“Conversing Learning as a Crowd-Powered Approach to Diminish the Effects of Semantic Drift in Bootstrap Learning Algorithms”

ALUNO: Saulo Pedro
ORIENTADOR: Prof. Dr. Estevam R. Hruschka Jr.

São Carlos
Agosto 2019

Saulo Pedro

Conversing Learning as a Crowd-Powered Approach to Diminish the Effects of Semantic Drift in Bootstrap Learning Algorithms

Thesis presented to the Programa de Pós-Graduação em Ciência da Computação of Universidade Federal de São Carlos as part of the requirements to obtain the title of PhD in Computer Science.

Universidade Federal de São Carlos - UFSCar

Supervisor: Estevam R. Hruschka Jr

São Carlos
August 2019

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Tese de Doutorado do candidato Saulo Domingos de Souza Pedro, realizada em 21/08/2019:

[illegible]

Profa. Dra. Heloisa de Arruda Camargo
UFSCar

Prof. Dr. Estevam Rafael Hruschka Junior
UFSCar

Diego Silva

Prof. Dr. Diego Furtado Silva
UFSCar

Profa. Dra. Ana Paula Appel
IBM Research

Profa. Dra. Dunja Mladenic
IJS

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Estevam Rafael Hruschka Junior, Ana Paula Appel, Dunja Mladenic e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.

[illegible]

Profa. Dra. Heloisa de Arruda Camargo

Abstract

Internet and social Web made possible the acquisition of information to feed a growing number of Machine Learning (ML) applications and, in addition, brought light to the use of crowdsourcing approaches, commonly applied to problems that are easy for humans but difficult for computers to solve, building the crowd-powered systems. In this work, we consider the issue of semantic drift in a bootstrap learning algorithm and propose the novel idea of a crowd-powered approach to diminish the effects of such issue. To put this idea to test we built a hybrid version of the Coupled Pattern Learner (CPL), a bootstrap learning algorithm that extract contextual patterns from an unstructured text, and SSCrowd, a component that allows conversation between learning systems and Web users, in an attempt to actively and autonomously look for human supervision by asking people to take part into the knowledge acquisition process, thus using the intelligence of the crowd to improve the learning capabilities of CPL. We take advantage of the ease that humans have to understand language in unstructured text, and we show the results of using a hybrid crowd-powered approach to diminish the effects of semantic drift.

List of Figures

Acronyms

AI Artificial Intelligence. 10

AL Active Learning. 9–13

AMT Amazon's Mechanical Turk. 9, 14, 23, 39

API Application Programming Interface. 14, 16–20, 30, 39

CL Conversing Learning. 3, 9–21, 23, 24, 29, 39

CPL Coupled Pattern Learner. 2, 10, 13, 18, 28–35, 39

CS Computer Science. 10

IL Interactive Learning. 11

IR Information Retrieval. 13

KB knowledge base. 10, 13, 15, 16, 19, 23

MAS Multi-Agent System. 10, 15, 20, 23, 29, 39

ML Machine Learning. 2, 9–13, 15, 16, 20, 21, 23–25, 39

NELL Never-Ending Language Learner. 10, 13, 15, 16, 18, 19, 21, 28, 39

NLP Natural Language Processing. 20, 27, 28

QA Question Answering. 13, 17, 21

RL Rule Learner. 16

SVM Support Vector Machines. 25

Contents

1 Introduction

Machine Learning can be seen as an area of study focused on building computer systems that learn. Such systems are usually designed based on three main components (which where defined in [1]): i) a Task (T) to be executed; ii) a performance metric P, and iii) a source of Experience (E). A system is said to learn if it can improve its performance P on task T, after having access to experience E.

In ML systems, we usually want to understand a real world problem and use it to define a task T. Afterwards, we can teach a machine to resolve it. Many applications in Machine Learning depend on large unstructured datasets to be used as experience E, to help the learning process. Sometimes, even when E is based on structured data, developers cannot be sure whether the data is accurate or useful. When facing a problem like that, we usually want to have an oracle, that is, someone with an extensive knowledge of the problem to look at the data and select the best from it to compose the dataset. This approach is known as Active Learning (AL) [2, 3, 4] and it has been used to improve ML tasks by filtering irrelevant data, thus reducing the cost of labeling them.

During a Machine Learning system development, engineers often have to find a way to avoid overfitting, due to problems such as not having enough accurate data to feed the system and insufficient validation and revision methods. Such problems could be resolved by an oracle, but in many applications, this oracle must be a human and if the volume of data is too big, it may not be possible for the oracle to provide enough assistance in reasonable time. Such situation encourages the discussion of having an automatic agent assisting in ML system demands like validation, revision and feeding.

This scenario describes a situation where a problem can be easily resolved by human experts (an oracle) and poorly resolved by machines. It also motivates *human-in-the-loop* approaches [5, 6, 7] that consider adding humans to take part into the knowledge acquisition process of a Machine Learning system. In order to provide information with enough accuracy, in such *human-in-the-loop* approaches, agents would have to access human generated content and extract information from it automatically, whenever the system needs it. To make that happen, one possibility, which has not yet been explored in depth by Active Learning community, neither by Crowdsourcing community, would be to build an agent capable to identify its own needs for human-supervision, and based on these needs, proactively start conversation with people in social network environments, asking them to assist the intelligent machine. In this work, we propose such approach through a model of learning we call Conversing Learning.

Crowdsourcing can be seen as a process to complete a task with the contribution of a large set of people, and it is often used to gather resources for collaborative systems such as Wikipedia[1] and for work market places such as Amazon's Mechanical Turk[2] and raise money for projects (crowdfunding). One of the ideas of Conversing Learning, is to take advantage from tasks performed by the crowd whose results outperform a single person in quality, quantity and responsiveness. Encouraged by large datasets that need human attention, crowdsourcing has been used as input of computational systems. Applications include boosting projects such as image recognition [8] and word processing [9]. Such solutions can be called crowd-powered systems [10], and include algorithms that gathers and considers the intelligence of humans.

In this work, we considered the use of crowd-powered systems to assist a class of algorithms known as bootstrap learning. Usually, in these algorithms, multiple learning methods are combined and each method

[1] wikipedia.org
[2] mturk.com

learns the prerequisites for subsequent stages. Bootstrap learning has been applied to tasks in robotics such as learning properties of objects in natural environments, where object shapes learned in initial stages are used to recognize more complex objects in the future [11] as well as place recognition [12]. Bootstrap learning is also widely applied to extract named entity categories from unstructured text, using a small set of labeled entity instances (also known as *seed instances*) to discover patterns and use those patterns to find more instances in text [13]. This idea made possible applications such as learning subjective nouns [14] and dictionaries [15]. In this field, one of the largest and most successful implementation of a bootstrap learning system is Never-Ending Language Learner (NELL) [16], a system that couples multiple algorithms that perform the task of populating a structured knowledge base (KB) of structured facts and use its own KB to perform this task better every day, in a lifelong learning fashion. One of the main components of NELL is the Coupled Pattern Learner, which is itself a bootstrap learning algorithm responsible for learning contextual patterns that recognizes part-of-speech tokens into entity instance, having a major role in populating NELL's ontology [17].

Bootstrap learning algorithms can suffer from "semantic drift" a condition where wrongfully labeled data in earlier stages of learning are used to discover new information in further stages, thus propagating the error and causing the KB to "drift" away from a feasible representation of the problem being learned. This issue is usually addressed by constraining earlier stages of learning. The constraint is usually based in the comparison of the labeled data and a separate source of reliable information which could be for instance entity checking with a known KB or coupling multiple algorithms learning the same task and supervising each other. In this work we consider the problem of semantic drift in bootstrap learning algorithms for the task of extracting named entity categories and while taking advantage of the ease that humans have to understand language, we propose the novel idea of using a crowd-powered approach to reduce the effects of semantic drift in bootstrap learning algorithms.

To put this idea to test, we developed a Multi-Agent System that includes a conversing learning agent with access to humans - which are the best machines in the world to understand language - with CPL, one of the most important bootstrap learning algorithms that learns from text in natural language. The implementation of this multi-agent is a hybrid version of CPL with the SSCrowd component, an agent that allows a learning system to post questions and read answers from users on social networks. Through SSCrowd, we ask humans to tell CPL how good its learned patterns are, allowing CPL to learn better. This approach keeps the idea of building machines that resolve their questions like humans do, that is, talking to other humans, as attempted in [18]. The SSCrowd component has been put to test on Yahoo! Answers and Twitter [19, 20].

This work proposes combining the power of the crowd and ML tasks by talking to Web users. We discuss multiple topics on Artificial Intelligence and Computer Science. To build a system that talks to humans and improve an intelligent system, raise questions that are common to Machine Learning such as the capability of a ML system to find its own issues that could be easily resolved by the crowd, and Active Learning to select the bits of the issue that are going to be assessed by humans.

In Chapter 2, we define the Conversing Learning (CL) model, detail the concerns towards building a CL task and present cases of implementation of the model. In Chapter 3, we show a background of crowd-powered systems and discuss the work done regarding human generated content, such as encouraging human participation, assure quality and manage malicious people online. In Chapter 4, we discuss a background in bootstrap learning and semantic drift and describe our Multi-Agent System (MAS) approach to diminish the effects of semantic drift in CPL. Finally in Chapter 5, we summarize this work and discuss future work.

2 Conversing Learning

To help us illustrate the main characteristics and properties of Conversing Learning, consider the task of building a software agent to detect spam messages received by an email account. Such agent is called *spam tracker*, in this work. Based on Machine Learning techniques, a spam tracker may have learned its own model (set of initial rules or facts to find spam messages). In addition, it is possible to go beyond, allowing the spam tracker to explore Active Learning to select candidate messages to be labeled as spam by the users, and to be used as new labeled training instances, as performed in [21]. Still based on Active Learning, or not, the spam tracker can keep interactively asking the e-mail account owner for new labels, thus continuously updating its own e-mail classification model (which can be seen as an Interactive Learning approach). In the spam tracker scenario, when adopting the Interactive Learning approach, the e-mail owner is usually the only human that can interact with the machine, and the machine has no need (and usually, no capability) to look for help from nowhere/nobody else. Therefore, the machine prompts the user with questions and *passively* waits for collaboration.

An Interactive Learning (IL) spam tracker counts on the user (e-mail account owner), and only on the user, to complete its IL task, often with no interaction with other humans. As well as in other ML applications, a spam tracker usually learns from a set of (labeled) annotated instances (e-mail messages). Since improvements in the accuracy of the tracker depends on the user, the tracker is not *proactive* in looking for help.

With Conversing Learning, we want to add to the system the ability to *actively* look for help in other sources whenever needed. Labeling e-mails (as spam or not spam) may not represent a great effort for a regular user, but other ML tasks, such as the ones present in a Never Ending Learning architecture [22], such as NELL [16] might have a larger set of data to be verified or labeled, and higher number of different labeling tasks to be performed (not only classifying an e-mail message as *spam*, or *not spam*). Also, in many different labeling tasks needed in NELL, the opinion of a single user may not be enough. In such scenario, a CL system can, for instance, share the validation task among several human users, and count on their different opinions as an advantage to explore redundancy. The core difference between CL tasks to other learning tasks resides in *proactively* and automatically seeking for information from human users instead of passively waiting for their collaboration. Figure 1 shows the scheme of CL.

The IL task might lose precision and confidence due to noisy feedback while it looks for help from many different human supervisors. To rely on human generated content in a Conversing Learning fashion, there are important aspects that must be taken into account, that can be informally summarized as:

- Decide **which task** could be enhanced by conversing with humans
- Based on an Active Learning-like approach, determine which **subset of knowledge** will be put to human attention
- Propose a method to **convert machine knowledge** into human understandable content
- Determine **who are the oracles** the ML system is going to consult
- Propose a method of **conversation with oracles**
- Propose a method to **interpret the collaboration** from the oracles

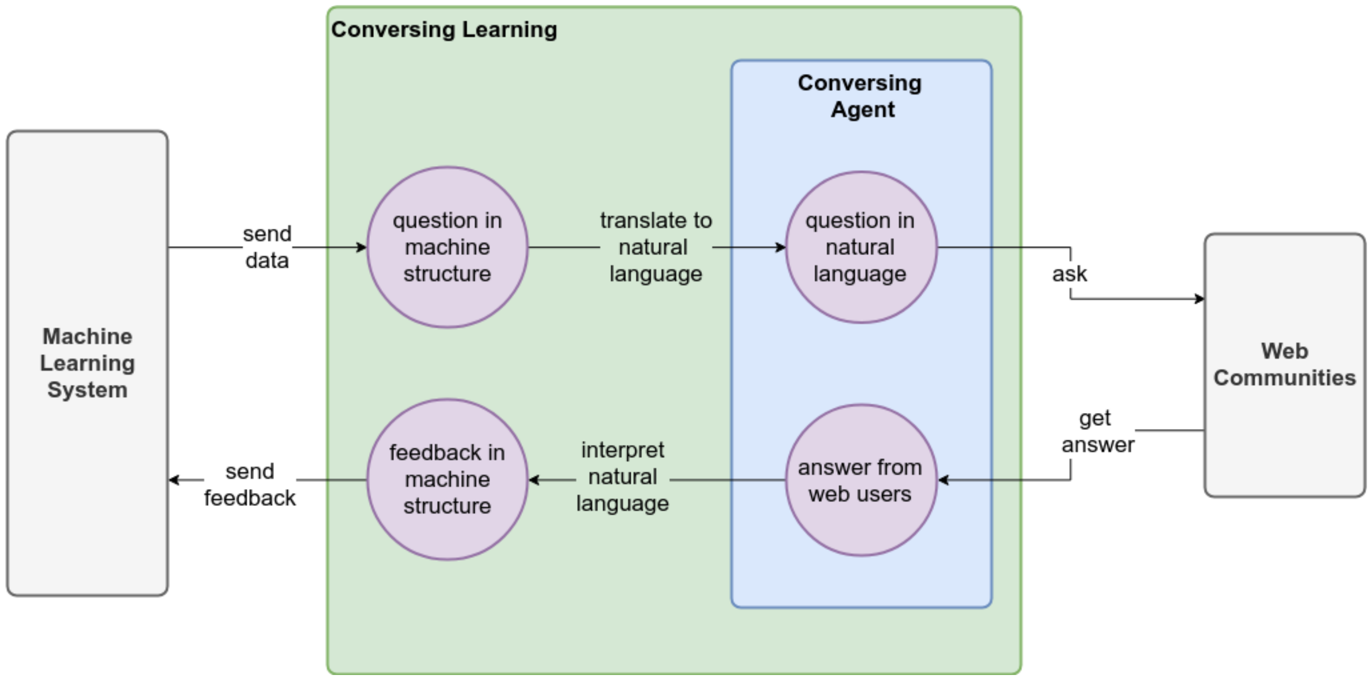

Conversing Learning
Conversing Agent
Machine Learning System
send data
question in machine structure
translate to natural language
question in natural language
ask
Web Communities
get answer
answer from web users
interpret natural language
feedback in machine structure
send feedback

Figure 1 – Scheme of Conversing Learning components

- Determine **how to feedback** the ML system with the oracle inputs

2.1 Conversing Learning Model

The core aforementioned Conversing Learning aspects can be more formally described as a 7-uple represented as by $\mathbf{CL} = (R, M, f, \mathbf{A}, \mathbf{I}, e, S)$ with R being the set of requests made by a Machine Learning system and M being the set of messages to be sent for human appreciation. f is a function to convert requests into human understandable messages and can be described as $f(r) \rightarrow m \quad | \quad r \in R, m \in M$.

$\mathbf{A}$ represents the Conversing Agent, which is a module of Conversing Learning that interfaces the communication between machine and the human oracles in an Active Learning task. $\mathbf{A}$ can be described as a 4-uple represented by $\mathbf{A} = (M, C, \Delta, p)$ with C being the set of communities where a message $m \in M$ is going to be posted and seen by the oracles. $\mathbf{A}$ also compreehends Δ, being the set of list of responses sent by the oracles as a reply for a single message m. The function p represents the action of posting a message. p can be described as $p(m, c) \rightarrow \delta \quad | \quad m \in M, c \in C, \delta \in \Delta$.

Conversing Learning also has the component $\mathbf{I}$ to interpret the message in natural language and provide structured information that can be absorbed by the machine. $\mathbf{I}$ can be described as the 3-uple $\mathbf{I} = (\Delta, \Delta_i, t)$ that has Δ and a set of interpreted responses Δ_i as well as the function t that performs the interpretation itself. t can be described as $t(\delta) \rightarrow \delta_i \quad | \quad \delta \in \Delta, \delta_i \in \Delta_i$.

Finally, Conversing Learning has the component e, a function to combine a list of responses into a single response $s \in S$ representing all of them. e can be represented as $e(\delta_i) \rightarrow s \quad | \quad \delta_i \in \Delta_i, s \in S$. With S being a set of responses for the requests in R, for each request $r \in R$ there is a response $s \in S$. This combined response is ready to be taken by the learning system and further used to improve the learning system's model. The model can be applied using the following description.

2.2 Determining a Conversing Learning Task

Conversing Learning aims at assisting ML tasks (or agents). To place a CL task, it is needed a ML system that has a task that is easier to be resolved by a group of people than by a machine.

The oracles could be providing a range of different services, such as label validation and revision, as is going to be shown in Section 2.8.1, feature acquisition, as it is going to be shown in Section 2.8.2, and also model quality improvement, that can be done, for instance, by assessing the quality of features used by a specific machine learning model.

In this work, Conversing Learning has been applied to NELL and CPL because of these system's capability to use their acquired knowledge to keep learning forever in a bootstrap learning fashion. Although the label validation on NELL can be performed by a few experts, it is expected that a ML system with the capability of continuously growing its own KB would have a tendency to require more interaction with external oracles.

2.3 Choosing Data for a Conversing Learning Task

When the KB of the learning system is large, it might be unpractical to put every bit of knowledge to human appreciation, thus it is necessary to prioritize the knowledge that is going to be sent to the web community. Also, querying the users for more information than it is usually done in a given community, may constrain the users to keep collaborating or make them tired of participating. In AL tasks, oracles *actively* select from the dataset, the information that can leverage the learning process. In CL a similar suggestion is done when the CL agent prompts the external oracles (Web users) with only the portions of knowledge that bring more benefits to the learning system.

2.4 Driven Feedback

To decide the method of communication with users, it is critical to retrieve information with good accuracy and large recall (relevant collaboration). The noisy human generated content might be complex for machine reading. In such cases, the Information Retrieval (IR) algorithm may lose part of the users responses due to difficulties in extracting relevant information from their collaboration. An alternative to cope with this issue is to encourage the oracles to provide more "machine friendly" content, that is, messages that are easier for the machine to read. Approaches like this can restrict the format of the answers provided by the users. We call such approach "driven feedback". Driven feedback can be enforced by asking the oracles to answer yes or no questions, to answer to polls or simply asking them to provide the shortest answer possible.

2.5 Choosing a Web Community

Since the members of the communities are going to represent the oracles of the CL task, the determination of the right community is also key to the success of a CL task. Much of this decision will impact in trust and responsiveness of a Web user [20]. In this work, two concepts are highlighted to support this decision:

Different communities have different purposes and gather users with different objectives. In this sense, users might be looking for different benefits in the same community. Culture, knowledge, age and language are examples of the factors modifying the response taken from them. For instance, a travel recommending system that reads from a KB, could be better fed by users from travel communities than from a Question Answering (QA) system with unrestricted domain. The work in [23, 24] suggests it is possible to work with non-experts without losing quality, and working with communities of a specific domain has the disadvantage of interacting with a smaller (and possibly biased) set of users thus, risking to retrieve a smaller set of collaboration.

The communication between humans change as the interlocutor changes. If the members of a community know that the interlocutor is a machine, they can feel uncomfortable and shy or, on the contrary, stimulated and challenged. If a community is used to assist scientific research, such as Amazon's Mechanical Turk, then the answer is more likely to be machine friendly, which favors readability. To introduce a system as a machine or not, depends on the objective of the learning system. Either way, when working with Web communities it is important to guarantee that the use of its resources does not bypass its security and privacy policies.

2.6 Converse with Humans and Interpret Answers

When the community is determined and the translated knowledge is ready to be put to users appreciation, a method to post the questions and wait for answers is required. The method will often rely on a conversing agent that access the Application Programming Interface (API) of the communities for interaction.

To increase the quality of the collaborations, the agent should apply the *driven feedback* concept in the questions and if possible, a way to find the best collaborators in the community. In crowdsourcing this is usually done by creating a model of the behavior of the crowd based on gold standard labels provided by an expert, creating a rank of the best people in the crowd. We discuss more about works regarding this matter in Section 3.2.

One interesting feature of the conversing agent is that it could be an intelligent system itself. If the CL system is going to work with multiple communities the agent could have an algorithm to decide the community that is going to receive the question or even weight the influence of different communities in the CL system.

Because the opinion of the crowd might have high variance, the conversing agent should find a consensus, an agreement between the multiple answers for a single question, and use any rank available to filter the best collaboration. If the application of the *driven feedback* concept is not possible, then the algorithm to feedback the learning system might require a new translation from the human understandable format back to machine data. Sometimes, reading human generated content is difficult and some collaboration might have to be discarded to avoid noise being propagated through the learning process.

2.7 SSCrowd

SSCrowd (Self-Supervisor Agent Based on the Wisdom of Crowds) [18], is a component designed to take advantage of the wisdom of crowds and put to test the idea of using human collaboration to assist in Machine Learning tasks. Its steps could be defined as follows:

1. receive information from the learning system that needs to be assessed by humans
2. translate the information into a human understandable question
3. post the question to a Web community and wait for answers
4. find consensus in the answers and return it to the learning system

This component was designed to perform a proactive and autonomous task that allows self-supervision on demand to learning systems, and it was first tested to validate the knowledge acquired by NELL(Never Ending

Language Learner) a system that aims to learn forever by taking advantage of the self-supervision provided by coupling different models with a shared objective [16]. SSCrowd can also be seen as an implementation of a Conversing Learning(CL) agent [20]. CL focuses on enabling an intelligent system to proactively start and keep conversations with Web users, actively and autonomously looking for human supervision, by asking people (Web users) to take part into the knowledge acquisition process, thus using the intelligence from the crowd to improve the agent's learning capabilities. [18, 19]

2.8 Case of Study

One of the most important issue to be resolved when designing a CL task, is the determination of the problem to be solved. Conversing Learning would have a greater impact in ML systems that have problems that are more easily solved by humans than machines. Usually ML systems has a human validation phase to ensure the learning model is not semantic drifting. Thus, CL can assist with automatized supervision, allowing the ML system to be more independent. Although this kind of self-supervision could be a promising application of CL, as shown in Section 2.8.1 and Section 4.3 a ML system could also benefit from interacting with humans to enhance its knowledge model.

Usually a Machine Learning system have a model of learning that describes a real world problem. This description is represented by a set of data observed by humans or sensors. Each feature that can be observed to represent the data is considered an attribute. If the content of observed data can be better acquired by the crowd, than sensors or a small group of people, we can use CL to not only validate information but to extend, create and revise the attributes which would impact how the learning model works, thus allowing the construction of a KB that has more of the common sense of the community accessed by CL as suggested by the preliminary results presented in Section 2.8.2.

The following experiments are implementations of CL tasks to improve the capabilities of NELL, which is one of the most successful bootstrap learning agents. NELL couples several algorithms with the same task of extracting information from text and populate a KB of structured facts. In a MAS approach, the algorithms learn a shared KB and use the knowledge of each component to self-supervise each other in a cooperative multi-agent setting. In addition, in a bootstrap learning task, NELL use its KB to learn more structured facts, forever.

NELL organizes its structured facts in set of categories. Each category is a generalization of the facts (instances). For an example, `food` and `bedroomItem` are categories while `hamburger` and `cheese` are instances of `food`, while `bed` and `closet` are instances of `bedroomItem`. In summary, categories are organized as a tree of classes defined by an ontology.

The first case is the implementation of a CL validation task for NELL learned facts. The second experiment shows a method to use CL to acquire seeds to start the bootstrap learning process of NELL.

2.8.1 Case A: Validation Task for NELL

Conversing Learning gather the collective intelligence [25] from the crowd and uses it to *proactively* assist ML tasks. As an example of taking advantage of this intelligence we applied the SSCrowd algorithm to ask humans about the validity of concepts extracted from NELL's KB in an attempt to test the idea of providing supervision to learning systems through autonomous assistance. This section describe the SSCrowd algorithm in the context of CL for this validation task. Detailed discussion and results of this work can be found in [18].

To show the potential of Q&A forums on helping to improve NELL, SSCrowd was experimented focusing on one of its main components, namely the Rule Learner. This component induces probabilistic first order rules based on NELL's KB. These rules are logic driven and simply tell the system how different categories should interact with each other (semantic relations discovery). When the induced rules are applied to the system, NELL's beliefs on the categories covered by these rules can be modified. Although Rule Learner (RL) is no longer being used by NELL, its rules can still be used to show how CL can assist in validation tasks. Here is an example of a rule created by RL (in a Prolog-like syntax):

```
athletePlaysInLeague(x,NFL) ← athletePlaysForTeam(x,Giants)
```

Applying this rule over NELL's database will raise its belief that every athlete that plays for the *Giants*, plays in the *NFL*. The rule is too general because does not specify whether the team is *San Francisco Giants* or *New York Giants* is being referred.[1]

Considering the imperfectly extracted knowledge present in NELL's KB, some of the fact (as well as rules) induced by NELL's learning algorithms might not represent the universe as humans know (their common sense) or might even be wrong. At this point it was seen the opportunity to work with the collected intelligence from Yahoo! Answers users to help validating NELL's beliefs in an automated way, giving NELL an extra source of self-supervision.

As seen in Section 2.1, a Conversing Learning task is formally described as the 7-uple $CL = (R, M, f, A, I, e, S)$. For this case of study, the ML system wants its rules to be verified, thus the CL task starts by receiving a set of requests R, that are in this case, rules to be verified. The feedback S is a boolean value for each rule that classifies them as *valid* or *invalid*.

A conversing agent is described as the 4-uple $A = (M, C, \Delta, p)$ which is implemented by a version of SSCrowd with the following capabilities:

- Implement the function f to convert the rules r from NELL's knowledge base into messages $m \in M$, represented here as human understandable questions.
- Implement the function p to post the question on $c =$ *Yahoo! Answers.*
- Gather the set of opinion Δ from users through Yahoo! Answers API and combine them into a single opinion s.

2.8.1.1 Implementing the function f: Convert rules into understandable questions

This is an example of a rule created by RL:

```
R1:statelocatedincountry(x, y)← statehascapital(x, z), citylocatedincoutry(z, y)
```

A question was created to represent each of the induced rules from RL. One of the main focus here was to set an algorithm which would generate straightforward questions so the user would not be confused about what to answer. Since the rules are logic driven formatted (in a PROLOG style), the conversion is simple as in the following example. The *R1* rule mentioned above, would be converted into a question as follows:

[1] Actually there are several mentions of teams worldwide called Giants.

```
Is this statement always true?  If state X has capital Z and city Z is located in country Y
then state X is located in country Y.
```

Notice that the question above is not as natural as human speech usually is. Here, we apply the idea of *driven feedback*, mentioned in Section 2.4. The generated questions reproduce the logic form of the rule and encourage the user to provide a straightforward answer, thus, complex answers that are dificult to read can be avoided. Predicate names were translated using a "human readable" version of those predicates. These translated predicates are available metadata in the Read The Web Project [2] as shown in the following example:

```
statelocatedincountry(arg1,arg2) → "arg1 is a state located in country arg2"
```

2.8.1.2 Implementing the function p: Post questions and monitor for answers

The deploy of automatic posting depends on ways of interfacing with the community system, mostly APIs, and Yahoo! Answers did not have any available at the time of the experiment, therefore it was not possible to perform an automatic posting and the action representing the function p was executed manually. After posting the generated questions to the QA forum, SSCrowd started to monitor it, looking for the answers to be retrieved. The answers were evaluated by matching them with regular expression patterns defining keywords.

2.8.1.3 Implementing the function e: Gather the set of opinion Δ from users and provide final result.

The Algorithm 1 [18] implements the function e. The combination of the users response is leveraged by the *best answer*, which is a rank assigned by the community for the answer they believe to be better. The Algorithm outputs the message *approved* or *rejected* representing the final response $s \in S$ of the Conversing Learning model.

[2] http://rtw.ml.cmu.edu/rtw/kbbrowser/predmeta:statelocatedincountry

Algorithm 1: retrieves the users opinion about a question

```
input  : consensus opinion extracted by the function $e$
output: boolean value approving or rejecting the rule
/* app is the counter of approved rules                    */
/* ref is the counter of rejected rules                    */
for $s \in S$ do
    if $s = approved$ then
        $app++$
    else if $s = rejected$ then
        $rej++$
    end
end
$score \leftarrow app - rej$
if $bestAnswer = approved$ then
    $score \leftarrow score + (app + rej)/rej$
else if $bestAnswer = rejected$ then
    $score \leftarrow score - (rej + app)/app$
end
if $score > 0$ then
    return $approved$
else if $score < 0$ then
    return $rejected$
else /* best answer is a tie breaker                       */
    return $bestAnswer$
end
```

2.8.2 Case B: Collecting Seed Instances for NELL

As most bootstrap learning based systems, NELL depends on small set of labeled instances to start learning. Those initial instances, also called seeds are, in this case, a set of examples for each category. Like NELL, CPL is a bootstrap learning algorithm and they depend on the same set of seeds to start learning. We will cover more about bootstrap learning systems in Section 4.3.

In NELL, seed instances are usually defined by human experts and are set once, when the category is created. This task requires time and expertise. As mentioned in [26], the seeds used to feed a bootstrap learning algorithm have a major role in the propagation of undesired noise. In addition, a learning system NELL could potentially to create its own categories, which would also require the definition of seed instances. Such scenario motivates this experiment, which introduce a CL task to collect seed instances from Twitter users. We intend to measure the effectiveness of such method by comparing the volume of knowledge generated (part-of-speech tokes recognized as category instances) with the seeds from Twitter users and the seeds proposed by NELL's experts.

The SSCrowd algorithm defined in Section 2.7 was used as a conversing agent for this experiment. The tweets were posted periodically and automatically using the Twitter API. The time of posting is controlled by a counter that knows the frequency of people answering per hour and tries to post the questions when users usually answer. Actually, this is a very simple approach to try to increase the volume of answers on posted questions.

Following the directions of the CL model, the conversing agent would have a set of messages m being the questions to be posted on the community $c = Twitter$, and the function p executed through Twitter's API, which allows posting questions to NELL's followers.

Seed extraction from Twitter replies (representing the function t), and the list of responses Δ, are represented by the set of answers for the question. The questions were created using a simple and fixed question template. The example below with the category clothing was extracted from the experiments.

`Question: Could you please give me some examples of clothing?`

`Answer 1: Snowshoes, rain ponchos, galoshes, sunhats, visors, scarves, mittens, and wellies are all examples of weather specific clothing!`

`Answer 2: pants`

`Answer 3: Training shoes can be worn by anyone for any purpose, but the term means to train in sports`

To help with the extraction implementing the function e, the Stanford Log-linear Part-Of-Speech Tagger developed by The Stanford NLP Group was used [27]. `Answer 1` would be tagged as follows:

```
Snowshoes_NNS, rain_NN ponchos_NNS, galoshes_NNS, sunhats_NNS, visors_NNS, scarves_NNS, mittens_NNS,
and_CC wellies_NNS are_VBP all_DT examples_NNS of_IN weather_NN specific_JJ clothing_NN !_.
```

The commas are seen as phrase delimiters and the extraction algorithm is instructed to get tokens where all the words in the phrase are names (indicated by the tags NNS and NN). Thus, `wellies` is not considered a seed because it is mixed with other elements that are difficult to read. For the same reason, the whole `Answer 3` was discarded by SSCrowd. As explained in Section 2.6, sometimes reading is too difficult and a few collaborations are discarded to prevent propagating noisy errors. In this experiment, complex answers such as `Answer 3` are rare. Since the Twitter users know their answers are taking part in a scientific experiment, thus they are more likely to provide readable answers. In this experiment, the feedback s, that is, seeds after the extraction, are used to create a new training set (with the new seeds).

2.8.2.1 Results

In the performed experiments, instead of using current NELL's KB (the one available at http://rtw.ml.cmu.edu), a fresh KB was created from an initial training set having 270 categories and values for attributes defined by NELL's experts. NELL ran 2 iterations from that training set and promoted a total of 11K new instances for the categories.

In a second moment CL was used to ask for seeds for each one of the 270 categories for the training set. The community responded with 552 seeds for 129 categories. This means not all categories received new seeds because not all questions were answered. Besides the CL issues with gathering collaboration, part of this issue is due to troubles of function t in resolving answers. The 552 new seeds helped NELL to find 5300 concepts. Seeds defined by NELL's experts would promote 6K concepts.

The preliminary results shows that although there are subtleties of NELL and CL open for more deep research, using CL acquire seeds is a promising task. After 2 iterations, seeds from CL helped NELL to promote concepts with only 10% gap from the seeds proposed by NELL's experts. An extension of this particular experiment could open the possibility of using CL as a surrogate of manual seed acquisition.

2.9 Discussion

Conversing Learning uses the wisdom of the crowds to leverage ML systems to resolve problems where the intelligence from a group of human oracles outperforms the machine for a given task. CL raises challenges such as building an intelligent system that can talk to people on the Web, assure collaboration and quality on human generated content and design communication channels that connects human and machine in a Multi-Agent System fashion. In this section, we discuss limitations, insights, plans and possibilities to address these challenges and also show some of the subtleties of working with CL so far.

The SSCrowd presented in Section 2.7 is an implementation of a conversing agent. The SSCrowd algorithm allows the exchange of messages between the learning system and Web users in a student-teacher MAS design. Although the agent can successfully ask questions and retrieve answers, it does not learn how to perform such task. That is, the conversation is predefined and will not change after experience.

However, since the agent is exposed to data such as Web users behavior and a lot of textual contributions, there is a chance to make SSCrowd retrieve better answers by learning how to interpret answers, and how to trust in the collaborations. In addition, SSCrowd can work with multiple platforms of communication for a single CL task, as suggested in [20], which would make possible to learn where to post questions and get more relevant answers (increase recall). Such ideas would make the conversing agent a ML system itself and increase the capabilities of CL to be a reliable source of information to intelligent systems.

2.9.1 Conversing Learning Limitations

Usually, crowd-powered applications have their own way to connect with humans. However accessing the crowd through systems that were not created for this purpose may limit the flow of the conversation. The example described in Section 2.8.1 for an example, uses keywords as an approach to understand answers, which means that answers not matching such keywords would be dismissed. Furthermore, at the time of the implementation, Yahoo! Answers did not have a way to post questions through the API, which means the questions have been posted manually, that is, the capability of function p to be automatic was compromised.

Those challenges raise the discussion of considering the method of conversation as part CL problem itself. Therefore, some learning systems will not be able to implement solutions for all the CL core issues due to problems in conversions for the function f, lack of Natural Language Processing (NLP) techniques for a satisfactory function t, and even the lack of reliable environments (Web communities) to act as the conversing agent.

One approach to develop a conversing agent, are the chatterbots [28], agents designed to chat with humans with an interface that allows conversation through typing. Usually the main goal of such agents is to pass the Turing test [29], but in CL we can implement a conversing agent by creating a chatterbot that asks the questions defined for the CL task. The increasing market of social networks and messaging API's such as the Telegram Bot API[3] could make the interaction between humans and machine to be more dynamic and create a sense of context that could drive the users to give more precise collaboration.

2.9.2 Subtleties of Different Web Communities

Several intelligent systems using crowdsourcing built their own MAS which usually rely on its own set of humans [30, 31]. Because of the controlled set of users and the straightforward task description, these systems are a possible solution to increase the collaboration and ensure quality in human generated content. However,

[3] https://core.telegram.org/bots/api

not learning from task-designed talking systems, gives the machine the ability to learn from humans as we do, that is, asking questions on open domain QA websites or looking for help from a broader audience. Popular social applications usually have a greater set of users, thus the answers for the ML system would have a higher tendency to better represent the common sense. The following example comparing SSCrowd with Yahoo! Answers and Twitter, can be used to show of how common sense can be extracted from CL tasks [20].

To have CL validating a rule from NELL's rule learner, users from Twitter and Yahoo! Answers were asked the same following question:

`(Yes or No?) If athlete Z is member of team X and athlete Z plays in league Y, then team X plays in league Y.`

Answer sampled from a Twitter user:
`No. (Z in X) not (Z in Y)` *to* `(X in Y)`

Answer sampled from a Yahoo! Answers users:
`NO, Not in EVERY case. Athlete Z could be a member of football team X and he could also play in his pub's Friday nights dart team. The Dart team could play in league Y (and Z therefore by definition plays in league Y). This does not mean that the football team plays in the darts league!`

The example above shows the difference of an answer given from a Twitter user that knows the purpose the CL task and a Yahoo! Answers user that does not know. Notice that the first is something that a shallow machine reader really needs, a straightforward answer with no space for misunderstanding which is easy to automatic interpret, while the second is something humans want, a full explanation of an opinion. In the experiment in Section 2.8.1, the SSCrowd algorithm dismissed both answers because the function t (keyword based) could not extract an accurate response. However these messages gives a glance on the value contained in open domain conversational systems.

With SSCrowd and Yahoo! Answers, although the usage of the keyword approach, most of the answers could be understood by suggesting the users to respond to a Yes/No question. And it was possible to make use of the *best answer*, provided by Yahoo! system, that is the opinion from the users about which answer represents better the interests of the community.

3 Crowdsourcing and Crowd-Powered Systems

The work exploring the idea of bringing human attention to Machine Learning tasks increased since the Internet and social media shortened the distance between people and machine intelligence. The power of the crowd has been applied in ML to support tasks like sentiment analysis [32], product classification [33] and labeling images [34]. Crowdsourcing has been widely used to annotate labels for datasets in ML tasks. However, more than provide labels, it has been used as part of the intelligent system itself, in a Multi-Agent System configuration like the work in [35], where sentences translated by people are combined to a ML system allowing the provision of a translation that is more cost effective than professional translations.

Gathering the crowd to assist ML tasks raise challenges, such as discovering how to keep the crowd motivated, figuring how to manage communication channels like Amazon's Mechanical Turk (AMT) and how to ensure quality by avoiding noisy collaborations and malicious people [36, 37, 38]. These issues are usually addressed by identifying the best annotations and annotators [39, 35, 40]. One possible way to reach such annotation, as well as annotators identification, is to study the trade-off between consensus and coverage, which has been done in [32]. In this work we approach the problem of annotation/annotator quality estimation by introducing the concept of *driven feedback*, mentioned in Section 2.4, which restricts people to provide specific collaboration, thus avoiding the propagation of error in ML systems due noisy to feedback, specially in bootstrap learning algorithms.

In Machine Learning, most of the communication management has been done through Amazon's Mechanical Turk, however, to achieve more accurate annotations or to control the humans participation, some intelligent systems have their own platform to connect with humans [30, 31]. On the contrary, in this work, we describe a way to connect humans and ML through conversation on social networks. Conversing Learning should allow machine to ask for help in Web communities and collect and find consensus in the crowd's answers.

An interesting work on knowledge acquisition through the crowd is the Curious Cat [38], an agent that talks to people through an app to collect knowledge. Their work also brings approaches to resolve malicious content from humans by checking redundancy with other people, checking consistency with a large KB like CyC [41]. Moreover, the agent learns context from the users creating a channel of communication that targets a single user allowing for an example, to ask specific questions for users that are more likely to be able to answer. Ideas like that open the field of using other known KB as Google's Knowledge Vault [42] and Freebase [43] as part of the consistency checking to prevent the issues caused by human generated content.

3.1 Quality Control in Crowdsourcing

To rely on human generated content may be difficult if there is no way to assess and measure quality of their collaboration. One way to diminish this issue is to restrict the collaboration only to qualified people, which could be done in different levels. From hiring people with restricted agreements where quality is a priority [33], to systems like AMT where the workers receive some instruction. Usually, the more control you have over the workers, the more expensive is the job they perform. In addition, crowdsourcing tasks closed to a restricted audience would have to consider the problem of collecting less information than it would from a broader audience.

Another approach to improve quality would be to have experts to supervise the content sent by the audience. Although the experts have excellent knowledge of the domain of the task, the demand of data to process could be larger than they could manage. Moreover, the supervision itself could biased by the expert personality.

We also investigated a third approach where the workers can vote on the best collaboration [18], which has also been done in [44], where the study pointed the problems of information cascading, a bias that leads people to vote on collaborations that are "winning", which could diminish their authenticity. In addition, systems that use voting to infer quality will usually take the majority of votes as an important parameter for a final consensus. The work in [37] brings the discussion of the "minority voice" a situation where the majority of people is wrong and a single voice have a valid interpretation of a problem or question.

The number of votes that crowdsourcing systems rely on are limited to the will of participation of the people. For requests that have not been satisfactorily answered , there is the possibility to try the same crowdsourcing task for better results. Since the request for help is limited, we need to either look for more collaboration for a single task and calculate consensus, or look for a broader coverage of more tasks with possibly less collaboration. A work [32] studied the trade-off between consensus and coverage and brought the following questions:

Consensus: How does the agreement between the annotators, or lack thereof, affect the performance of a given task?

Coverage: With limited resources, how can we make effective use of the budget to adequately cover the domain?

The process of answering such questions assists the design of the CL task. The definitions of features like the window to wait for collaborations before closing the task and the minimum amount of collaboration for each question depends on how we deal with consensus and coverage.

In Machine Learning tasks, the success of a model usually depends on a domain expert to assist in finding and tuning features of the learning algorithm. The ability of crowd-powered systems to learn from experts can be an important feature of a successful task. However, it will not always be possible to count with those experts, specially in crowdsourcing tasks with open domain or little resources where specialists could be expensive or unavailable. To address this issue, the GLAD system [40] created an algorithm to test and infer the expertise of each worker. With this idea one could simply hire a pool of less expensive and (potentially) less skilled workers and optimize their participation in a ML task. Similarly, a work on translation [35] used features extracted from experts collaboration to score the collaboration from non-experts. They also discovered that for their task, the non-experts are cheaper and can perform high-quality translations.

3.2 Managing Crowdturfers and Increasing Motivation

While working with crowdsourcing, specially when interacting with people in open environments such as Twitter and Yahoo! Answers, we might have to deal with people that are intentionally trying to harm the machine they are talking to, these people are also known as *crowdturfers* [45]. The crowdsourcing community acknowledge such threat and have discussed and implemented a few solutions.

One study, aimed at understanding the behavior of malicious people in online surveys and proposes a few guidelines which includes pre-screening to filter ineligible workers and minimum time threshold to identify workers that finish too early [46]. The same study propose a measurement of maliciousness based on a history of unsatisfactory responses for a single user. In a similar fashion, researchers developed a ML model to identify crowdturfers on Weibo[1] (a Chinese microblogger, a website similar to Twitter) [47]. The task is to identify fake profiles. They employ features such as number of followers, total of comments and retweets, and time-intervals of tweeting. With simple ML models such as Naive-Bayes and Support Vector Machines (SVM), they could identify crowdturfers with an accuracy up to 99%.

Other approaches to deal with malicious people include using gold standard annotations to measure how far the workers are from expected responses [35], and use support from known knowledge bases to check consistency. The work on Curious Cat [38], performs such approach and also includes checking consistency by asking for the same information for different people. In this work we do not apply techniques to ensure the collaboration are intentionally harmful because the Twitter accounts we used are openly used for research and most of audience is interested in the results so they are less likely to intentionally provide noisy information. As mentioned in Section 2.5, the way the communication presents itself to the audience, influences the audience behavior.

Crowdsourcing tasks depends on the collaboration of people, usually this demand is addressed by using money, and rewards such as gifts and points which would turn into scores in competitions for status [30, 48]. Although these approaches has given good results, a study [49] suggests that people tend to become inactive after a few submissions, it is faster if money is not involved. Moreover, in competition, people tend to select tasks where they are competing against fewer opponents to increase their chances of winning and are prone to select less popular and higher rewards tasks.

[1] weibo.com

4 Reducing the Effects of Semantic Drift in CPL

In Natural Language Processing, bootstrap learning algorithms have been mostly used to extract lexicons from text [15], but they were also used in tasks such as web page classification [50] and word sense disambiguation [51]. They can learn from a large amount of unlabeled data and a small set of labeled data, called "seeds", in a semi-supervised learning task. The seeds are used to train an initial model that labels part of the unlabeled data, and then use this model to train a new model that uses the seeds and the recently self-labeled examples.

4.1 Background

The topic of bootstrap learning algorithms for NLP have been widely studied. The work in [52] describes three phases of iteration that are common to these algorithms:

1. **Pattern Induction:** Induce patterns from a corpus given seed instances.
2. **Pattern Ranking/Selection:** Create a pattern ranker from a corpus using instances as features and select patterns which co-occur with seed instances for the next instance extraction phase.
3. **Instance Extraction:** Select high confidence instances to the seed instance set.

To the best of our knowledge, each work in bootstrap learning for NLP has its own version of the framework above. We show our approach in Section 4.4.2. The process described above is iterative, which allows the amount of labeled data to expand with minimal supervision. However, the ambiguity of language introduce errors in the iterative process which can lead to semantic drift, a situation where wrongfully labeled data propagates through the learning iterations [53].

A straightforward approach to avoid semantic drift is to terminate iterations before hitting generic patterns, but the optimal number of iterations is task dependent and hard to define. Most algorithms try one of the following approaches: A) run a fixed number of iterations, which requires some form of semantic drift measurement to guide when iterations should stop [54] or B) try to infer the best stopping criterion from a separate set of labeled data, which requires expensive labeled data [55]. The problem of semantic drift is considered a property of bootstrap learning algorithms and have been studied widely by the NLP community. We address some of them below.

Attempts to reduce semantic drift by comparing learned data with previously learned data:

The work in [52] proposed the use of graph-based algorithms that exploit generic patterns. Von Neumann kernels are usually used to measure similarity between documents, but in this work, it was applied to a pattern-instance co-occurrence matrix instead of the document-word matrix, to estimate how learned instances are related to the seed instances for the task of word sense disambiguation. Still relying on seed instances as gold standards, an interesting work in [56] reduced semantic drift in the bootstrap learning task of relation extraction. The approach extracted features from seed contextual patterns and built a cluster of patterns. Each cluster represents a relation between two entities. At each iteration, during the rank step, the distance between named entity pairs and the target cluster is calculated and if the distance is too long the iteration process stops, which prevents semantic drift. The work in [57] identified bio-medical semantic lexicons from text, applying a semantic drift score measurement based on average distributional similarities between instances extracted at iteration *i* and instances extracted at iteration *i-1*. During the rank step, instances that score below a given threshold are filtered out of the learning process, thus preventing semantic drift. To address the problem of semantic drift, the work in [26] identified that the choice of seeds affects the propagation of errors and proposed a bagging of random seeds that reduce the semantic drift in the task of extracting semantic lexicons from text.

Attempts to reduce semantic drift based on co-occurrence counts and coupled algorithms: The work in [58] introduced Espresso, a bootstrap learning algorithm that learns binary semantic relations and uses a instance and pattern reliability score, which is proportional to how often a particular pattern co-occurs with a particular instance. In NLP bootstrap learning algorithms, the most successful approach to diminish semantic drift, is to couple different functions that perform the same task, which allows the functions to supervise each other through the integration of their knowledge [59, 15]. Another important example of coupling functions to reduce semantic drift is the Coupled Pattern Learner, which is a model that learns to extract predicates from an unstructured text in a bootstrap learning fashion. CPL relies only on a few seeds of those predicates for training and uses its previously learned predicates to learn new predicates [17]. Because CPL is the major bootstrap learning component of NELL, which is one of the most successful bootstrap learning algorithm, it was used as the object of study of this work. We discuss the methodology of CPL in Section 4.2 and our approach to reduce semantic drift in CPL in Section 4.4.

4.2 Coupled Pattern Learner

The knowledge representation of CPL is a hierarchical structure of predicates divided into categories and relations. A category can be seen as a class of instances. For example, John Travolta is an actor because *"John Travolta"* is an instance of the category *actor*. In addition, John Travolta starred in the movie Pulp Fiction because the pair *(John Travolta, Pulp Fiction)* is an instance of the relation *actorstarredinmovie*. This relation must be satisfied by the constraint that the pair *(X, Y)*, X must be an instance of the category *actor* and Y must be an instance of the category *movie*.

CPL access unstructured text data sets to learn contextual patterns that are good extractors of predicates. For example, the sentences *"X and other actors"* and *"X agreed to star in Y"* are contextual patterns that could be used to respectively extract instances of the category *actor* and pairs *(actor, movie)* for the relation *actorstarredinmovie*. To learn contextual patterns, CPL performs two distinct operations:

(i) Part-of-speech tags found in the sentences that match X and Y, are said to co-occur with the contextual patterns and turn into candidate instances of the category or relation extracted by the pattern. The candidates are filtered and ranked to be promoted into actual instances.

(ii) CPL then re-access the data set looking for candidate contextual patterns that co-occurs with the promoted instances. These candidates are also filtered and ranked to be promoted into actual contextual patterns.

CPL performs the operations (i) and (ii) described above iteratively. The contextual patterns learned in operation (ii) are reused to extract more instances in operation (i), which closes a loop that allows CPL to learn in a bootstrap learning fashion. Although this model can learn predicates for categories and relations, in this work, we are only addressing categories due to space constraints. The steps to promote candidate patterns and instances are described in Section 4.4.2.

4.3 Using Human Generated Content to Avoid Semantic Drift in CPL

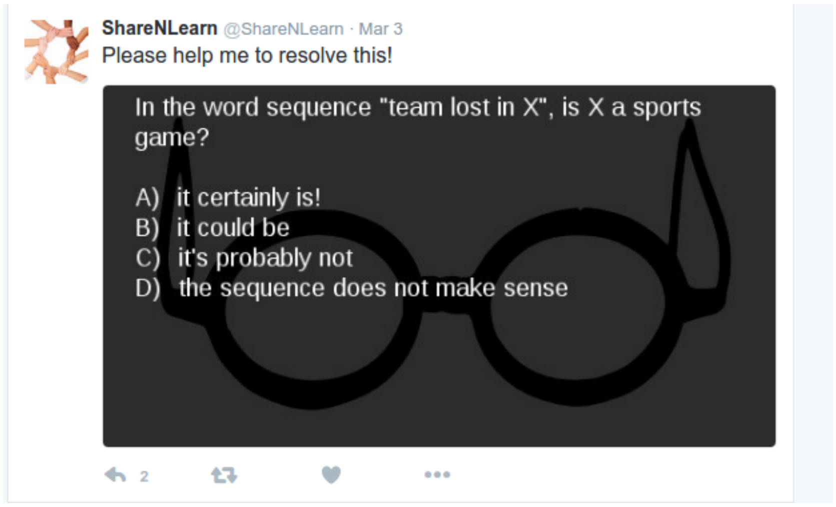
To use contextual patterns and instances promoted in previous iterations to keep learning could allow CPL to perform tasks such as Named Entity Recognition. However, since language can be ambiguous and under-constrained, it can also lead to semantic drifting. In [17], this problem is approached by learning predicates for multiple categories simultaneously and penalizing predicates that co-occurs with predicates from mutual exclusive categories. For example, John Travolta is an actor, and *"cities such as X"* is a contextual pattern that extracts cities. Since actors cannot be cities, the categories are mutually exclusive and therefore any candidate instance of *actors* that co-occurs with *"cities such as X"* are violating the coupling and will be penalized, hence, being less likely to be promoted.

The semantic drift in CPL raises when the instances learned in previous iterations for a given category c yield contextual patterns that are extractors of a wider range of categories than c. For example, consider that for the category *food*, CPL has learned the instances *carrots*, *onions* and *tomatoes*, which yielded the contextual pattern *"asian species of X"*. This contextual pattern extracts those instances, but it could also extract *bears* and *monkeys*, which are instances of the category *animals*.

In this work, we propose the novel idea of using a *human-in-the-loop* Multi-Agent System approach to reduce the effects of semantic drift in a bootstrap learning algorithm. We put our idea to test through the improvement of CPL performance by asking humans to evaluate CPL's candidate contextual patterns and reward those that are more likely to extract only instances from the pattern's category. Such evaluation task is hard for machines to do, due to the ambiguity of language, but it is an easier task for humans.

4.4 Enabling CPL to work with SSCrowd

The scenario of an evaluation task that is hard for machines and easy for humans, motivated us to use a Conversing Learning setup as an extra step to the CPL pipeline. Algorithm 2 shows the adapted version of the CPL algorithm originally proposed in [17]. In our version, at the end of the promotion step, we take all the candidate contextual patterns that were promoted and build a question to be posted on Twitter through the SSCrowd algorithm. The consolidated answers from users creates a score that we use to modify how the filter and the rank step work for candidate instances. We could also use SSCrowd to validate candidate instances, as it has been done in [18], however CPL is more sensitive to contextual patterns since there are less of them, and they can yield more instances than instances could yield patterns.


ShareNLearn @ShareNLearn · Mar 3

Please help me to resolve this!

In the word sequence "team lost in X", is X a sports game?

A) it certainly is!
B) it could be
C) it's probably not
D) the sequence does not make sense

Figure 2 – Example of a question asked in Twitter

Algorithm 2: CPL Algorithm with SSCrowd

```
input  : seed instances and seed patterns for each category $c \in C$
output: instances and patterns for each category $c \in C$
for $i$=1,2,...,$\infty$ do
    for $c \in C$ do
        EXTRACT new candidate instances/contextual patterns using recently promoted
          patterns/instances
        FILTER candidate contextual patterns that violate coupling
        FILTER candidate instances with SSCrowd validated contextual patterns
        RANK candidate contextual patterns
        RANK candidate instances with SSCrowd validated contextual patterns
        PROMOTE top candidate instances/contextual patterns
        ASK Twitter users to validate candidate contextual patterns
    end
end
```

4.4.1 Gathering the Opinion from the Crowd

We translated CPL needs of evaluation of contextual patterns by questioning people whether they believe a noun fits a given contextual pattern. Figure 2 shows an example with the category *sportsgame* and the contextual pattern *"team lost in X"* asked in Twitter. We have tried to place the question in a natural manner, as human speech is. The question template and the options for answers are always the same, and we expected people to either answer with representative letters (A, B, C, D) or the sentence or both.

In this work, we do not aim to micro-read the answers of the users, therefore we needed a way to motivate people to give a straightforward answer. To do that, we took the idea of *driven feedback*, that is, to modify the question in a way that it yields answers that are easier for machines to understand, thus reducing efforts in understanding language. This idea was mentioned in [20], and we implemented it by using polls to capture the intention from the users. By the time we performed the experiments, the Twitter API would not allow polls, so we have used images instead. The feedback for CPL is the consensus through majority vote. Ties were decided in favor of worse contextual patterns. We then use the consensual votes to modify the filter and the rank step. The consensual votes determine *sscrowd*(p) for the pattern p as follows:

$$sscrowd(p) = \begin{cases} 0 & \text{if majority vote is C or D} \\ 1 & \text{if majority vote is B} \\ 2 & \text{if majority vote is A} \end{cases}$$

With this setup, the contextual patterns that receive an "A" answer are the best extractors, followed by "B". "C and D" are the answers for bad extractors. Figure 3 shows the organization of our approach.

4.4.2 Execution Steps for Modified CPL

4.4.2.1 Extraction Step:

At its first iteration, CPL uses a few labeled seed examples to start its execution. Those seeds are automatically promoted into instances and patterns. In the following iterations, CPL will use previously promoted instances and patterns to learn new instances and new patterns. During the extraction step, it finds instances and contextual patterns that co-occur with each other to build a list of candidate instances and candidate contextual patterns.

4.4.2.2 Filter Step:

In this step, for each category, some candidate instances will be rejected if the number of times they co-occur with promoted patterns $p \in P$ is not greater than τ times the number of times they co-occur with patterns from mutually exclusive categories $p \in \bar{P}$. In our version, CPL uses the votes gathered from SSCrowd to improve filtering. To do that, we added *sscrowd*(p) to this step in a way that instances that co-occur with good extractors are less likely to be rejected and instances that co-occur with bad extractors are rejected. The τ parameter controls the softness of the filtering. The boolean value for rejection in the filter step is calculated as follows:

$$reject(i) = \sum_{p \in P} count(i,p) \times sscrowd(p) < \sum_{p \in \bar{P}} count(i,p) \times \tau \tag{4.1}$$

In Equation 4.1, *count*(i,p) is the number of times the instance i co-ocurrs with the pattern p. Candidate patterns are filtered in the same manner, using promoted instances instead of promoted contextual patterns. However, they would not use any form of assistance from SSCrowd, since, in this work, SSCrowd was not used to evaluate instances.

4.4.2.3 Rank Step:

During the rank step, category instances are ranked higher when they co-occur with most of the contextual patterns. *sscrowd*(p) was added in the rank step to rank higher candidate instances that co-occurs with the best extractors, our approach to rank candidate instances is calculated as follows:

$$rank(i) = \sum_{p \in P} pair(i,p) \times sscrowd(p) \tag{4.2}$$

where,

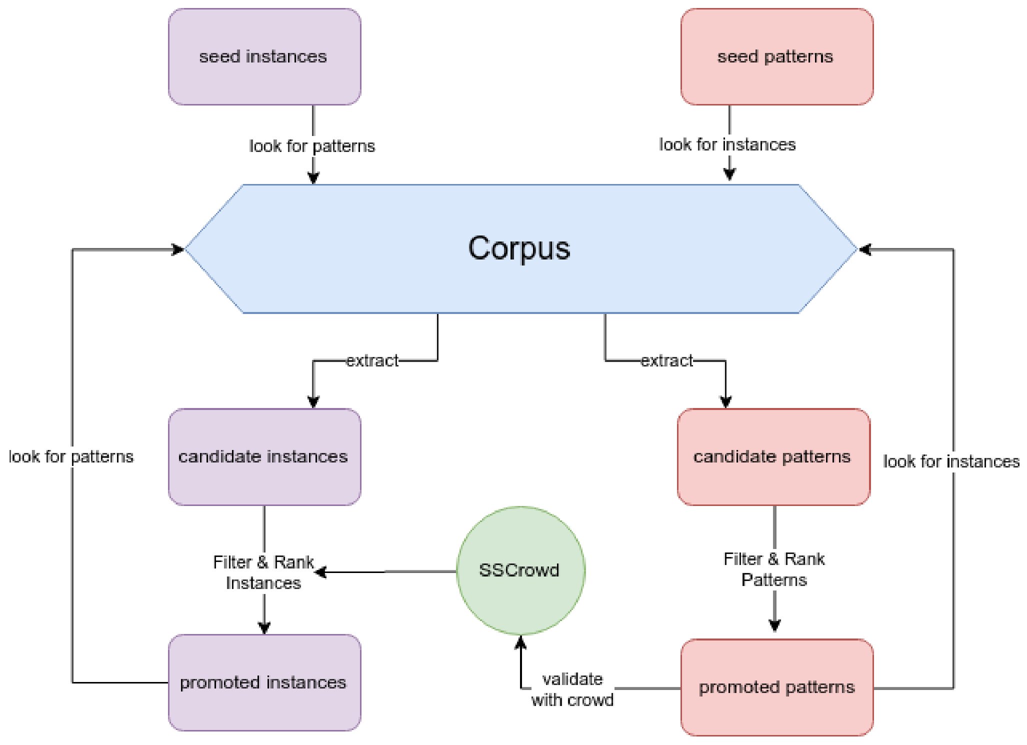


Figure 3 – CPL + SSCrowd Diagram

$$pair(i,p) = \begin{cases} 0 & \text{if } i \text{ do not co-occurs with } p \\ 1 & \text{if } i \text{ co-occurs with } p \end{cases}$$

Candidate patterns are ranked differently, based on a score that favors candidates that co-occurs with most promoted instances. For candidate patterns, we use the same rank setup as described in [17].

4.4.2.4 Promotion Step:

This step promotes the instances and contextual patterns that are ranked higher based on a threshold parameter. In our work we promoted the top 3 contextual patterns and the top 20 instances.

4.5 Experiments

To show that a crowd-powered approach can be used to reduce semantic drift in a bootstrap learning algorithm like CPL, in this section, through our experiments, we want to answer the following question: How the evaluation of contextual patterns through talking to people would affect the amount of promotions and precision of the category instances learned by CPL?

4.5.1 Experimental Setup

CPL was first implemented using a data set from 200 million web pages [17]. In our experiments we used the English Wikipedia, which, by the time this document was written, had 6 million pages. We made available

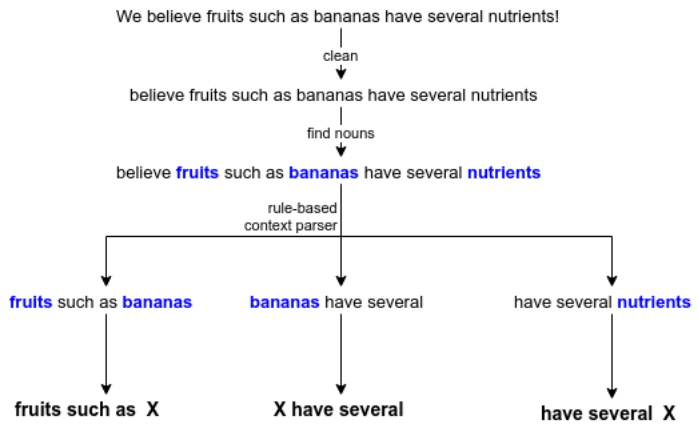

We believe fruits such as bananas have several nutrients!
clean
believe fruits such as bananas have several nutrients
find nouns
believe fruits such as bananas have several nutrients
rule-based context parser
fruits such as bananas
bananas have several
have several nutrients
fruits such as X
X have several
have several X

Figure 4 – Contextual pattern extraction

a Python version of the CPL algorithm in GitHub[1]. The Wikipedia articles were tokenized and parsed with spaCy[2] to extract noun phrases. We then created a data set of unique noun phrase and contextual pattern co-occurrence counts, and filtered out those that occurred only once to avoid noise. This process yielded a 70 million unique pairs of noun phrases and contextual patterns. Figure 4 shows the extraction of contextual patterns that pairs with the nouns *"banana"* and *nutrients* from a sentence. It is important to notice that both patterns *"fruits such as X"* and *"X have several"* could extract fruits, however, while the first is good at it, the second could lead to semantic drift. The unique pair extraction algorithm, also known as all-pairs data creator, was made available at GitHub[3].

As shown in Algorithm 2, at each iteration CPL calls for SSCrowd's assistance. CPL sends to SSCrowd a contextual pattern and a category name. Then, SSCrowd builds the question and post it on Twitter. To give time for people to answer, the questions were posted every 2 hours, which means that a queue of questions is formed, which delays the execution of CPL. The answers were gathered after 24 hours of posting. Answers received after that window were not considered. This is reasonable time, since questions receive most of the answers immediately, mostly because on Twitter, older messages are quickly replaced by new ones in the user feed. Therefore, in this experiment, the decision about how to manage the trade-off between consensus and coverage mentioned in Section 3.1 was biased by the properties of Twitter.

The experiments were performed over a set of 42 categories based on the original experiments described in CPL's first implementation [17] plus the addition of the categories *disease* and *fruit*. At each iteration CPL would learn, for each category, at most 3 contextual patterns and 20 instances. After the end of each iteration CPL waits for one day before closing the window for answers. The answers are gathered and a consensus is determined, as described in Section 4.4. For the experiments described in this work, we posted 375 questions, one for each contextual pattern promoted by CPL using two different Twitter accounts. In both accounts, people would know that they were taking part in a scientific experiment, however, no instructions were given, so we could capture most of the Web users intentions while answering the questions. The questions received a total 1372 answers, from which 1248 were useful (some answers were too complex to understand). With this setup, after 10 iterations, CPL would not learn any new instances or patterns, therefore, we have ended the execution at this point.

[1] https://github.com/saulodspedro/cpl
[2] https://spacy.io/
[3] https://github.com/saulodspedro/all-pairs

We ran two sets of experiments, the first with CPL without considering the $sscrowd(p)$ parameter. We will reference this experiment as "CPL". The second experiment is the combination of CPL and SSCrowd as described in Equations 4.1 and 4.2. We will reference this experiment as "CPL + SSCrowd". For both experiments, we used $\tau = 3$ to enforce filtering. As the work in [17], we also did not perform deep sensitivity analysis for τ.

After the execution of both experiments, we have manually evaluated all the instances from CPL. The result of this evaluation is shown in Table 1.

4.5.2 Experimental Evaluation

We noticed that CPL itself does not perform very well with a small corpus for a considerably large number of categories. More data allows CPL to grow its coupling constraints, thus learning more accurate instances. In Table 1, we show the promotions and precision of the category instances learned by "CPL" and "CPL + SSCrowd", where both were trained using Wikipedia as corpus and compare them with the results described in [17], where the same "CPL" algorithm was trained with 200 million Web pages as corpus. This data set is a preliminary version of the ClueWeb09 data set [60]. In the table, we refer to these results as "CPL CW". For each category, the precision measure is the total of category instances correctly identified divided by the sum of categories instances found by CPL. The promotion measure is the number of category instances that made through the promotion step defined in Section 4.4.2.

	Estimated Precision			Promotions		
	CPL CW	CPL	CPL SSCrowd	CPL CW	CPL	CPL SSCrowd
academic field	70	68	91	46	16	12
actor	100	58	75	199	74	88
animal	80	100	100	741	36	36
athlete	87	46	80	132	128	66
award trophy tournament	57	25	54	86	60	35
board game	80	31	47	10	57	38
body part	77	45	52	176	51	44
building	33	55	81	597	106	58
celebrity	100	55	66	347	115	84
ceo	33	58	62	3	34	37
city	97	72	83	1000	85	97
clothing	97	52	89	83	95	57
coach	93	44	56	188	87	64
company	97	39	56	1000	107	75
conference	93	30	50	95	81	65
country	57	56	72	1000	89	76
disease	n/a	66	80	n/a	69	55
economic sector	60	16	21	1000	105	123
emotion	77	65	65	483	55	55
food	90	54	100	811	112	95
fruit	n/a	26	54	n/a	114	57

furniture	100	47	79	55	55	34
geometric shape	77	26	25	43	56	58
hobby	77	31	43	357	57	46
kitchen item	73	20	61	11	77	31
mammal	83	53	87	224	64	40
movie	97	35	66	718	74	39
newspaper	90	80	80	179	20	20
politician	80	28	42	178	121	77
product	90	34	93	1000	129	45
profession	73	22	33	916	155	109
professional organization	93	23	23	104	52	52
reptile	95	93	93	19	30	30
room	64	15	27	25	132	83
scientist	97	34	56	83	129	75
sport	77	45	52	283	71	61
sports equipment	20	22	100	58	80	17
sports league	100	88	88	11	35	36
sports team	90	63	100	301	55	35
stadium	93	24	28	102	81	78
state or province	77	92	85	202	56	68
tool	40	27	41	561	115	77
university	93	47	64	1000	80	62
vehicle	67	44	50	460	74	62
mean	75	47	65	338	78	58

Table 1 – Promotions(#) and estimated precision(%) for all categories

As shown in Figure 6, CPL + SSCrowd would not promote as many instances as CPL did, however, it will assist in diminishing the effects of semantic drift. Over 10 iterations, CPL learned 3474 instances with a precision of 47% and CPL + SSCrowd learned 2552 instances with a precision of 65%, also improving the precision of 37 out of the 44 categories tested as shown in Table 1. For both algorithms, most of the instances are learned in the first iterations, where the precision is higher, boosted by the gold standard quality of the seed instances. As iterations move forward, CPL was less likely to accurately find new instances. Because people can easily understand language and flag bad extractors, their opinion would block a great part of the noise of bad contextual patterns thus, learning fewer instances, but more accurately. One particular example of the power of the crowd to reduce semantic drift is our approach performance with the category *food* for this experiment, which can be seen in Figure 5. At iterations 1, 3 and 5, the crowd flagged the respective contexts as good *"vegetables such as X*, *"herbs such as X"* and *"spices such as X"*. These three contexts allowed CPL to learn almost completely true positive instances throughout 10 iterations. Without SSCrowd, contexts like *"served with raw X"* and *"large amount of X"* were promoted and because these contexts have a very broad range, they yielded almost as many instances as "CPL + SSCrowd". Table 2 shows the rank for the *food* contexts we discussed and some others for the categories *city* and *actor* to display the kind of assistance that comes from Twitter users. One distinct aspect is the pattern *"large amount of X"* being flagged as bad extractor. Although it could extract *foods*, it is too generic and it was rejected.

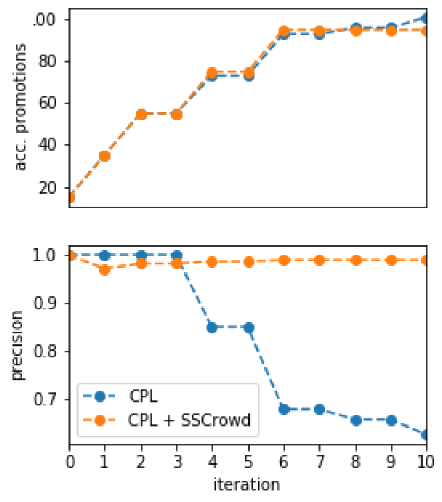


Figure 5 – Accumulated promotions and precision for the category food

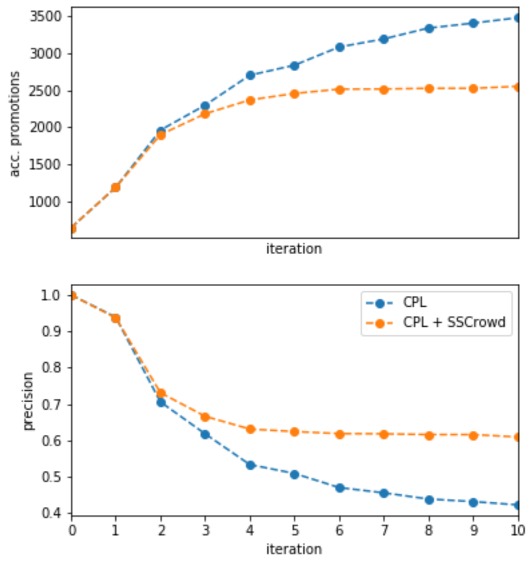


Figure 6 – Accumulated promotions and precision for all categories

	A	B	C and D
food	herbs such as X	served with raw X	large amount of X
city	X important cities like X	player born in X	X and other countries
actor	foreign actor including X	Neo played by X	is forced to by X

Table 2 – Sample of consensual votes for contextual patterns by Twitter users

5 Discussion and Future Work

In this work, we introduced the novel idea of applying a Multi-Agent System crowd-powered approach to reduce the effects of semantic drift in a bootstrap learning algorithm. We put our idea to test through an attempt to reduce the semantic drift in the CPL, a major component of NELL, one of the most successful bootstrap learning algorithm [16]. CPL extracts instances of categories and relations from unstructured text. The crowd is added to the loop of learning through SSCrowd, an algorithm that takes questions from learning systems and asks them on Web communities. We built a modified version of CPL that uses inputs from humans through SSCrowd in a student(machine) teacher(humans) MAS approach. Our results show that the ease of humans to understand ambiguity in language is an advantage to address the problem of semantic drift. The supervision of humans help to prevent CPL from learning wrongfully instances, thus reducing the semantic drift.

Conversing Learning tasks have already been performed through platforms like Twitter and Yahoo! Answers and there is an implementation of the SSCrowd algorithm for both of them. One possible sequence to this work, is to extend the SS-Crowd to work inside chatterbots [28]. Systems like this could be one approach to develop a conversing agent. Chatterbots are agents designed to chat with humans with an interface that allows conversation though typing. Usually the main goal of such agents is to pass the Turing Test [29], but in CL we can implement a conversing agent by creating a chatterbot that asks the questions defined for the CL task. The increasing market of social networks and messaging API's such as Telegram Bot API[1] and Facebook Messenger Platform[2] could make the interaction between human and machine more dynamic and create a sense of context that could drive the users to give more precise collaboration.

In Conversing Learning, intelligent systems use the crowd as high performance processors in a Multi-Agent System human-machine computer. To work with human generated content raises concerns that are known to crowdsourcing tasks, such as keeping workers motivated and assure quality on content [61]. The latest is one of the greatest challenge in the adoption of crowdsourcing in ML and could be an interesting next step to this work. One way to manage the people that responds to the system's questions is described in [24], where a model of the reliability of individual AMT workers was created by comparing their responses with gold standard responses, which allowed a voting system that leverage the responses that are closer to the gold standards. Other crowdsourcing applications also attempted to increase the confidence on the crowd by comparing the collaboration from the crowd with experts/gold collaborations [62, 63]. To determine who are the people that are experts to an subject, or have better reputation on the web could help the CL system to trust more on users answers, and maybe resolve the problem of finding confidence on a large set of unknown sources.

The most known use of CPL is its implementation as a tinker toy for NELL. In this model, NELL avoids semantic drift by receiving supervision of multiple tinker toys that share the same task of populating an ontology with instances for categories. In NELL this process requires heavy computational power and a large data set. To use SSCrowd to assist the extraction contextual patterns and category instances like we did, encourages the idea of building applications for text classification with little data, for specific domains. CPL could be used in tasks such as Named Entity Recognition, only relying on a few seed instances. In such task, CPL could use its descriptive power to assist in explaining Named Entity Recognition in Deep Learning

[1] https://core.telegram.org/bots/api
[2] https://developers.facebook.com/docs/messenger-platform

approaches, where the results achieve state-of-the-art performance, but are hard to explain. In a scenario where specialist opinion is available, text data sets are scarce or data scientists need to explain complex models, SSCrowd could be of great help to connect the machine that needs assistance to the specialists that can provide it.

Bibliography

[1] MITCHELL, T. M. *Machine Learning*. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

[2] BALCAN, M.-F.; URNER, R. Active learning–modern learning theory. *Encyclopedia of Algorithms*, Springer, p. 8–13, 2016.

[3] HANNEKE, S. *Theoretical foundations of active learning*. [S.l.]: ProQuest, 2009.

[4] SETTLES, B. Active learning literature survey. *University of Wisconsin, Madison*, v. 52, n. 55-66, p. 11, 2010.

[5] AMERSHI, S. et al. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, v. 35, n. 4, p. 105–120, 2014.

[6] DALVI, B. et al. Ike-an interactive tool for knowledge extraction. In: *5th AKBC Workshop*. [S.l.: s.n.], 2016.

[7] KULESZA, T. et al. Structured labeling for facilitating concept evolution in machine learning. In: ACM. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. [S.l.], 2014. p. 3075–3084.

[8] AHN, L. V. et al. recaptcha: Human-based character recognition via web security measures. *Science*, American Association for the Advancement of Science, v. 321, n. 5895, p. 1465–1468, 2008.

[9] BERNSTEIN, M. S. et al. Soylent: a word processor with a crowd inside. In: ACM. *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. [S.l.], 2010. p. 313–322.

[10] BERNSTEIN, M. S. Crowd-powered systems. *KI-Künstliche Intelligenz*, Springer, v. 27, n. 1, p. 69–73, 2013.

[11] MODAYIL, J.; KUIPERS, B. Bootstrap learning for object discovery. In: IEEE. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. [S.l.], 2004. v. 1, p. 742–747.

[12] KUIPERS, B.; BEESON, P. Bootstrap learning for place recognition. In: *AAAI/IAAI*. [S.l.: s.n.], 2002. p. 174–180.

[13] JONES, R. et al. Bootstrapping for text learning tasks. In: *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*. [S.l.: s.n.], 1999. v. 1, n. 7.

[14] RILOFF, E.; WIEBE, J.; WILSON, T. Learning subjective nouns using extraction pattern bootstrapping. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. [S.l.], 2003. p. 25–32.

[15] RILOFF, E.; JONES, R. et al. Learning dictionaries for information extraction by multi-level bootstrapping. In: *AAAI/IAAI*. [S.l.: s.n.], 1999. p. 474–479.

[16] CARLSON, A. et al. Toward an architecture for never-ending language learning. In: *AAAI*. [S.l.: s.n.], 2010. v. 5, p. 3.

[17] CARLSON, A. *Coupled semi-supervised learning.* [S.l.], 2010.

[18] PEDRO, S.; JR, E. H. Collective intelligence as a source for machine learning self-supervision. In: ACM. *Proceedings of the 4th International Workshop on Web Intelligence & Communities. In conjunction with WWW2012.* [S.l.], 2012. p. 5.

[19] PEDRO, S. D.; APPEL, A. P.; JR, E. R. H. Autonomously reviewing and validating the knowledge base of a never-ending learning system. In: ACM. *Proceedings of the 22nd International Conference on World Wide Web.* [S.l.], 2013. p. 1195–1204.

[20] PEDRO, S. D.; JR, E. R. H. Conversing learning: Active learning and active social interaction for human supervision in never-ending learning systems. In: SPRINGER. *Ibero-American Conference on Artificial Intelligence.* [S.l.], 2012. p. 231–240.

[21] SCULLEY, D. Online active learning methods for fast label-efficient spam filtering. In: *CEAS.* [S.l.: s.n.], 2007. v. 7, p. 143.

[22] MITCHELL, T. et al. Never-ending learning. *Communications of the ACM*, ACM, v. 61, n. 5, p. 103–115, 2018.

[23] JEPPESEN, L. B.; LAKHANI, K. R. Marginality and problem-solving effectiveness in broadcast search. *Organization science*, INFORMS, v. 21, n. 5, p. 1016–1033, 2010.

[24] SNOW, R. et al. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the conference on empirical methods in natural language processing.* [S.l.], 2008. p. 254–263.

[25] GRUBER, T. Collective knowledge systems: Where the social web meets the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, Elsevier, v. 6, n. 1, p. 4–13, 2008.

[26] MCINTOSH, T.; CURRAN, J. R. Reducing semantic drift with bagging and distributional similarity. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP.* [S.l.: s.n.], 2009. p. 396–404.

[27] MANNING, C. D. et al. The stanford corenlp natural language processing toolkit. In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations.* [S.l.: s.n.], 2014. p. 55–60.

[28] MAULDIN, M. L. Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In: *AAAI.* [S.l.: s.n.], 1994. v. 94, p. 16–21.

[29] TURING, A. Computing machinery and intelligence. *Mind*, JSTOR, v. 59, n. 236, p. 433–460, 1950.

[30] LASECKI, W. S. et al. Chorus: a crowd-powered conversational assistant. In: ACM. *Proceedings of the 26th annual ACM symposium on User interface software and technology.* [S.l.], 2013. p. 151–162.

[31] CHENG, J.; BERNSTEIN, M. S. Flock: Hybrid crowd-machine learning classifiers. In: ACM. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing.* [S.l.], 2015. p. 600–611.

[32] BREW, A.; GREENE, D.; CUNNINGHAM, P. Using crowdsourcing and active learning to track sentiment in online media. In: *ECAI.* [S.l.: s.n.], 2010. p. 145–150.

[33] SUN, C. et al. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 7, n. 13, p. 1529–1540, 2014.

[34] KAMAR, E.; HACKER, S.; HORVITZ, E. Combining human and machine intelligence in large-scale crowdsourcing. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1.* [S.l.], 2012. p. 467–474.

[35] ZAIDAN, O. F.; CALLISON-BURCH, C. Crowdsourcing translation: Professional quality from non-professionals. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1.* [S.l.], 2011. p. 1220–1229.

[36] KITTUR, A. et al. The future of crowd work. In: ACM. *Proceedings of the 2013 conference on Computer supported cooperative work.* [S.l.], 2013. p. 1301–1318.

[37] LEASE, M. On quality control and machine learning in crowdsourcing. *Human Computation*, v. 11, n. 11, 2011.

[38] BRADEŠKO, L. et al. Curious cat conversational crowd based and context aware knowledge acquisition chat bot. In: IEEE. *Intelligent Systems (IS), 2016 IEEE 8th International Conference on.* [S.l.], 2016. p. 239–252.

[39] KARGER, D. R.; OH, S.; SHAH, D. Iterative learning for reliable crowdsourcing systems. In: *Advances in neural information processing systems.* [S.l.: s.n.], 2011. p. 1953–1961.

[40] WHITEHILL, J. et al. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In: *Advances in neural information processing systems.* [S.l.: s.n.], 2009. p. 2035–2043.

[41] LENAT, D. B. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, ACM, v. 38, n. 11, p. 33–38, 1995.

[42] DONG, X. et al. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: ACM. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.* [S.l.], 2014. p. 601–610.

[43] BOLLACKER, K. et al. Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data.* [S.l.], 2008. p. 1247–1250.

[44] FU, W.-T.; LIAO, V. Crowdsourcing quality control of online information: a quality-based cascade model. In: SPRINGER. *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction.* [S.l.], 2011. p. 147–154.

[45] WANG, G. et al. Serf and turf: crowdturfing for fun and profit. In: ACM. *Proceedings of the 21st international conference on World Wide Web.* [S.l.], 2012. p. 679–688.

[46] GADIRAJU, U. et al. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In: ACM. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems.* [S.l.], 2015. p. 1631–1640.

[47] WANG, G. et al. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In: *Usenix Security*. [S.l.: s.n.], 2014. v. 14.

[48] FRANKLIN, M. J. et al. Crowddb: answering queries with crowdsourcing. In: ACM. *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. [S.l.], 2011. p. 61–72.

[49] YANG, J.; ADAMIC, L. A.; ACKERMAN, M. S. Crowdsourcing and knowledge sharing: strategic user behavior on taskcn. In: ACM. *Proceedings of the 9th ACM conference on Electronic commerce*. [S.l.], 2008. p. 246–255.

[50] BLUM, A.; MITCHELL, T. Combining labeled and unlabeled data with co-training. In: ACM. *Proceedings of the eleventh annual conference on Computational learning theory*. [S.l.], 1998. p. 92–100.

[51] YAROWSKY, D. Unsupervised word sense disambiguation rivaling supervised methods. In: *33rd annual meeting of the association for computational linguistics*. [S.l.: s.n.], 1995.

[52] KOMACHI, M. et al. Graph-based analysis of semantic drift in espresso-like bootstrapping algorithms. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. [S.l.], 2008. p. 1011–1020.

[53] CURRAN, J. R.; MURPHY, T.; SCHOLZ, B. Minimising semantic drift with mutual exclusion bootstrapping. In: CITESEER. *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*. [S.l.], 2007. v. 6, p. 172–180.

[54] AGICHTEIN, E.; GRAVANO, L. Snowball: Extracting relations from large plain-text collections. In: ACM. *Proceedings of the fifth ACM conference on Digital libraries*. [S.l.], 2000. p. 85–94.

[55] ABNEY, S. *Semisupervised learning for computational linguistics*. [S.l.]: Chapman and Hall/CRC, 2007.

[56] SUN, A.; GRISHMAN, R. Semi-supervised semantic pattern discovery with guidance from unsupervised pattern clusters. In: *Coling 2010: Posters*. [S.l.: s.n.], 2010. p. 1194–1202.

[57] MCINTOSH, T. *Reducing Semantic Drift in Biomedical Lexicon Bootstrapping*. [S.l.]: Citeseer, 2010.

[58] PANTEL, P.; PENNACCHIOTTI, M. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. [S.l.], 2006. p. 113–120.

[59] YANGARBER, R. Counter-training in discovery of semantic patterns. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. [S.l.], 2003. p. 343–350.

[60] CALLAN, J. et al. *Clueweb09 data set*. 2009.

[61] ZHAO, Y.; ZHU, Q. Evaluation on crowdsourcing research: Current status and future direction. *Information Systems Frontiers*, Springer, v. 16, n. 3, p. 417–434, 2014.

[62] GALLOWAY, A. W.; TUDOR, M. T.; HAEGEN, W. M. V. The reliability of citizen science: a case study of oregon white oak stand surveys. *Wildlife Society Bulletin*, BioOne, v. 34, n. 5, p. 1425–1429, 2006.

[63] DELANEY, D. G. et al. Marine invasive species: validation of citizen science and implications for national monitoring networks. *Biological Invasions*, Springer, v. 10, n. 1, p. 117–128, 2008.