their average success rate. In average, at question 30 the users take 6 seconds (with a standard deviation of $\sigma = 2.53$) and identify the species with a success rate of 82% ($\sigma = 0.4$).

In Figure 8 we can see that the average individual success rate of each annotator ranges from 80% (with a high standard deviation of $\sigma = 0.36$) to 94% (with a small standard deviation of $\sigma = 0.1$). The small gap between the different annotators' categories means that they have been trained on observations corresponding to their level of expertise – although beginners tend to have a high variability. Thus, beginners have an average success rate of 80% while experts reach 86%. The high variance of the beginners' results confirm that they tend to answer at random when they don't know the species yet in contrast to experts who refuse to answer in such cases.

In Figure 12, we observe the evolution of the average success rate of the users when they played three successive quizzes with more and more possible species in the answers (*i.e.* 6, 8 and 10). We can observe that within a quiz, the more they play, the better they become. However, as soon as they start a new quiz with more choices, their success rate decreases before increasing again to a higher level. This is due to the fact that new species were added and cognitive biases

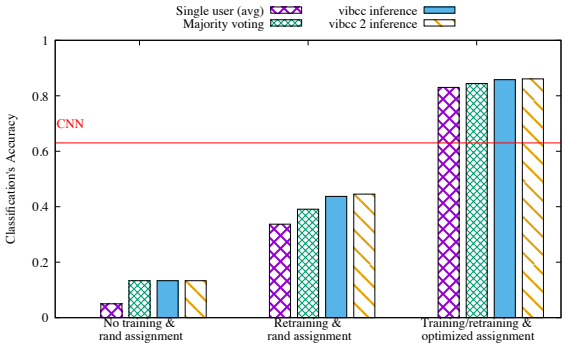

Fig. 9. Classification quality for several assignment and training approaches. (1) The user is not trained and plant assignments/observations are chosen randomly. (2) The user has been trained on disambiguating some species, however, assignments are still chosen randomly. (3) the user is trained and he or she is assigned observations that he or she is likely to identify correctly. When comparing the results in going from (1) to (2) or (2) to (3), the $p - val < 1/10000$ indicating that training and adaptive assignment bring statistically significant benefits.

are reduced (*cf.* Section IV). At the end of the third session (*i.e.* 90 questions asked), the average success rate per user reaches nearly 90% ($\sigma = 0.362$).

Finally, Figure 13 shows that the more a single user trains on a single species the better he/she becomes. Specifically, the success rate rises from 68% ($\sigma = 0.468$) to almost 84% ($\sigma = 0.424$).

Training alone is not sufficient when assignments are random. Figure 9 also shows that when the users only have the possibility to retrain on their assignment (*i.e.* retraining & rand assignment), they still have a moderate accuracy of around 33.7% ($\sigma = 0.26$). This is much better than without any training but, it still does not outperform the automatic *CNN* classification. Assignment is thus crucial to boost performance.

2) Assignment: Figure 9 shows that when using task assignment (*i.e.* assign items to users to maximize the likelihood of success), the users have an individual success rate of more than 80% ($\sigma = 0.309$).

Note that around 13% of the time, the users were not capable to identify the species on the first try and had to perform a retrain quiz to recall how to classify the item (even though they were already trained on it). When doing so, they reached an accuracy of 90.66% ($\sigma = 0.283$). Notice that users who are willing to retrain obtain better results than those who think they know the answer.

We can observe in Figure 14 that the identification success rate does not increase with the annotators' declared expertise. Beginners actually have approximately the same success rate as

more experienced identifiers providing that they have been correctly trained and have been given classification tasks adapted to their skills.

However, the high variance of the results obtained here confirms the need of redundant votes for each observation and effective aggregation methods.

3) Inference: We now compare the different inference methods: majority voting MV, Variational Independent Bayesian Classifier Combination methods (*VIBCC*, state-of-the-art) and *VIBCC 2* (ours). Figure 9 first shows that our new inference method has a better accuracy than the state-of-the-art method in all training and assignment scenarios. However, the gap is too small to reject the null-hypothesis (*i.e.* VIBCC 2 performs similarly to VIBCC).

To determine possible secondary benefits of our inference method, we conducted further analysis of the results as reported in Table I. In particular, we measured the mean average precision instead of the brute force top-1 accuracy, so as to reflect the relevance of the estimated probabilities that are crucial for the active training and the optimized assignment. We also recomputed the inference of the three methods using fixed number of votes per observations (*i.e.* 2, 3, 4 and $4+$). For each number of votes, we did a k-fold inference with $1,000$ random observations each time with $k = 6$. Based on the results we computed the p-value with the null hypothesis being that *VIBCC 2* performs similarly to *VIBCC* or *MV* – ns stands for not significant when it is not possible to reject the null hypothesis.

When the number of distinct votes is low, *VIBCC 2* outperforms *VIBCC*. When the number of votes reaches 4 (or more), both method seem to converge (the p-value is no longer significant after 4 votes per observation). It can be shown analytically that when the size of the dataset is big enough (*i.e.* in terms of redundancy per species and users), both methods converge to the same result at a ratio that depends on how the priors of the models has been set.

All methods degrade when the number of votes increases, but Bayesian methods less so. The reason is that more votes reflects more uncertainty (more classes are possible). The results clearly show that when the problem becomes difficult and requires more votes, Bayesian approaches outperform the simple majority voting in a statistically significant way.

4) Global Framework: Figure 9 shows the combination of inference method, training and assignment that attains the highest classification quality. The majority of the gain is due to the combination of active training and adaptive assignment. By contrast, the inference algorithm matters little. Figure 15 contains the precision results of the observations considered as "validated" (*i.e.* probability of one class higher than a threshold set to 99%). It shows the impressive

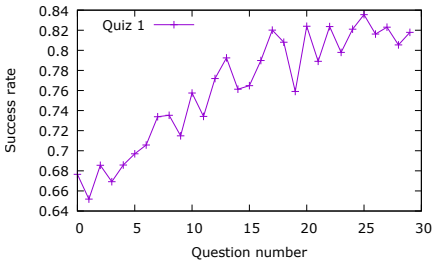

Fig. 10. Average success rate in Quiz 1.

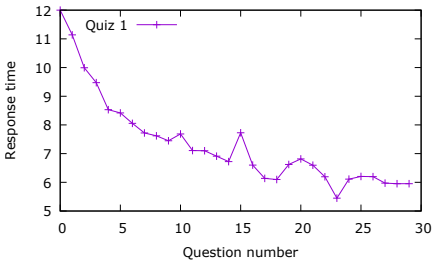

Fig. 11. Average response time in Quiz 1.

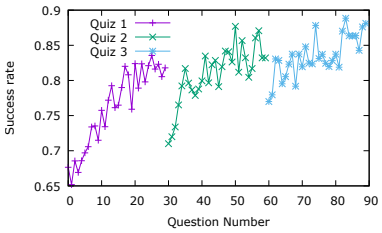

Fig. 12. Average success rate per quiz (*i.e.* 1, 2 & 3).

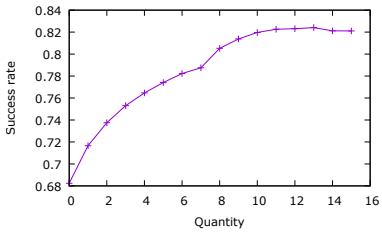

Fig. 13. Average success rate per species.

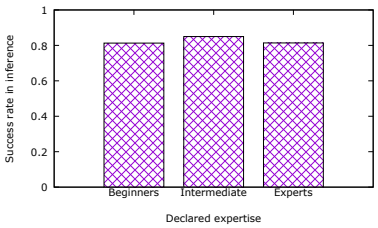

Fig. 14. Average identification success rate per declared expertise.

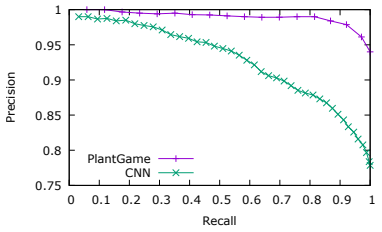

Fig. 15. Recall precision: Human VS Machine.

gain of *The Plant Game* compared to the standalone *Convolutional Neural Network* used as the input of our framework (*i.e.* GoogleNet [32] fine tuned on LifeClef's data [8]). For instance, at the point of 90% recall, *the Plant Game* obtains a precision of 98% while the *CNN* alone only obtains 85%. In other words, when we consider that the platform has correctly classified an item, classification quality is much higher than using machine learning alone.

Thus, adaptive training-assisted crowdsourcing vastly reduces the uncertainty of automated predictions for this application and we suspect for any multi-class domain where each class demands high expertise.

TABLE I

MEAN AVERAGE PRECISION GAIN WITH P-VALUE (NS STANDS FOR NOT SIGNIFICANT).

votes/obs	VIBCC 2	VIBCC (p-val)	MV (p-val)
2	**0.864**	0.8529 (0.02)	0.85 $(2.5.10^{-3})$
3	**0.884**	0.8734 (0.012)	0.831 $(< 1/10000)$
4	0.857	$\mathbf{0.857}$ $\mathbf{(ns)}$	0.748 $(< 1/10000)$
4+	**0.848**	0.844 (ns)	0.752 $(< 1/10000)$

VIII. CONCLUSION

In this paper we have presented a framework for crowdsourced classification in the context of thousands of complex classes. Our strategy is built on three ideas:

First users can be trained on how to disambiguate classes even if they had absolutely no knowledge about the task at first. Training on demand appears to help even further.

Second, data items should be assigned to users according to their acquired skills. This allows non-experts to obtain the same success rate as experts.

Finally, a generalized version of the Variational Bayesian Independent Classifier Combination introduced in [31] better takes into account the uncertainty introduced by sparse data. The sparseness arises because annotators only train on a very small subset of the thousands of specialized classes. Therefore, only partial knowledge is available.

Overall, our experiments clearly show the effectiveness of our framework for achieving correct predictions of complex tasks such as plant identification.

Future work may address the problem of how to continuously train the *CNN* based on the newly validated data and how to improve users training through more accurate modeling of their learning capacity and forgetfulness.

IX. ACKNOWLEDGMENTS

We would like to particularly thanks Jérémy Paul for the design of The Plant Game that helped us acquire enough players. Also we would like to thanks Pierre Fernique for his insights. Finally, Dennis Shasha's work was supported by an INRIA international chair.

REFERENCES

[1] S. Basu and J. Christensen. Teaching classification boundaries to humans. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pages 109–115. AAAI Press, 2013.

[2] S. Basu Roy, I. Lykourentzou, S. Thirumuruganathan, S. Amer-Yahia, and G. Das. Task assignment optimization in knowledge-intensive crowdsourcing. *The VLDB Journal*, 24(4):467–491, 2015.

[3] J.-I. Biel and D. Gatica-Perez. The youtube lens: Crowdsourced personality impressions and audiovisual analysis of vlogs. *IEEE Transactions on Multimedia*, 15(1):41–55, 2013.

[4] P. Bonnet, W.-P. Vellinga, R. Planqué, A. Rauber, S. Palazzo, B. Fisher, and H. Müller. Lifeclef 2015: Multimedia life species identification challenges. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 6th International Conference of the CLEF Association, CLEF'15, Toulouse, France, September 8-11, 2015, Proceedings*, volume 9283, page 462. Springer, 2015.

[5] K. Borne and Z. Team. The zooniverse: A framework for knowledge discovery from citizen science data. In *AGU Fall Meeting Abstracts*, volume 1, page 0650, 2011.

[6] J. Bragg, Mausam, and D. S. Weld. Optimal testing for crowd workers. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 966–974, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems.

[7] J. Champ, A. Joly, and P. Bonnet. Fine-grained visual faceted search. In *Proceedings of the ACM International Conference on Multimedia*, pages 721–722. ACM, 2014.

[8] J. Champ, T. Lorieul, M. Servajean, and A. Joly. A comparative study of fine-grained classification methods in the context of the lifeclef plant identification challenge 2015. In *CLEF 2015*, volume 1391, 2015.

[9] G. B. Dantzig. Discrete-variable extremum problems. *Operations research*, 5(2):266–288, 1957.

[10] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[12] M. Fang, X. Zhu, B. Li, W. Ding, and X. Wu. Self-taught active learning from crowds. In *2012 IEEE 12th International Conference on Data Mining*, pages 858–863. IEEE, 2012.

[13] L. Gottlieb, G. Friedland, J. Choi, P. Kelm, and T. Sikora. Creating experts from the crowd: Techniques for finding workers for difficult tasks. *IEEE Transactions on Multimedia*, 16(7):2075–2079, 2014.

[14] P. Ipeirotis. Crowdsourcing using mechanical turk: quality management and scalability. In *Proceedings of the 8th International Workshop on Information Integration on the Web: in conjunction with WWW 2011*, page 1. ACM, 2011.

[15] A. Joly, P. Bonnet, H. Goëau, J. Barbe, S. Selmi, J. Champ, S. Dufour-Kowalski, A. Affouard, J. Carré, J.-F. Molino, N. Boujemaa, and D. Barthélémy. A look inside the Pl@ntNet experience. *Multimedia Systems*, page 16, 2015.

[16] A. Joly, H. Goëau, P. Bonnet, V. Bakić, J. Barbe, S. Selmi, I. Yahiaoui, J. Carré, E. Mouysset, J.-F. Molino, et al. Interactive plant identification based on social image data. *Ecological Informatics*, 23:22–34, 2014.

[17] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. *AAMAS '12 Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pages 467–474, 2012.

[18] H.-C. Kim and Z. Ghahramani. Bayesian classifier combination. In *International conference on artificial intelligence and statistics*, pages 619–627, 2012.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] D. Liu, S. Yan, X.-S. Hua, and H.-J. Zhang. Image retagging using collaborative tag propagation. *IEEE Transactions on Multimedia*, 13(4):702–712, 2011.

[21] A. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *Proceedings of the VLDB Endowment*, 4(5):267–278, 2011.

[22] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: Algorithms for filtering data with humans. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 361–372. ACM, 2012.

[23] A. Quinn, B. Bederson, T. Yeh, and J. Lin. CrowdFlow: Integrating machine learning with mechanical turk for speed-cost-quality flexibility. *Better performance over iterations*, 2010.

[24] H. Rahman, L. Joppa, and S. B. Roy. Feature based task recommendation in crowdsourcing with implicit observations. *CoRR*, abs/1602.03291, 2016.

[25] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *The Journal of Machine Learning Research*, 11:1297–1322, 2010.

[26] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[27] S. Sahni and T. Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, July 1976.

[28] J. Sang, C. Xu, and J. Liu. User-aware image tag refinement via ternary semantic analysis. *IEEE Transactions on Multimedia*, 14(3):883–895, 2012.

[29] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM, 2008.

[30] E. Simpson and S. Roberts. Bayesian methods for intelligent task assignment in crowdsourcing systems. In V. T. Guy, M. Kárný, and H. D. Wolpert, editors, *Decision Making: Uncertainty, Imperfection, Deliberation and Scalability*, pages 1–32. Springer International Publishing, Cham, 2015.

[31] E. Simpson, S. Roberts, I. Psorakis, and A. Smith. Dynamic bayesian combination of multiple imperfect classifiers. In *Decision Making and Imperfection*, pages 1–35. Springer, 2013.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[33] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

[34] L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 901–908, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[35] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann. Review of classifier combination methods. *Studies in Computational Intelligence*, 90(Figure 1):361–386, 2008.

[36] M. Venanzi, J. Guiver, and G. Kazai. Community-based bayesian aggregation models for crowdsourcing. *Proceedings of the 23rd ...*, 22:63–87, 2014.

[37] C. Wang and D. M. Blei. Variational Inference in Nonconjugate Models. *Journal of Machine Learning Research*, 14:1005–1031, 2013.

[38] P. Welinder and P. Perona. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, pages 25–32, 2010.

APPENDIX

A. Variational Model

This family of models is used to approximate the computation of intractable integrals. In our particular context, it consists in finding a tractable distribution q minimizing the *KL* divergence with respect to the true posterior distribution:

$$\begin{aligned} D_{KL}(P||Q) &= \int q(z)\log\frac{q(z)}{p(z|x)} \\ &= \int q(z)\log\frac{q(z)}{p(z,x)} + \log\ p(x) \\ \log\ p(x) &= D_{KL}(P||Q) + \mathcal{L}(Q) \end{aligned} \tag{4}$$

Where $\mathcal{L}(Q)$ is called the variational free energy. The distribution $q(Z)$ is generally assumed to factorize among its latent variables and parameters:

$$q(Z) = \prod_{i=1}^{M} q_i(Z_i|X) \tag{5}$$

It can be shown using the calculus of variations that the best form of q is as follows:

$$\log\ q_i(Z_i|X) = \mathbb{E}_{j\neq i}[\log\ p(Z,X)] + const \tag{6}$$

When developing this equation, the fact that the priors are chosen to be conjugated imply that $q_i(Z_i)$ will actually decompose to the same form as its true posterior. Then, circular dependencies will appear among the different variational equations. These dependencies actually describe the algorithm that will iterate until convergence. Convergence can be evaluated using the variational free energy.

Recall that the joint probability of our model is:

$$\begin{aligned} p(\kappa,\Pi,t,c,\theta|A,B,\nu) = &\prod_{i=1}^{N}\{\kappa_{t_i}\prod_{k=1}^{K}\pi^{(k)}_{t_i,c_i^{(k)}}\}p(\kappa|\nu) \\ &p(\Pi|\theta)p(\theta|A,B) \end{aligned} \tag{7}$$

Similarly to other variational Bayesian approaches (*cf.* Appending A), the form of our variational approach factorizes in the following way:

$$q(\kappa,t,\Pi,\theta) = q(t)q(\kappa)q(\Pi)q(\theta) \tag{8}$$

As we chose a nonconjugate prior over the Dirichlet used to generate the confusion matrices of each annotator, it is however non trivial to determine all the variational equations as in the classical case

B. Items and Classes distributions (Conjugate branch of the network)

Each part of this equation will now be evaluated, starting with the estimator of $q(t)$.

$$\log\ q^{\star}(t) = \mathbb{E}_{\kappa,\Pi}[\log\ p(\kappa, t, \Pi, c)] + const \tag{9}$$

For simplification and similarly to Simpson *et al.* [31], we use the following notation:

$$\begin{aligned}\log\ \rho_{i,j} =& \mathbb{E}_{\kappa_j,\pi_j}[\log\ p(\kappa_j, t_i, \pi_j, c_i)] \\ =& \mathbb{E}_{\kappa_j}[\log\ \kappa_j] + \sum_{k=1}^{K} \mathbb{E}_{\pi_j}[\log\ \pi^{(k)}_{j,c_i^{(k)}}]\end{aligned} \tag{10}$$

Based on ρ, the probability of a true label given an object i can be estimated in the following way:

$$q^{\star}(t_i = j) = \mathbb{E}_t[t_i = j] = \frac{\rho_{i,j}}{\sum_{\iota=1}^{J} \rho_{i,\iota}} \tag{11}$$

Equation 11 is the typical posterior probability of t_i.

For simplification we use the following notations:

$$N_j = \sum_{i=1}^{N} \mathbb{E}_t[t_i = j] \tag{12}$$

which corresponds to the expected number of occurrences of each true class. Similarly, we compute the expected number of rejects (*i.e.* the parameters β) and of accepts (*i.e.* the parameters α) in each true class per user.

$$N_{jl}^{(k)} = \sum_{i=1}^{N} \delta_{c_i^{(k)} l} \mathbb{E}_t[t_i = j] \tag{13}$$

where $\delta_{c_i^{(k)} l}$ is the dirac function which equals 1 when $c_i^{(k)} = l$.

The development of $q^{\star}(\kappa)$ follows what Simpson *et al.* proposed using the fact that κ is generated through a Dirichlet distribution:

$$\begin{aligned}\log\ q^{\star}(\kappa) =& \mathbb{E}_t[\sum_{i=1}^{N}\sum_{j=1}^{J} \log\ \kappa_{t_i,j}] + \log\ p(\kappa|\nu_0) + const \\ =& \sum_{j=1}^{J} N_j \log\ \kappa_j + \sum_{j=1}^{J} (\nu_{0,j} - 1) \log\ \kappa_j + const \\ =& \sum_{j=1}^{J} (N_j + \nu_{0,j} - 1) \log\ \kappa_j + const\end{aligned} \tag{14}$$

We can see in Equation 14 that we obtain a posterior Dirichlet distribution:

$$q^{\star}(\kappa) \propto Dir(\kappa|\nu_1, ..., \nu_J) \tag{15}$$

where ν is updated in the following manner:

$$\nu_j = \nu_{0,j} + N_j \tag{16}$$

We can now compute the expectation of log κ_j:

$$\mathbb{E}[\log\ \kappa_j] = \psi(\nu_j) - \psi(\sum_{\iota=1}^{L} \nu_\iota) \tag{17}$$

where ψ is the digamma function.

C. Users Distributions (Nonconjugate branch of the network)

In order to approximate the posterior distribution of the parameters θ using Laplace method, several assumptions are necessary as presented in [37].

1) First, θ must be real valued which is the case in our context and $p(\theta|A, B)$ must be twice differentiable with respect to θ which is the case of the gamma distribution.
2) Second, $p(\pi_j^{(k)}|\theta_j^{(k)})$ must be in the exponential family, which is the case of the Dirichlet distribution and therefore can be expressed as:

$$\begin{aligned} p(\pi_j^{(k)}|\theta_j^{(k)}) = h(\pi_j^{(k)})\text{exp}\{\eta(\theta_j^{(k)}))^T t(\pi_j^{(k)}) \\ - a(\eta(\theta_j^{(k)}))\} \end{aligned} \tag{18}$$

   where $h(\pi_j^{(k)})$ is a function called *base measure*, $t(\pi_j^{(k)})$ is the *sufficient statistic*, $\eta(\theta_j^{(k)})$ is the natural parameters and $a(\eta(\theta_j^{(k)}))$ is the log partition function. $\eta(\theta_j^{(k)})$ is assumed to be twice differentiable: this is the case of the Dirichlet. In our context, this elements take the following values:

   - $h(\pi_j^{(k)}) = 1$
   - $t(\pi_j^{(k)}) = [\log\ \pi_{j1}^{(k)}, ..., \log\ \pi_{jJ}^{(k)}]^T$
   - $\eta(\theta_j^{(k)}) = [\theta_{j1}^{(k)} - 1, ..., \theta_{jJ}^{(k)} - 1]^T$
   - $a(\eta(\theta_j^{(k)})) = \sum_i \log\Gamma(\theta_{ji}^{(k)}) - \log\Gamma(\sum_i \theta_{ji}^{(k)})$

3) Finally, we also assume $p(c_i^{(k)} = j|\pi_j^{(k)}))$ to be in the exponential family which is the case of the multinomial:

$$\begin{aligned} p(c_i^{(k)} = j|\pi_j^{(k)})) = h(c_i^{(k)})\text{exp}\{\eta(\pi_j^{(k)}))^T t(c_i^{(k)}) \\ - a(\eta(\pi_j^{(k)}))\} \end{aligned} \tag{19}$$

   where the important parameters can be described as follows:

   - $t(c_i^{(k)}) = [c_i^{(k)} = 1, ..., c_i^{(k)} = J]^T$
   - $\eta(\pi_j^{(k)}) = [\log\ \pi_{j1}^{(k)}, ..., \log\ \pi_{jJ}^{(k)}]^T$
   - $a(\eta(\pi_j^{(k)})) = 0$

   Notice that $\eta(\pi_j^{(k)}) = t(\pi_j^{(k)})$. Thus equation 19 can be reformulated as follows:

$$p(c_i^{(k)} = j|\pi_j^{(k)})) = h(c_i^{(k)})\text{exp}\{t(\pi_j^{(k)})^T t(c_i^{(k)}) \tag{20}$$

We now estimate $q^\star(\theta)$ using Laplace method such as presented in [37].

$$\begin{aligned} q^\star(\theta_j^{(k)}) \propto & \exp\{\eta(\theta_j^{(k)}))^T \mathbb{E}_{\pi_j^{(k)}}[t(\pi_j^{(k)})] \\ & - a(\eta(\theta_j^{(k)})) + \log\ p(\theta_j^{(k)})\} \\ \propto & \exp\{f(\theta_j^{(k)})\} \end{aligned} \tag{21}$$

$$\begin{aligned} f(\theta_j^{(k)}) \triangleq & \eta(\theta_j^{(k)}))^T \mathbb{E}_{\pi_j^{(k)}}[t(\pi_j^{(k)})] \\ & - a(\eta(\theta_j^{(k)})) + \log\ p(\theta_j^{(k)}) \end{aligned} \tag{22}$$

We can now approximate f using the second-order Taylor approximation around the value of $\theta_j^{(k)}$ that maximizes the function which we note $\widehat{\theta_j^{(k)}}$.

$$\begin{aligned} f(\theta_j^{(k)}) \approx & f(\widehat{\theta_j^{(k)}}) + \nabla f(\widehat{\theta_j^{(k)}})(\theta_j^{(k)} - \widehat{\theta_j^{(k)}}) \\ & + \frac{1}{2}(\theta_j^{(k)} - \widehat{\theta_j^{(k)}})^T \nabla^2 f(\widehat{\theta_j^{(k)}})(\theta_j^{(k)} - \widehat{\theta_j^{(k)}}) \end{aligned} \tag{23}$$

Notice that since $\widehat{\theta_j^{(k)}}$ maximizes f, then $\nabla f(\widehat{\theta_j^{(k)}}) = 0$.

Then,

$$\begin{aligned} q^\star(\theta_j^{(k)}) \propto & \exp\{f(\widehat{\theta_j^{(k)}}) + \frac{1}{2}(\theta_j^{(k)} \\ & - \widehat{\theta_j^{(k)}})^T \nabla^2 f(\widehat{\theta_j^{(k)}})(\theta_j^{(k)} - \widehat{\theta_j^{(k)}})\} \end{aligned} \tag{24}$$

$$q^\star(\theta_j^{(k)}) \approx \mathcal{N}(\widehat{\theta_j^{(k)}}, -\nabla^2 f(\widehat{\theta_j^{(k)}})^{-1}) \tag{25}$$

The Gaussian form of our approximation results from the Taylor approximation. Recall that we did the approximation around the value $\widehat{\theta_j^{(k)}}$ which maximizes the function f. Thus we still need to compute $\widehat{\theta_j^{(k)}}$. To do so, we first develop f.

$$\begin{aligned} f(\theta_j^{(k)}) = & (\theta_j^{(k)})^T \mathbb{E}_{\pi_j^{(k)}}[t(\pi_j^{(k)})] \\ & - \sum_i \log\Gamma(\theta_{ji}^{(k)}) - (\log\Gamma(\sum_i \theta_{ji}^{(k)})) \\ & + \sum_i \{(\alpha_{ji}^{(k)} \log\ \theta_{ji}^{(k)} - \beta_{ji}^{(k)} \theta_{ji}^{(k)}\} \end{aligned} \tag{26}$$

where

$$\mathbb{E}_{\pi_j^{(k)}}[t(\pi_j^{(k)})_l] = \psi(\pi_{jl}^{(k)}) - \psi(\sum_i \pi_{ji}^{(k)}) \tag{27}$$

Here, we use a numerical method (*i.e.* Gradient Descent) to find $\widehat{\theta_j^{(k)}}$.

Once $\widehat{\theta_j^{(k)}}$ has been estimated, we can develop $q^\star(\pi_j^{(k)})$.

$$\begin{aligned} \log\ q^\star(\pi_j^{(k)}) = & \log\ p(c^{(k)}|t=j,\pi_j) + \log\ h(\pi_j^{(k)}) \\ & + \mathbb{E}_{\theta_j^{(k)}}[\eta(\theta_j^{(k)})]^T t(\pi_j^{(k)}) + C \end{aligned} \tag{28}$$

Which can be reformulated using Equation 20 as:

$$q^{\star}(\pi_j^{(k)}) \propto h(\pi_j^{(k)})\exp\{(\eta(\theta_j^{(k)})] + t(c))^T t(\pi_j^{(k)})\} \tag{29}$$

$$q(\pi_j^{(k)}) = \text{Dir}(\pi_j^{(k)}|\theta_{j1}^{(k)}, ..., \theta_{jJ}^{(k)}) \tag{30}$$

Where

$$\theta_{jl}^{(k)} = \widehat{\theta}_{jl}^{(k)} + N_{jl}^{(k)} \tag{31}$$

D. Sparsity Discussion

In this subsection, we present how we dealt with data sparsity to speed up the inference's algorithm. As a preliminary, one should consider that all variables are initialized with default value superior to 0 to avoid wrong inferences (*e.g.* an annotator never confused two classes but start to confuse them at some point).

First, in Π, some confusions do not exist. Thus, if we only need to compute the probabilities $\pi_{j,l}^{(k)}$ on existing confusion as they will be the only one useful when inferring the true class of an item i.

Second, when estimating θ, the problem is more complex as we proceed to a maximization to find its value. Hence, the solution is to factorize all sparse cells in equation 26. Then, since all this cells will end up with the same value, we can consider them as a single variable:

A sparse matrix implementation may consider default values for sparse cells on each line and benefit from this optimization.