DISSERTAÇÃO DE MESTRADO

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM

CIÊNCIA DA COMPUTAÇÃO

"Conversing Learning: Applying the Wisdom of Crowds to Assist Never Ending Learning Tasks"

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

São Carlos
Agosto/2011

Conversing Learning: Applying the Wisdom of Crowds to Assist Never Ending Learning Tasks

Saulo Domingos de Souza Pedro

August 2013

"Conversing Learning: Applying the Wisdom of the Crowds to Assist Never-Ending Learning Tasks"

Saulo Domingos de Souza Pedro

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

[illegible]
Prof. Dr. Estevam Rafael Hruschka Júnior
(Orientador - DC/UFSCar)

Roseli Ap Francelin Romero
Profa. Dra. Roseli Aparecida Francelin Romero
(ICMC/USP)

[illegible]
Prof. Dr. William Weston Cohen
(Carnegie Mellon University)

São Carlos
Setembro/2013

Abstract

In Machine Learning systems we often apply techniques are often applied to learn about real world problems behavior from data. Traditionally, such data comes from instances of datasets (that represents the target problem) which we want to learn from. This approach has been broadly used in many different application domains such as recommending systems, meteorological prediction, medical diagnosis, etc. The recent years of quick development of communications technology, that made the Internet faster and available, made possible the acquisition of information to feed a growing number of Machine Learning applications and, in addition, brought light to the use of human computation and crowdsourcing approaches commonly applied to problems that are easy for human but difficult for computers. Thus, the Social Web has been the focus of many research in Artificial Intelligence and Machine Learning. In this work we want to show how we can take advantage from the Social Web to add value to Machine Learning (ML) systems which can actively and autonomously ask for web users help to improve learning performance. This work proposes a model of learning called Conversing Learning that is is based on both, Active Learning and Interactive Learning, and is intended to allow machines to convert its knowledge base into human understandable content and then, actively and autonomously ask people (Web users) to take part into the knowledge acquisition (and labeling) process. The work presents how to apply Conversing Learning (CL) tasks to assist ML tasks, and discusses the success of experiments exploring the subtleties of this model.

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 1, 5, 9, 35, 60

AL Active Learning. 1, 3, 5, 6, 13–15, 35–37, 58, 60, 61

API Application Programming Interface. 21, 23, 39

CL Conversing Learning. 1, 3–6, 9–12, 14–16, 19, 30–41, 45–48, 50–54, 56–61

FAQ Frequently Asked Questions. 7

HCI Human-Computer Interaction. 57

IL Interactive Learning. 1, 6, 35, 36, 40

IR Information Retrieval. 10, 15, 16, 38

KB knowledge base. 10, 12–14, 16, 19, 20, 24–26, 29, 30, 33, 37, 39, 42, 43, 46–50, 52, 53, 56–59, 61

ML Machine Learning. 1, 3, 5, 6, 9–13, 16, 19, 20, 28, 35–38, 40, 41, 46, 49, 52, 57, 58, 60, 61

NELL Never-Ending Language Learner. 1, 4, 6, 7, 13, 19, 20, 22, 24–26, 29–31, 33, 34, 37, 41–56, 58, 59, 61

NLP Narural Language Processing. 15, 38, 40, 60

QA Question Answering. 3, 7, 15–20, 23, 25, 28, 30, 32, 39

RHCI Reversed Human-Computer Interaction. 28, 30, 57

RL Rule Learner. 13, 20–27, 29

Contents

Chapter 1

Introduction

The Artificial Intelligence research community is in a moment of new discoveries due to the recent growth of Internet, specially the Social Web. In a scenario where thousands of human generated content are created every minute, new ways to explore the volume and value of data arise.

For Machine Learning, this avalanche of information represents an opportunity to focus on real world problems with applications and methods that take advantage of large volume of data. However, processing these data is challenging and it might not be an easy task. Usually ML tasks require methods to validate the learning task, that is, a way to check whether the learning model being constructed really represents the problem being resolved.

To help ML systems to resolve this matter, the authors of this work proposed an algorithm called SS-Crowd. Such algorithm would allow ML systems to reach human members of Web communities. The idea is that, once this contact is done, the ML system would be, then, able to automatically validate its own knowledge by asking the Web communities whether that knowledge is right or not, thus allowing the ML system to have self-supervision. This approach was well received in the ML community and the challenges of its building process motivated the suggestion of a model of learning that allows ML systems to autonomously access human users and take advantage of their common sense to resolve ML tasks that goes beyond validation. The product of this work is such model called Conversing Learning.

This model is inspired by other Machine Learning techniques such as Active Learning [7, 25] and Interactive Learning [24]. Active Learning allows an oracle to select the most relevant subset of the training set. Meanwhile, Interactive Learning (IL) explores the oracle to continuously improve a learning system in each cycle of consulting. In the IL approach, the learning system integrates the results acquired from the last consult to an oracle and deploy an improved knowledge base as a new iteration.

Conversing Learning is based on both, Active Learning and Interactive Learning, and is intended to allow machines to be proactive and convert its knowledge base into human understandable content and then, actively and autonomously ask people (Web users) to take part into the knowledge acquisition (and labeling) process[17]. The idea of Conversing Learning was first introduced in [17, 18] and has been applied mainly to NELL (as in [16]), a Never-Ending Language Learner that is running since January 2010 [5].

The challenges of building Conversing Learning determined that designing a CL task requires the resolution of a few core issues.

It is needed to decide what is the problem that is going to be put into human consideration. These problems can include, for example, validation and revision. Another feature that needs definition is the method of conversation with these web users. So far, social media applications were the best options but there could be several other ways to perform this task such as surveys and forums. The method of conversation also includes addressing how to convert the machine knowledge into a human understandable form which might require resolution of Narural Language Processing issues, since human language is used. Finally the CL system must include a way to analyze the Web users contributions and feedback the Machine Learning system.

The main goal of this thesis is to propose this CL model, detail the issues mentioned above and discuss ideas of assignments for them through practical exemples. This thesis does not perform a deep analysis of all possibilities concerning those issues, but it shows how these core issues were first approached in research work performed during this Master's thesis construction and how CL has been applied so far. This document also discusses important issues found during its design and the challenges that are opened for deep research.

Chapter 2

Related Work

NELL is the first Never Ending Learning system described in the literature. It was developed at Carnegie Mellon University [5] and uses its acquired knowledge to learn better each day. The research team fed the system with an initial ontology and initial seeds. The system, then, takes advantage of the combination of several algorithms to continuously induce new knowledge from millions of web pages. The combination of the algorithms is itself a kind of self-supervision, but the system also counts on some shallow human supervision to ensure it is free from errors, thus avoiding concept drifting.

The idea of taking advantage of the redundancy of information from large content available on the web is focused in [9] to resolve Question Answering (QA) problems. In that work, the amount of data available on-line makes answer extraction easier and the task presents a good performance even working on large datasets and applying only simple natural language processing. Another interesting use of human generated content is presented in [4], the work applies (Frequently Asked Questions) instead of traditional text files as a source to retrieve answers for a QA problem. Also, it introduces the FAQFinder system and has an approach to reduce the costs of natural language processing to understand complex questions. The system proposed matching the user's questions with existing questions on Frequently Asked Questions files.

In this paper we explore the usage of SS-Crowd in Twitter, as well as in Yahoo! Answers communities to implement Conversing Learning. In addition, we investigate the use of a supervised learning method to learn to interpret the answers from both Web Communities. Twitter has been the focus of recent interesting researches. The work in [26] presents a network of stream-based measures called Tweetonomy that combines messages, users and content of messages to allow the measure to compare stream aggregations. Also, the work presented in [21] proposes a method to generate answers to status of Twitter users. Although the method is not intended to be a dialog machine, it succeeded in generating meaningful answers to the twitter statuses.

The collected intelligence as mentioned in [10], is the data retrieved from the social web and contains high valued information to web semantic development. Gruber suggests that the real collective intelligence comes from the creation of knowledge which is impossible to be obtained manually, and also from new ways of learning through the recombination of data from social web. Gruber describes the class of systems that can deliver at this opportunity as *collective knowledge systems* and he suggests four key properties that characterizes them. They are: user generated content, human-machine synergy, increasing results with scale and emergent knowledge.

Chapter 3

Theory

Working with a model that receives input from studies from several different areas require the understanding of their key features and an analysis of how to make different methods work together. This work did not reach deep studies from all areas related to CL, but it focused on working on the properties that tend to affect CL the most. In this chapter we present the features that were important to the development of this work, as well as some new definitions that allow CL to be consistent.

3.1 Machine Learning

Machine Learning is the area of Artificial Intelligence that studies algorithms and methods to build learning systems, that is, systems able to learn from previous experiences, and how to resolve and apply the solution to a problem. This work will show how ML tasks can have their tasks improved by Conversing Learning systems. This section presents features about modeling and designing learning systems known in the literature and briefly discuss how CL is affected by them, since CL is a learning system itself. Most of the theory concerning ML that affects CL was motivated by the work in the book Machine Learning[14].

It is common that, before leaving home, people look to the clouds, feel the temperature and air humidity and then take a decision about taking or not an umbrella. In this activity, features of the real world gives information that humans take into consideration for the decision making. Although it is not possible to guarantee it is going to rain (based only on those simple and limited features), the belief that rain is going to happen is increased depending on what is observed. In Machine Learning, the features of the real world are mapped into attributes, and the mind model considering the information from these aspects of the real world are mapped into objective functions. These functions will consider the appearance of the clouds, the current temperature and humidity to decide whether it is going to rain or not.

Information Retrieval techniques bring relevant data from a collection of documents and ML systems go beyond and uses data (that can be obtained as retrieved information) to learn about a real world problem behavior. In ML data can be processed to obtain a knowledge base[1], then the ML system can provide new information querying the knowledge base (KB).

3.1.1 Design of a Learning System

Learning systems usually represent a real world problem and a better representation of this problem might depend on a good choice of the data for the training phase. In Section 5 there is a brief explanation on the influence of selecting data to represent problems in Conversing Learning.

During the development of learning systems, once the real world problem is understood, it begins the determination of the objective functions that will guide the learning process. Such a function, considers the data from the training set as well as other capabilities such as feature selection and weighting attributes. Since the function maps a real world problem it can also be seen as a learning model.

Whatever the learning task is, the objective function must map the problem to something measurable, so the machine will be able to take decisions over this measure. In practice, this mapping often means an oracle finding relevant attributes and possible values and relations for them.

In a supervised learning process, the test phase of a learning system will return one of the possible values of the target function, also known as labels. Therefore, the determination of this

[1]Machine Learning systems does not require a Knowledge Base, but many systems are based on it.

function is key to design a ML system. In fact, the accuracy of this system can be measured comparing the labels found in the test phase (which are a subset of the target function's image) and a previously labeled (by an oracle) dataset representing the same problem.

3.1.2 Classifiers

In ML, when the real world problem is mapped to a learning model, it is possible to look for the attributes that define this problem, send them to the learning model and ask about its label. Just as it is described in the rain example in the beginning of this section. This particular observation of the world can be seen as an instance of the problem.

Classifiers are algorithms that aims to determine the result of the objective function for a set of given instances, where the learning model *knows* the structure of the instances. This is possible because the training data and the test data often have the same structure, that is, the same attributes with the same possible values.

These algorithms can be categorized by the kind of training data used to teach the learning system. In a first scenario, it is possible to have in advance a training set containing values for all weather attributes and details about the occurrence of rain from the last few months. This kind of training set configures a supervised learning, that is, the training set is labeled.

In a second scenario, it is possible to have a training set containing values for all weather attributes, but no information about the occurrence of rain. In this case, although it is not possible to determine whether it is a sunny or a rainy day, it is possible to group similar instances. In examples like this, when the training set does not have any label, the learning model is known as unsupervised.

Some systems will configure a third scenario where a large training set is given but the cost of labeling is high, so there are no labels for all instances. In such scenario, an algorithm can use the labeled instances to predict the missing labels of the remaining instances. This model of learning is known as semi-supervised[2].

Conversing Learning algorithms may include classifiers as components as shown in Sec-

[2]There are some different approaches in semi-supervised learning as shown in [28]

tion 4.2. But determining the model of learning of CL components is directly connected to the problem going to be resolved by CL and therefore it is not possible to say, without knowing the target problem, which model suits CL better.

3.1.3 Self-Supervision, Self-Revision and Self-Reflection

As aforementioned, the accuracy of a learning system can be measured by the evaluation of the target function's results during the test phase. This evaluation can be performed by an expert, that is someone with great understanding of the real problem. This activity might not scale well as the test set increases. If the learning system has some way to automatically perform the evaluation, then there is self-supervision. A Machine Learning system can acquire self-supervision by coupling different algorithms. In such scenario, each system would be responsible by the supervision of the others.

Semi-supervised learning and supervised learning are only possible when the labels for the training set are available in advance. Usually, those labels are provided by an expert and the labeling activity can also be called supervision. Therefore a system able to automatically label knowledge bases would be considered a self-supervised system.

Self-supervision in learning systems allows the identification of incoherence and inconsistency in the KB. In any of the cases, it is possible to change part of the KB to fix these mistakes and prevent them from being propagated. This revision of the KB can be performed by an automatic system that query, for instance, an external KB or by a CL task querying users for fixes in the KB. A system able to decide whether a portion of the KB should be changed or not and take actions when the change is needed can be considered to have self-revision.

Even featuring self-supervision and self-revision, a ML system can be exposed to several problems such as noise in the KB and overfitting in the training set. To cope with this matter, it is possible to change the learning algorithm to have a strategy to identify its own *difficulties*. If a system has means to find these issues and take actions to resolve them, then this system can be considered to have self-reflection. Design a self-reflection task could begin in the construction of a component internal to the learning system, which is capable of evaluating the accuracy of other components, inferring the reliability of each component and, then, taking appropriate actions, like weighting the influence of the components in the final result or adjusting the learning algorithms (substituting algorithms or learning features).

3.1.4 Never Ending Learning

NELL (Never-Ending Language Learner) is a computer system that runs 24 hours per day, 7 days per week. It was started up on January, 12th, 2010 and should be running forever, gathering more and more knowledge from the web. In a nutshell, NELL's knowledge base is an ontology defining hundreds of categories (e.g., `person, sportsTeam, fruit, emotion`) and relations (e.g., `PlaysFor(athlete,sportsTeam), playsInstrument(musician,instrument)`).

NELL's knowledge base is one of the results of the *Read the Web* project[3] (RTW). RTW aims at developing a probabilistic, symbolic knowledge base that mirrors the content of the web. If successful, this will make text information on the web available in computer-understandable form, enabling much more sophisticated information retrieval, natural language understanding, and general problem solving.

The main components of NELL are: CPL, which is described in more details in [5] and works as a free-text knowledge extractor which learns and uses contextual patterns like "mayor of X" and "X plays for Y" to extract instances of categories and relations. CSEAL is based on [27] and implements a semi-structured extractor which queries the Internet with sets of beliefs from each category or relation, and then mines lists and tables to extract novel instances of the corresponding predicate. CMC is a simple set of binary L2-regularized logistic regression models which classify noun phrases based on various morphological features (words, capitalization, affixes, parts-of-speech, etc.). Finally, the Rule Learner is a first-order relational learning algorithm similar to FOIL [20], which learns probabilistic Horn clauses from the ontology.

[3]http://rtw.ml.cmu.edu/rtw/

3.2 Active Learning

In Machine Learning, problems are solved based on learning algorithms which use previous experiences. To model a learning system that represents this problem it needs to be fed with data (previous experience). Some learning algorithms, specially supervised and semi-supervised ones, require this data to be labeled, which often means that an agent needs to assign a representative value for the instances composing the training set. Labeling an instance can be a difficult task. For a machine, it can be too resourceful and for a human it might take too long. Active Learning techniques might be applied to cope with this matter. Usually, that includes the system using some method (heuristic or not) to select a subset of more relevant instances to be appreciated by an oracle that is going to feed the system with labels, thus reducing the cost of labeling.

Although reducing the cost of labeling is the most known application for Active Learning, in this work it was applied in a different way. In Conversing Learning, specially when working with Never Ending Learning it's not possible to consult the web communities for every bit of knowledge in the KB due to many issues, sucha as web communities technical limitations, the volume of data to be labeled, etc.). Therefore, Active Learning techniques were applied to ensure that only the subset of the KB that will most benefit the learning system is going to have web users attention.

The development of Active Learning tasks are influenced by the learning task, but most of them can be described by one of the most known scenarios that are *membership query synthesis, selective sampling* and *pool based Active Learning.*

In the *membership query synthesis*, any unlabeled instance is going to have the oracle attention, which can be dangerous when costs of the labeling process itself is high, but suitable when heuristic methods for selection are higher.

The *stream based selective sampling* also known as *selective sampling* which can be defined as methods that select the instances going to be presented to the oracle appreciation (web users in this work). There are several methods to select which instances are going to be verified and most of them use the amount of information of the instance as seen in *commitee-based* *sampling*[8]. Thus, the oracle will only see instances that strongly represent the problem and have higher impact in the learning task.

The model of Active Learning used in this work is mostly aligned with the *pool based Active Learning.* This method works just like *selective sampling*, but instead of applying the heuristic to the instances sequentially, it will evaluate and rank the whole collection of instances and then select the instances that are going to be appreciated by the oracle. This approach adds an overhead to process group evaluation and ranking, so this cost needs previous analysis, as it might become a bottleneck.

The experiments executed in this work count on human web users and the literature covers some key factors regarding costs and benefits when working with Interactive Learning based models.[23]. The heuristics and the attention to machine and human interaction influenced the design of CL and can be summarized as follows:

- When the annotation task is very simple for a human, one can assume that the annotation cost is constant over the instances. If that is the case, the AL task can ignore the costs and use other parameter to select the subset of instances that are going to take part in the learning process. However, if that is not the case (which is more usual), to ignore the costs of annotating instances may have a performance that is similar to random selection methods
- Costs to evaluate an instance may vary from person to person
- Evaluation costs may contain random elements. Two of the most present factors are: delays that comes from fatigue and latency of human oracles and pauses that are variation of the response time of the human oracle.

3.3 Question Answering

Traditionally, Information Retrieval systems works on returning documents for searches and context analysis. Although these systems can bring the information needed by a human or a machine, they may not focus on answering specific questions. Question Answering systems works in this matter and are designed to perform such task. In a QA system, it is necessary an analysis of the question, the search for an answer and finally the returning of the answer to whom is asking. Therefore the study of Question Answering techniques are closely related to Information Retrieval (IR) and Narural Language Processing (NLP). The problem being resolved with QA should determine features of these techniques such as the model of the question. In this work the models applied were the *factual questions* and *the opinions*. The former model looks for specific answer in questions starting with "What" and "Who", while the latter performs asks for content that depends on the interlocutor judgment such as yes/no questions. Question Answering is broadly used in applications with restricted domain such as consulting a database of books or planning a trip by asking which are cool places to visit next summer. However, in this work the focus is going to be on Question Answering with unrestricted domain since most of the experiments regarding Conversing Learning executed here are based on a knowledge base with unrestricted domains. In order to evaluate QA systems, a few criteria are available in the literature[3].

Relevance The response to the question must contain the answer to the question.

Correctness The answer must be correct.

Concision The answer must not contain irrelevant or redundant information.

Completeness The answer must be complete.

Coherence The answer must easy to understand.

Justification The answer must have enough content so the interlocutor can determine why it was chosen.

From the above described criteria, the relevance has particular importance in this work. Although it is expected that CL tasks interacts with users using any method, questions are the only really tested so far. Therefore, it is imperative that the answer is connected to the question (even when it is a wrong answer). Section 4.1 presents a way to workaround wrong answers by leveraging answers with a higher tendency to be right. But, if there is no answer at all, the CL task will be even more challenging. Usually, IR literature refers to *recall* as the relevant instances retrieved from a collection, in this work, however, we apply the same term to refer to the portion of answers that are usable, that is, answers that contain an answer linked to the question.

3.3.1 Applying Question Answering to Conversing Learning Tasks

Usually, Question Answering systems will query a knowledge base to answer from a question given by a human. In Conversing Learning, in the contrary, the system asks the web community and uses its answers to improve a Machine Learning task. Conversing Learning systems, require a component to connect with web users. In this work, such component is called Conversing Agent and it will be covered in Chapter 5. The connection between the machine and the web communities can be achieved by converting the machine knowledge into human understandable content. In this work, this understandable content are questions. Therefore, instead of adopting a Deep QA strategy, the Macro Reading idea taken from [15] is used to define a Macro-QA task.

Consider that "Micro-QA" can be defined as in *Definition 1*:

Definition 1: "Micro-QA" can refer to the traditional QA task (or even the "Deep QA" approach) where a single question is given as input, and the QA system must fully understand every bit of information present in the natural language sentence used as question. The desired output of such a system is the full information content in form of a natural language sentence that can be used as an answer.

In contrast, *"Macro-QA"* was defined as follows:

Definition 2: "Macro-QA" is a task where the input is a set of questions (paraphrasing a single target question), and the QA system must try to get the general idea embedded in most of the questions. The desired output is a simple answer (e.g yes or no) that reflects the main idea behind the majority of the given questions.

Macro-QA tends to be much easier than **Micro-QA** because when analyzing a set of questions (instead of a single question) two main issues are raised. First, **Macro-QA** does not require "understanding" every bit of information embedded in all questions. If a specific question is given in a very complex form the system can simply discard it and focus in the questions that are easier to be automatically interpreted. Second, facts will be stated redundantly, using different wordings. In this sense, a **Macro-QA** can benefit from this redundancy by focusing on analyzing only the simple wordings, ignoring hopelessly complex sentences, and by statistically combining evidence from many text fragments in order to determine how strongly to believe a particular candidate hypothesis.

Definition 3: "Reversed QA" is a task where the questions are proposed by the computational system which receives a set of answers (for each question) from human users.

Based on *Definition 3*, the **Reversed Macro-QA** idea can be defined as follows.

Definition 4: A "Reversed Macro QA" task has a set of answers (to a specific question) as input, and the system must "understand" the main idea in the most simple answers (discarding the long and complex answers) and it should base its "answer understanding" also on the redundancy of the main ideas identified in the set of answers.

Following *Definition 4*, in a "Reversed Macro QA" task, for a question like "Is it true that If an athlete X has coach Z and coach Z coaches team Y, then athlete X plays for team Y?". If the system receives a set of 5 answers like[4]:

1) no.

2) no not always.

3) no it is not always true BYE

4) No, not unless you postulate that coach z coaches team y exclusively

5) athletes run jump etc they dont play for any team

It should discard the last ones (answers 4 and 5), try to get the main idea behind the simplest ones and observe the redundancy present in them.

When using the "Reversed Macro QA" approach to model a self-supervision component to a Never Ending Learning system (as done in this work) an interesting aspect is worth mentioning:

[4]Here it is shown a real example of a question generated by SS-Crowd and the answers given by Yahoo! Answers users. It was extracted from the performed experiments described in more detail in Section 4.1

the system should ask questions to human users in a way the answers can be naturally generated in a format that can be easily interpreted by a machine. In this sense, answers given in a format that is *difficult* for machine understanding should be considered bad answers (even if they are correct answers). Therefore, in the same way researchers in the Human-Computer Interaction (HCI) area [2, 11] investigate interaction between people (users) and computers focusing on the human use of the “information”. This matter opens space for investigating the “Reverse” Computer-Human Interaction (RHCI) focusing on the machine use of “information” given by humans.

Chapter 4

Preliminary Works with Conversing Learning

The following empirical investigations were extracted from already published works (developed during this M.Sc. work) and their results and observations were used to define the CL model described in Section 5. The first empirical investigation explores the use of *collective intelligence* of the Web community users to autonomously validate NELL's knowledge, thus performing a self-supervision task. The second one explores the use of different Web communities to leverage CL tasks.

4.1 Using Collective Intelligence to Allow Self-Supervision in Learning Systems

Conversing Learning employs concepts of *collective intelligence* proposed by Gruber and uses it to *proactively* assist ML tasks. One of the first attempts to take advantage of this intelligence in our research, was to bring self-supervision to learning systems by asking users from QA forums about the validity of snippets extracted from NELL's KB. The following experiments explain how the so called "wisdom of crowds" was applied to NELL's knowledge and they also describe the SS-Crowd algorithm which is going to be a core component of CL (see Section 5). The complete discussion of this work can be found in [17]

To show the potential of QA forums on helping improving NELL, these experiments focused attention specifically to one component of NELL, namely the Rule Learner. Rule Learner (RL) 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.

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*. Which is too general (does not specify which *Giants* is being referred), thus, it is wrong.

Considering the imperfectly extracted knowledge present in NELL's knowledge base, 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 Web communities to help validating NELL's beliefs in an automated way, giving NELL an extra source of information to help self-supervision.

4.1.1 SS-Crowd: Exploring the Wisdom of Crowds to Allow ML Self-Supervision

SS-Crowd is a component designed to to take advantage of the wisdom of crowds and put to test the idea of using human collaboration in proactive and autonomous learning tasks. It was named SS-Crowd (Self-Supervisor Agent Based on the Wisdom of Crowds) and can be seen as one of the methods to connect with Web users through Web communities. Its basic work flow for can be described as follows:

- Convert the information from knowledge base into understandable questions.

- Input the question on Web communities
- Gather opinion from users and combine them into a single opinion

In this empirical work SS-Crowd, was applied to combine data extracted from RL and information extracted from Yahoo! Answers Application Programming Interface (API). Algorithm 1 below shows the basic implementation of SS-Crowd for this particular experiment. More details over the model are presented below.

Algorithm 1 retrieves the user's opinion about a question

```
for all $answers$ as $a$ do
    $opinion$ is the user's opinion retrieved from answer $ans$
    if $opinion = approved$ then
        $app \Leftarrow app + 1$
    else if $opinion = rejected$ then
        $rej \Leftarrow rej + 1$
    end if
end for
$score \Leftarrow app - rej$
if $bestAnswer = approved$ then
    $score \Leftarrow score + (app + rej)/rej$
else if $bestAnswer = rejected$ then
    $score \Leftarrow score - (rej + app)/app$
end if
if $score > 0$ then
    return $approved$
else if $score < 0$ then
    return $rejected$
else
    return $bestAnswer$
end if
```

Convert rules into understandable questions.

Rule Learner uses a variant of the FOIL algorithm [19]. The input for RL is a simple set of positive and negative examples of a rule's consequent. To learn the rule *R1*:

R1*: statelocatedincountry(x, y):- statehascapital(x, z), citylocatedincoutry(z, y)*

RL uses positive and negative examples for the consequent *statelocatedincountry(x, y)* to induce the complete rule based on a separate-and-conquer algorithm. Each rule is initially learned as a general rule, and progressively the algorithm specializes it, so that it still covers many positive examples but covers few negative examples. After a clause is learned, the examples covered by that clause are removed from the training set, and the process repeats until no positive examples remain [12].

For each rule, RL calculates an estimated conditional probability P(conclusion—preconditions) using a Dirichlet prior according to following formulation:

$$P = (N_+ + m * prior)/(N_+ + N_- + m) \tag{4.1}$$

where N_+ is the number of positive instances matched by the rule in the training data, N_- is the number of negative instances matched, m = 5 and prior = 0.5.

Having a dataset containing RL induced rules, our first step was to run an algorithm to automatically create one question to represent each induced rule. One of the main focus here was to set an algorithm which would generate very direct 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. Having the above mentioned *R1* rule, the created question is:

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.

Note that the question above is not as natural as human speech. The generated questions reproduce the logic form of the rule and force the user to provide a straightforward answer, thus, complex answers that may cause difficulties on analysis can be prevented. To translate the predicate names (e.g. statelocatedincountry(arg1,arg2)) into a more "readable" one, NELL's metadata were used (e.g. "arg1 is a sate located in country arg2") available in the Read The Web project website (`http://rtw.ml.cmu.edu/rtw/kbbrowser/predmeta:statelocatedincountry`).

During tests, some users answered our questions with concerns about the question formatting. That feedback was taken into consideration to improve our question generation algorithm and to raise expectations on receiving better and more accurate answers.

With the first observations after test results, it was noticed that even with straightforward question format, some opinions could not be resolved due to complex answers. To reduce the noise of unresolved answers, it would be possible to force them to send information that would be easier to resolve. Since simple opinions are being harvested, another set of experiments were performed and the question generation algorithm was modified to force the system to ask yes/no questions. The approach is quite simple as well. Phrases like *"please just answer yes or no"* were added before the question. Using the rule (***R1***) of our last example, with this approach, the conversion would be as follows:

Question created from rule ***R1***: (please just answer yes or no) 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.

The adherence of users contribution was very good, so it was decided to consider this approach in our final results.

Input the question on Yahoo! Answers.

The Yahoo! Answers system, as well as other Web QA systems have several categories and sub-categories to input and answer questions. These categories help users to collaborate in their area of interest. In this study, category "Trivia" was used. This is a category (session) where users share information of any area of expertise, thus it was possible to harvest information asking questions where people are more likely to answer.

After posting the generated questions to the QA forum, SS-Crowd started to monitor the forum waiting for the answers to be retrieved. The answers were evaluated by matching them with regular expression patterns defining keywords. Those keywords are used as indication of an user opinion and to give to the system an impression of the user about the correctness of the question. This impression means the pertinence of the rule in the real universe. Even using this very simple approach (keywords-based), the task of extracting the opinion from the answer had good results. Examples of the extraction keywords are in Table 4.1.

The creation of the question was automatized by defining a sentence for every relation from

RL and access the Web QA system through an API and Web parsing procedures. The bottleneck for an application like the one here proposed was the limitations of the QA system which defines a maximum number of questions that could be asked in one day and the "points per question" policy which defines that each question made costs points to the user asking. These points can be obtained answering questions from other users. This limitation was overcome (in this version of the proposed approach) having a human user answering some questions whenever Yahoo! Answers asked for it.

keywords	
for approved rules	for rejected rules
yes	no
correct	incorrect
true	false
yeah	not
yup	nope

Table 4.1: Examples of the simple approach for the keywords used to retrieve user opinion from the answer.

Gather opinion from users and provide final result.

In Yahoo! Answers, the community chooses the best answer for a question, that is, the answer that better fits their need for information. This specific answer was considered by SS-Crowd. A higher weight is assigned for the best answer to reproduce the community confidence about that collaboration. Although the best answer is chosen by the community, there is a chance it does not fully represent the other answers. To address that issue, SS-Crowd assigns to the best answer a bonus as high as its agreement with the other answers. The answers are then reduced to a single opinion, which is sent to feedback NELL.

Notice that it was attempted to use the collected intelligence from the answers to provide to NELL a background of human common sense, an inference over collected data that could improve the knowledge of a computer and its self-supervision capability.

4.1.2 Experimenting with SS-Crowd

For validation purposes, SS-Crowd was ran with rules extracted from NELL and already evaluated by human inspection. The developers of NELL assigned a flag for each rule induced by RL. The flags indicate if a rule is approved or rejected, that is, if the rule truly represents reality. With these flags (being used as labels), it was possible to confront results from SS-Crowd and the results NELL has used coming from its developers.

The questions were posted on US Yahoo! Answers because it uses the same language (English) than NELL's current database, then noises due to language translation issues were reduced. It would be difficult to run every rule generated by RL because Yahoo! Answers as other Web QA system, limits the inputs that a user can do in a period. Therefore, from a total of 639 rules induced from NELL's KB, It was pick selected (based on the idea of active learning) a dataset containing roughly 10% of the rules that would most affect NELL's belief if applied to the knowledge base. A total of 60 rules were converted into questions and asked with the regular approach, as well as withe the yes/no question approach generating 120 questions and 350 answers. The main motivations to perform these specific tests were as follows:

- Observe user's participation on answering logic driven questions.
- Find out user's behavior over questions about random fields of expertise and what the common sense would tell.
- Compare user's overall opinion with specialists judgment over the rules extracted from NELL.
- Asses how the proposed approach would help NELL's self-supervision ability.

It was intended to observe results through different perspectives, driving to achieve to benefit from results as much as possible. The experiments were ran with the whole amount of answers from users and compared these results with the Yes/No question approach that simplifies the question to obtain more understandable answers. Also, it were ran experiments with the best answers only to estimate how it affects the results with the Algorithm 1

Combining Answers to Extract Final Result

In this empirical evaluation, when a rule is converted to a question, and put into the QA system, the retrieved answers can be analyzed either individually or combined. For example, if a question has five answers and one of these five answers was chosen as the best one, then there are five individual results approving or rejecting the rule and another result that is the combination of those five, considering the influence of the best answer as described in Algorithm 1. In the combined result, as answers are compared together, the influence of unresolved opinions are dissolved by approved opinions, rejected opinions and the opinion from the best answer. The Figure 4.1, shows the progression of the improvements when comparing individual and combined answers.

Although it is curious that Table 4.2 shows same total of false negatives and true negatives, this coincidence was not the focus here. Otherwise, the 21 false positives in the same table, represent 7.19% of the total of resolved individual answers. These false positives indicates that it is difficult for the users to agree with some rules previously approved by the specialists. Users have low tolerance about questions acceptance that a single hypothesis that does not match the proposed rule would lead the rule to be rejected. The universe of knowledge of the specialists and the Web community are higher than NELL's one, but the analysis of the community is more restrictive. At the moment this low tolerance of Web users was positive to the intentions. That can be explained that by showing a rule example.

Rule extracted from RL: teamplayssport(x, hockey) :- teamplaysinleague(x, nhl)

This rule represents the belief that a team that plays in league NHL, plays the sport hockey. Although it might seem obvious, users pointed that NHL could refer to New Hampshire Lacrosse, and the rule would not be true for all values of X. With restrictive judgment like this, it is possible to point to NELL the characteristics of the KB that could be improved in a self-revision task, showing for an example, that NHL could refer to other entities.

SS-Crowd can also be used to identify not applicable rules that are created by aspects of NELL's KB.

Rule extracted from RL: athleteplaysinleague(x, nba) :- (athleteplayssport(x, basketball)

Notice that, applying the rule, all basketball players should be in the NBA league. The specialists as well as the Web community agreed that this is wrong and should not be included as knowledge. A rule like this could be generated because information retrieved from Web pages do not keep large data about less known basketball players and most of the known ones plays on NBA. Besides, today NELL works with Web pages in English only, which would make even more unlikely to find players from countries where English is not spoken. Therefore, the rules were generated by RL over the same limits of NELL's database, leading to a tendency to prefer relations that makes sense in a universe smaller than the user's common sense.

		Answers from Users	
		+	-
Developers Opinion	+	103	21
	-	84	84

Table 4.2: The confusion matrix for individual results over 60 rules converted into 120 questions that received 350 answers. The unresolved answers are not considered.

Analysis of Best Answers

Since the Web community chooses the best answer for a question, it is reasonable to think about using only best answers to compose our validation, but for the Web users the best answer is the one that explains better the problem described by the question. Most of users explanations are very detailed and contains examples and citations. In our experiments, 16.84% of the total of best answers were unresolved. Complex explanations are hard to evaluate and it is too much risk to rely the whole analysis on a single answer even if it's the better one and specially if it cannot be resolved as approved or rejected. Furthermore, there is a chance that the best answer does not represent the other opinion from users. The Table 4.3, presents a comparison between the best answers and the other answers from the community.

		Not Best Answers	
		+	-
Best Answers	+	48	11
	-	43	115

Table 4.3: The confusion matrix comparing the best answers and not best answers. The unresolved results were not considered.

Yes/No Questions Against Regular Questions

Even with straightforward question format, some opinions could not be resolved due to complex answers. The Yes/No questions approach had good acceptance on Yahoo! Answers community and the total amount of unresolved answers could be dramatically reduced in comparison to regular question approach. The reduction of unresolved answers are shown in Figure 4.1. With the good results on Yes/No questions approach to resolve answers, tests were run to compare the results with the regular questions and look for any difference to estimate how to take advantage of Yes/No Questions on final results. Our results confirmed a tendency in having yes/no answers with similar accuracy results when compared to regular answers specially between the regular-individual approach and the yes/no-combined approach. These results indicates Yes/No question approach can be used without forcing the community to return different opinion. The Yes/No questions are easier to read and interpret, but it is not biased to interfere in the user's common sense. The results are shown in Table 4.4. In addition, 23% more answers were received from Yes/No questions than the total amount received from regular questions. Thus, apparently, users are more likely to collaborate when prompted to answer simple questions, which are easier to manipulate and extract answer. Considering these facts together, prompting the user to answer driven questions is a good way to retrieve high valued information from QA systems, specially if it is known exactly what kind of answer would best fit our Machine Learning task. These results contribute to the idea of Reversed Human-Computer Interaction.

Question Type	TP	TN	FP	FN	Precision	Recall	Accuracy	F-Measure
Regular Individual	41	31	7	39	0.85	0.51	0.61	0.64
Regular Combined	27	2	17	7	0.61	0.79	0.54	0.69
Yes/No Individual	62	53	14	45	0.81	0.57	0.66	0.67
Yes/No Combined	27	7	11	11	0.71	0.71	0.60	0.71
Best Answers	19	28	3	29	0.86	0.39	0.59	0.54

Table 4.4: Evaluation of effectiveness. To generate these results, the unresolved answers were not considered

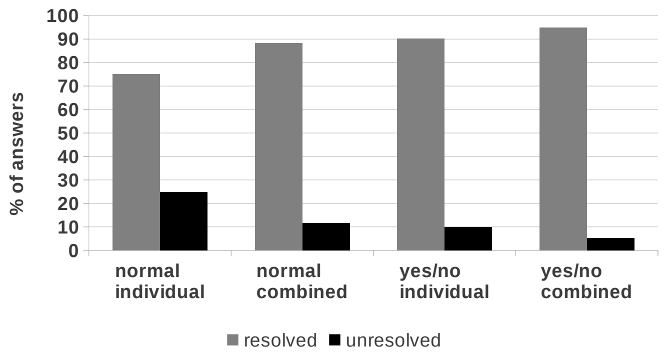

Figure 4.1: Percentage details of the comparison between regular and yes/no questions through different approaches. The approaches are individual answers and combined answers. This figure shows the progression of the improvements attempting to resolve user's opinion from answers.

4.1.3 Discussion

In a nutshell, the results of this empirical evaluation revealed that it is better to drive the user to a specific kind of answer such as Yes/No questions. This way, it is easier to extract opinions and even forcing the user to give a specific answer, our approach seems to not interfere in their common sense. Results also revealed false positives over the comparison between users and specialists in most experiments. The low accuracy values on Table 4.4, are the results of the rules approved by NELL that users don't agree to fit their common sense. The low values accuracy represents the difference between the universe of NELL's KB and the universe of web users knowledge.

Information that does not fit reality, or false positives could affect NELL more than other learning system since NELL learns infinitely and these false positives could be propagated leading the system to increase its false beliefs. The false negatives instead, is knowledge not acquired yet and although NELL learns forever, it still does not have enough data to cover it. Thus, the tendency to point false positives (revealed in the experiments) is an advantage for learning tasks.

Another interesting issue to be mentioned is that in a never-ending learning system, such as NELL, there are a number of different methods and algorithms for information extraction, learning and validation. Therefore, in a system having these properties, the Macro-QA approach becomes even more interesting. Consider, for example, a situation where, based on a rule R

SS-Crowd generates a question, posts it to Yahoo! Answers forum and cannot interpret any of the given answers (due to their complexity). The Macro-QA approach will induce the algorithm to simply ignore this rule (R), as well as its respective question and the process will continue with the next question to be generated (based on another rule). Considering that NELL has other algorithms (in addition to Rule Learner) that are also extracting rules from its knowledge base (such as the ones described in [1][12]) NELL's *Knowledge Integrator*, will have other ways to try to validate that specific rule. In other words, failing to asses the validity of a specific rul R, does not generated an error and is not a big problem to NELL. The intuition behind this example is that SS-Crowd should give feedback to NELL only on the rules that could be solved, and the unresolved rules should not have impact in the knowledge base.

4.2 Working with Different Communities

Processing human generated content is a challenging task. And the interaction that CL has with Web communities has great impact on how this content is going to be presented by the users. Therefore, to have knowledge of the behavior of a community is important to have success on CL tasks. To understand how the Reversed Human-Computer Interaction (RHCI) works in different communities, in this section the same task seen in Section 4.1 is experimented in both Twitter and Yahoo! Answers. The experiment also shows how to take advantage from accessing the *collective intelligence* of different communities. More details on modeling this experiment can be seen in [18]

4.2.1 Experimenting with Twitter and Yahoo! Answers

Although Twitter interface is not intended to perform as a QA system, users often use it to get answers for *question posts*. This experiment explores the behavior of different communities can affect the benefits of using social media as a source of information for learning tasks. As a measure of achievement, the very same rules used in the experiments from Section 4.1 were taken. The CL task received a set of 62 NELL's rules that were (automatically) converted (by SS-Crowd) into questions and then, were posted as questions in both communities. The questions generated 350 answers in Yahoo! Answers and 72 answers in Twitter. All those results were given as input to a classifier to learn how to interpret answers from both communities. Each question receives several kinds of sentences as answers and the SS-Crowd algorithm determined if those answers are approving or rejecting the the validity of the rule. If the algorithm cannot make such decision then the rule is marked as unresolved.

From the experiments, it was noticed how users collaborations differ from one community to another. In Yahoo! Answers, users are not aware that the questions are generated by a machine. In Twitter instead, that was made clear. As an instant effect, users asked (by Twitter) how restrictive should they be about the rule being evaluated. They were answered to be as restrictive as they think they should be, so the intents to not interfere in the user's opinion is kept. As Twitter users know the original intents of the question, they are more *machine friendly*, enough to try to help the learning system.

In Yahoo! Answers, people are encouraged to earn *points* and *respect* by answering questions. Users collaborate giving answers even when they are not sure about the answer. This behavior reflects in our results as a higher amount of collaboration. In Twitter instead, the collaboration has some restrictions. The user has to be *following* NELL to receive its updates (questions in our case). This means that the user is previously interested in the subject and because of this interest, while the amount of collaboration decreases, its quality increases. The example below extracted from our results explains it in a practical manner.

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)*** $\wedge$ ***(Z in Y)*** $\rightarrow$ ***(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!***

	Approved	Rejected	Unresolved
Twitter	51	17	4
Yahoo! Answers	124	168	58

Table 4.5: Total of approved, rejected and unresolved answers from Yahoo! Answers and Twitter

It can be inferred from the examples that, users from both communities are giving us the same opinion through different answers. The first contains a simple *No* answer and a justification in a logic-like format while, the latter, is pure natural language and includes an example. Everything that CL need from both answers is the *No* and since the first answer is shorter, the SS-Crowd is more accurate to extract the opinion from it. In Table 4.5, more unresolved answers were noticed in Yahoo! Answers (16.5%) than in Twitter (5.5%). It is also important to notice that the Yes/No nature of the question makes the resolution easier. This feature is based on the *driven feedback* discussed in Section 5.3 and is part of the SS-Crowd original algorithm. Overall, as illustrated in the example, one can state that if the users are different, the system is different and the answers are different.

The Conversing Learning system should be able to find how useful is the community contribution. If the human collaboration is not good enough, the system may take an action to help users to provide better feedback. This interaction between the learning system and the users aiming to allow human feedback in machine learning tasks is the main focus of Conversing Learning.

The *machine friendly* collaboration of Twitter users is good since it allows more accurate validation of knowledge. Furthermore, a QA environment such as Yahoo! Answers is more participatory and it has users from all kind of expertise and experience. There is in one side a more accurate and smaller set of answers and in the other a larger set of answers, human *identity* and an unbiased crowd. Those different biases present in each community were important to help to decide upon using these specific communities. Examining the simple sum of the totals of answers from both communities, it was noticed that users from Yahoo! Answers and Twitter have a substantial difference in their opinion. The results also pointed that users from one community disagree with users from the other community in 45% of the answers.

This increases the belief that this experiment does not deal with redundant information (but with independent sources).

A Conversing Learning system with multiple independent sources of human collaboration could use collective knowledge to improve its own ability to keep looking for information. Therefore, performing a self-revision task. To implement such capability, it was gathered information from SS-Crowd implementation with Twitter and Yahoo! Answers and represented its data as attributes to a classifier. Thus, the system is capable of assisting SS-Crowd to identify where to look for better information on web communities. The attributes retrieved from SS-Crowd are as follows: (i) Total number of rules resolved as approved and rejected by users in Twitter. (ii) Total number of rules resolved as approved and rejected by users in Yahoo! Answers. (iii) The best answer from Yahoo! Answer. (iv) The combined resolution of answers to a single question (taken from SS-Crowd, see Section 4.1).

The classifier can be used to infer more valuable knowledge from the behavior of the communities. If Conversing Learning is applied to improve a learning system KB, such classifier could bring the possibility to choose what information from the web community makes difference to that specific learning system. In this experiment traditional classifiers received the attributes from SS-Crowd and it was observed how the combination of different social media sources and the improved interaction with users, through Conversing Learning, could give us a deeper understanding of the machine knowledge validated through the eyes of humans.

The attributes retrieved from SS-Crowd created 62 instances (one for each rule) to train traditional classifiers. The learning task of the binary classifier is to identify whether a rule is right or wrong. The dataset composed by those instances were previously labeled with the judgment of NELL's developers and the tests were performed using a 10-fold cross-validation. With the outputs, it was possible to measure the relevance of the attributes. Although the average difference indicates no redundant information, it does not guarantee that every attribute is not independent. Some attributes may represent our problem better than others, and to resolve this matter, the classifiers were run in a ablation strategy removing attributes from the dataset. Therefore, it was possible to analyze how the classifier accuracy behaves in the lack of

	Classifier		
Removed Attribute	NaiveBayes	C4.5	ID3
None Removed	74.19	77.41	75.80
YahooApproved	75.80	74.19	75.80
YahooRejected	**69.35**	**69.35**	**72.58**
YahooUnresolved	75.80	79.03	80.64
TwitterApproved	75.80	61.29	61.29
TwitterRejected	74.19	83.87	77.41
TwitterUnesolved	70.96	75.80	75.80
YahooBest	77.41	75.80	77.41
YahooCombined	74.19	75.80	75.80
YahooOnly	70.96	77.41	74.19
TwitterOnly	**77.41**	**79.03**	**79.03**

Table 4.6: Classifier average correct classification rates over 10-fold cross validation using a dataset containing 62 instances

attributes.

4.2.2 Discussion

When comparing both systems outputs, it is possible to notice that Yahoo! Answers attributes are more relevant to the Conversing Learning system. The system could benefit from this inference to assign a different behavior for Yahoo! Answers collaboration, specially for the rejections, as evidenced in Table 4.6. On the other hand, when using a single community as source of human feedback, the use of Twitter brought better results than Yahoo!Answers (last two lines in Table 4.6). The *identity* of the Conversing Learning system and the knowledge of the humans about a subject might interfere in their decisions. While Twitter users gives us more straightforward response (which matches the NELL's developers), the Yahoo! Answers users gives more complex and more in-depth feedback. In a few words, while Twitter users improves a ML in self-supervision, Yahoo! Answer users help the ML system to ensure its completeness, that is, the system comprehension of all possibilities of the knowledge acquired.

In a nutshell, the results of our classifier can tell which attributes are more relevant to the rule validation task. The classifier loses accuracy when the YahooRejected attribute is removed. Implementing such a classifier can help the Conversing Learning system to tune itself in a self-supervised self-reflection task and to be able to be more effective. The classification task can also report to the ML system, which bit of the knowledge base could benefit from deeper investigation.

Chapter 5

Conversing Learning

In this work, Conversing Learning is presented as a model for learning systems instead of a learning task. In this chapter, the main characteristics and principles of this model are covered.

Popular Artificial Intelligence (AI) applications like spam trackers already presented good solutions using systems based on Interactive Learning approaches. With Machine Learning techniques, a spam tracker may have its own set of initial rules (or facts) to find spam messages. The spam trackers apply Active Learning to select candidate messages to be labeled as spam by the users. The tracker keeps interactively asking the user and updates it's policy rules to identify new spams. In the spam tracker case (following the Interactive Learning approach) the e-mail owner is the only human that can interact with the machine, and the machine has no need (and no capability) to look for help anywhere else. Therefore, the machine prompts the user with questions and *passively* waits for collaboration.

An IL spam tracker depends on the user to complete its IL task. Thus, with no interaction with other humans and no *proactive* search for extra collaboration. In Conversing Learning, on the contrary, we want the system to *actively* look for help in other sources when needed. In addition, tagging e-mails may not be a long effort task for a regular user, but other ML tasks, such as Never Ending Learning might have a large set of data to be verified (or labeled) and the opinion of a single user may not be enough to accurately feed the system. A CL system can share the validation task among several human users and use their different opinions as an advantage to explore redundancy. The core difference between CL tasks to other learning

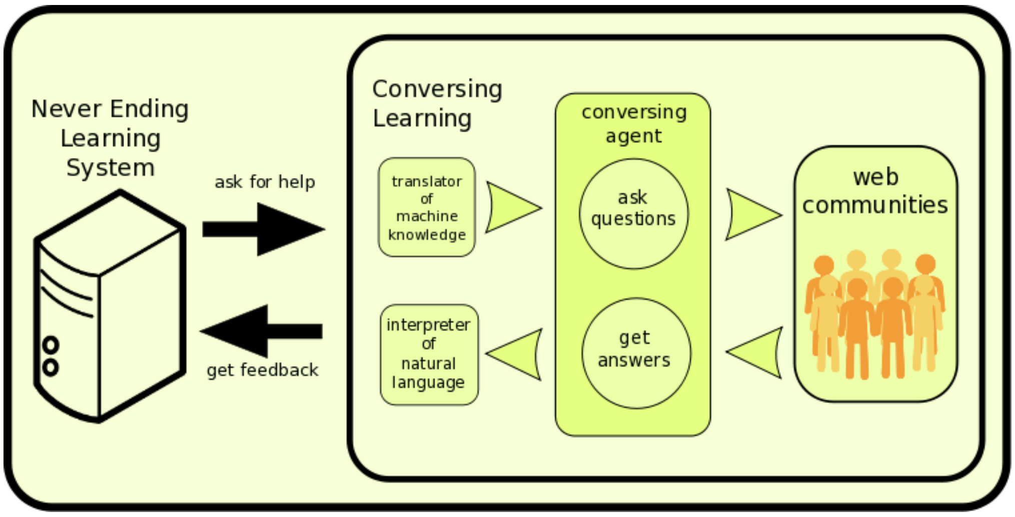

Figure 5.1: Scheme of Conversing Learning components and NELL.

tasks resides in (proactive and) automatically seeking for information from human users instead of passively waiting for their collaboration. Figure 5.1 shows the scheme of CL with NELL. Looking for help from many (and any) humans user may lead the IL task to lose precision and confidence due to noisy feedback. To rely on the human generated content, and answer the questions raised by Conversing Learning, we defined the following core issues.

- Decide **which task** is going to be put into human consideration
- In an Active Learning like method, 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**, often humans
- Propose a method to **interpret the collaboration** from the web community
- Determine **how to feedback** the ML system with the community inputs

As already mentioned, in Section 3.1, learning systems often intends to resolve real world problems. Conversing Learning is not an exception, and the determination of the target problem to be solved is the beginning of designing a CL task.

5.1 Determining the Conversing Learning Task

CL aims at assisting Machine Learning tasks. Therefore, to place a CL task, it is needed a ML system that can benefit from autonomous external assistance, often from humans. Examples of ML systems already applying CL include verification/validation tasks and metadata acquisition[1]. As described in Section 4.1, validation tasks executed by CL brings self-supervision to ML systems, because CL is intended to be automatic. If the CL task has enough accuracy and recall, it is possible to use it as a surrogate of other validation tasks. The metadata acquisition task will use CL to improve the core of a ML system. Instead of validating results, it will provide information that affects the learning model (and algorithms). If the ML system has any gaps on its modeling such as missing values for attributes, or lack of training instances a CL task might be suitable. Again, the key to decide whether the CL task is good enough to resolve a problem is to know weather it can deliver enough precision and recall.

Conversing Learning has been mostly applied to NELL because of its capability to use its own knowledge to acquire more knowledge (from the Web), forever. It is expected that a ML system with such capability would have a tendency to require more interaction with external oracles. The oracles could be providing a range of service such as validation and revision as seen in Chapter 4 and metadata acquisition as it is going to be shown in Chapter 6.2.

5.2 Choosing the Subset of Data for the Conversing Learning Task

When the KB of the learning system is large, it might be unpractical to put every bit of knowledge to human validation, therefore it is necessary to prioritize the knowledge that is going to be validated by the web community. Also, querying the users for more information than it is usually done in a given community, will constrain the user to keep collaborating because they will not be able to track all the messages from the CL task. When Machine Learning tasks *actively* select (from the dataset) the information that can leverage the learning

[1]Metadata acquisition can be used to automatically redefining learning algorithms

process an Active Learning (AL) task is configured. In CL a similar suggestion is done when portions of knowledge selected are those that bring more benefits when assessed by external oracles.

5.3 Converting Machine Knowledge into Human Knowledge

To decide the method of communication with users is critical to retrieve information with good accuracy and recall. The noises of human generated content might be complex for machine reading. In such cases, the IR algorithm may lose part of the users responses due to difficulties in extracting relevant information from the collaborations. An alternative to cope with this issue is to encourage the users to provide more "machine friendly" content, that is easier for the machine to read. Approaches like this can restrict the format of the answers provided by the users and, therefore in this work, it is called "driven feedback". Although the driven feedback concept is imperative in order to increase precision and recall, advances in NLP points to a near future with machines more capable to understand human generated content.

In Section 3.3 the principles of "Macro-QA" were introduced, and they were broadly used in the experiments of this work. However, the algorithm to create questions depends on the problem the ML system wants to resolve. For instance, metadata acquisition and metadata validation tasks can be converted to questions if the ML system has enough metadata to allow building question phrases, as seen in Section 6.2. Regardless the task, having an expert to evaluate the community behavior or to perform preliminary tests to watch the community acceptance for a query grants more collaboration from Web communities in general.

5.4 Choosing the Web Community

The experiments mentioned in Section 4.2 explores the subtleties of different Web Communities. Since the members of the communities are going to represent the oracles of 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. In this work, two concepts are highlighted to support this decision.

5.4.1 Scope of the Web Users

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 factors modifying the response taken from them. For instance, a travel recommending system that reads from a KB is better fed by users from travel communities than from a QA system with unrestricted domain. However, working with communities of a specific domain has the disadvantage of interacting with a smaller (and probably biased) set of users thus, risking to retrieve a smaller set of collaborations.

5.4.2 System Identity

The communication between humans change as the interlocutor changes. If the members of a community knows 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 Mechanical Turk, then the answer is probably more machine friendly which favors interpreting it. 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.

5.5 Designing a Conversing Agent

When the community is determined and the translated knowledge is ready to be put to users attention, a method to post the questions and wait for answers is required. The method will often rely on the API of the communities or parsing web pages for interaction. This approach requires being compliant to rate limit policies and other technical issues. In the experiments performed in this work, the Conversing Agent role was taken by the SS-Crowd algorithm. One interesting feature of the Conversing Agent is that it can be an intelligent system itself. If the CL system is going to work with multiple communities the agent can have an algorithm to decide the community that is going to receive the question as suggested in Section 4.2 or even weight the influence of different communities in the CL system. One other way to add intelligence to it is related to the nature of the Reversed Macro-QA approach. The Conversing Agent might have to combine multiple answers for a single question. The algorithm to reduce the the answers might consider subtleties of the community such as the best answers described in Section 4.1.

5.6 Interpret the Answers and Provide Feedback

After the reduction step of the Conversing Agent, the collaboration from the users must be prepared to feedback the learning system. The algorithm to feedback might require a new translation from the human understandable format back to machine data thus opening space for more NLP techniques. Sometimes reading human generated content is difficult and some collaboration might have to be discarded to avoid concept drifting when the ML system receives and applies the feedback.

In this work, a validation task usually have a single bit for every belief put to the CL task, while the metadata acquisition task required the feedback algorithm to include the metadada during a cycle of the IL system as shown in Section 6.2, in that point the CL system plays the role of the oracle of the IL system.

Chapter 6

Experiments with Conversing Learning

In this chapter, experiments using Conversing Learning as a model of learning are shown. In this sense, these experiments present case studies where a Machine Learning task is being assisted by a CL task.

6.1 Applying Conversing Learning to Assist Prophet's Outlier Detection System

As seen in Section 3.1.4, NELL is a system that learns from the Web, and uses its acquired knowledge to keep learning better each day, forever. Link prediction has been successfully used (in previous works) with Prophet[1] [1]. Prophet has a well-defined process to create new rules for NELL based on link prediction. Some of these predicted links, however, do not attend to all the requirements to have enough confidence and to be promoted as beliefs in NELL's KB . Therefore, Prophet flags these links as misplace edges (or misplaced connections). This section presents Prophet and the problem of validating misplaced edges with a Conversing Learning task.

[1]Prophet is a component to infer new rules for NELL with graph mining techniques

6.1.1 Understanding Prophet

Prophet [1] is a component that can coupled to NELL, and based on link prediction techniques, can be used to infer new rules from NELL's KB (which, in this case is represented as a graph). Prophet has two main goals, the first one is the same goal present in RL (described before): to discover new rules based on patterns present in NELL's KB. The main difference between these two methods is that, instead of implementing first order logic induction (as done in RL, Prophet is based on graph mining techniques and implements link prediction to infer new rules and instances, which can help increasing the size of knowledge base inferring new knowledge from facts already stored in the KB. The second main goal of Prophet is to discover new relations to extend NELL's KB. Both goals are achieved using algorithms based on link prediction.

Link prediction can be defined as follows: *Given a snapshot of a network (or graph) at time t, seek to accurately predict the edges that will be added to the network during the interval from time t to a given future time t'* [13]. However, when Prophet predicts new links on NELL's graph-based KB, a few previously known facts might represent negative evidence of the existence of the new edge, and in such cases those previously learned facts will be flagged (as mistaken ones).

Since the relations and categories in NELL are mapped in an ontology-based format, converting NELL's knowledge base to a graph is a natural process [1, 12], and this transformation allows the use of graph properties to investigate whether facts learned by NELL are correct or not. As an example, consider the situation where Prophet predicts a new link that connects every `sport` with a `sportsLeague` and, consider also, the presence of two previously learned facts (in NELL's KB): i) ***Cristiano Ronaldo*** *plays* ***soccer*** (represented by the link connecting both ***Cristiano Ronaldo*** and ***soccer***); ii) ***Cristiano Ronaldo*** *plays in league* ***NBA*** (represented by the link connecting ***Cristiano Ronaldo*** and ***NBA***). In this specific situation (graphically represented in Figure 6.1), human supervision can correctly identify that the link `(soccer, Cristiano Ronaldo)` is correct, but `(Cristiano Ronaldo, NBA)` is a misplaced connection.

Considering the complexity of the problem of identifying misplaced edges, in cases like the one represented in Figure 6.1, both edges are flagged, by Prophet, for future human supervision.

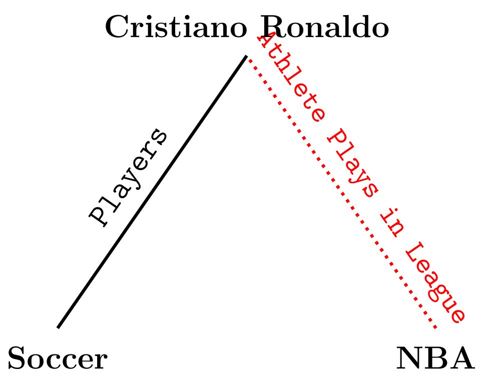

Figure 6.1: An example of misplaced edge - the connection between "Cristiano Ronaldo" and "NBA" (taken from [1])

Currently, NELL's developers[2] are responsible for human supervision in NELL's KB and they limit this verification task in, at most, 5 minutes a day. Time restriction is imposed in order to assure that NELL is autonomous enough to self-supervise its own KB even when its developers are not available to review and validate new learned facts.

As aforementioned, NELL's KB is composed by relations in the form `SportTeam(Basketball,Milwaukee Bucks)`. In short, when trying to discover new relations (to extend NELL's KB), Prophet converts all relations currently present in NELL's KB to a graph, *rtwgraph*, where relations are edges and categories (and instances) are nodes, and find open triangles. An open triangle is formed by two relation instances (from two different relations) connected by a single category instance. For example, the relations `SportTeam(Sport, SportsTeam)` and `TeamPlaysInLeague(SportsTeam, SportsLeague)` have `SportTeam(Basketball,Milwaukee Bucks)` and `TeamPlaysInLeague(Milwaukee Bucks, NBA)` as respective instances. Those two relation instances are connect by `Milwaukee Bucks` (an instance from category sportsTeam) as presented in Figure 6.2.

In *rtwgraph*, there might be more than one independent path (of size two) connecting two nodes that are not connected by a path of size one. Thus, Prophet combines the common neighbors measure with the number of independent paths of size two to decide when a path (link) between two nodes should be created or not. Figure 6.2 shows that Basketball may be connected with NBA following more than one independent path.

[2]see http://rtw.ml.cmu.edu/people

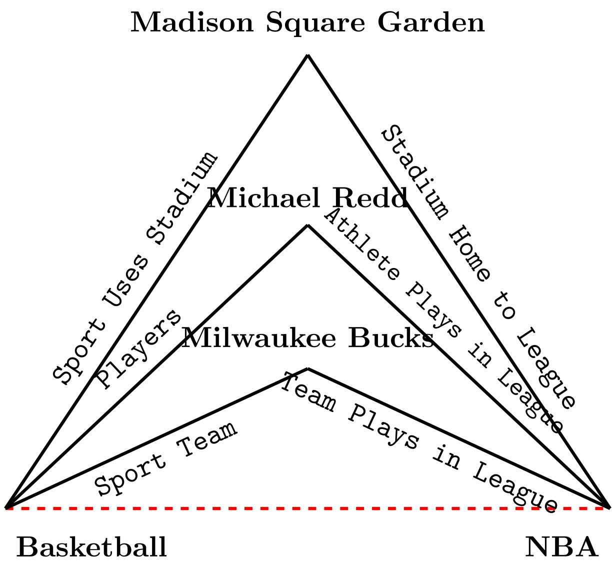

Figure 6.2: Predicates (nodes) and Relations (edges) extracted by NELL mapped into a complex network (*rtwgraph*). Two relations that share a predicate is viewed as an open triangle (taken from [1]).

Since *rtwgraph* is formed by the instances of the relations present in NELL's KB, one new relation can be predicted more than once. Thus Prophet uses all the instances to create a single new relation. However, some relations might not have a number of neighbors great or equal than the number required by Prophet or the number of independent paths is less than the number of independent paths defined by the new rules. For these new relation candidates a new edge is not created and the edges that compose the open triangle are flagged as misplaced, which means that these instances need to go through some process to be validated as correct or not. Figure 6.3 presents this situation. Suppose that Prophet create the rule `R(Sport, SportLeague)` as present as horn clauses bellow:

- `R12a(sport, sportsleague):- players(sport, athlete), athleteplaysinleague(athlete, sportsleague), numberof(athlete)` $\geq$ `10;`
- `R12b(sport, sportsleague):- sportteam(sport, sportsteam), teamplaysinleague(sportsteam, sportsleague), numberof(sportsteam)` $\geq$ `10;`
- `R12c(sport, sportsleague):- sportusesstadium(sport, stadiumoreventvenue), stadiumhometoleague(stadiumoreventvenue, sportsleague),`

`numberof(stadiumoreventvenue)` $\geq$ `10`

- `R12d(sport, sportsleague):- players(sport, athlete), athleteplaysinleague(athlete, sportsleague),sportteam(sport, sportsteam), teamplaysinleague(sportsteam, sportsleague), sportusesstadium(sport, stadiumoreventvenue), stadiumhometoleague(stadiumoreventvenue, sportsleague);`

Thus, baseball and NFL will only be connected if there are at least 10 common neighbors or all the independent paths between them. Figure 6.3 shows that not all independent paths exist, the dot line connecting the relations, `Sport Uses Stadium` and `Stadium Home to League` indicates that the connection is absent. Also, the number of common neighbors is less than 10, so the edge will not be created.

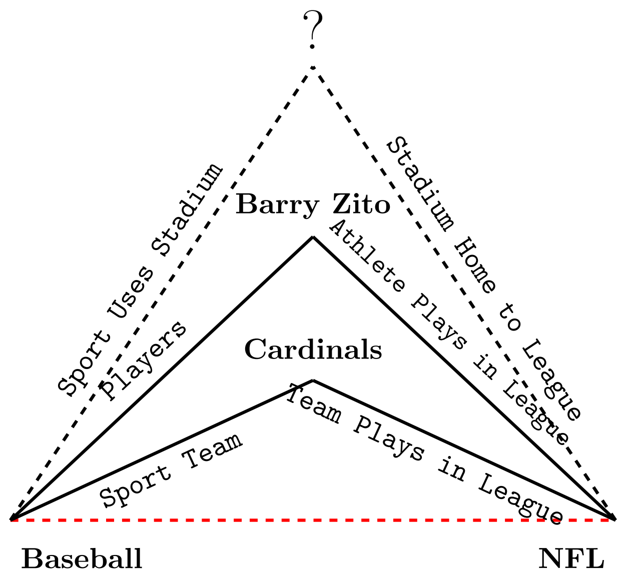

Figure 6.3: An open triangle that was flagged as uncertain because Baseball and NFL do not have the minimum number of neighbors (10) neither all the independent paths. The dotted lines represent the relations that are absent in this open triangle.

When relations are flagged as uncertain, Prophet assumes that at least one of the adjacent edges are false, which means that there might be wrong information on NELL or the adjacent edges are correct, but the edges predicted does not make sense. For example, looking at Cardinals that connect Baseball to NFL, both edges are correct but the connection does not make sense. This happens because Cardinals is a SportsTeam that plays both Baseball and

Football, so the connection between Football and NFL is correct but Baseball with NFL is not. In the aforementioned example about Christiano Ronaldo in Figure 6.1, the relation (edge) AthletePlaysInLeague is wrong knowledge, since Cristiano Ronaldo does not play in NBA.

However, it's impossible for a link prediction algorithm to decide based only in the graph structure when the edges combination make sense or not. Thus, although Prophet method seems pessimistic, it only adds relations of which it has complete certainty. This is an important issue when working with a self-supervised system where inserted errors can be propagated and generate dangerous concept drifting. Thus, when a relation is flagged as uncertain, NELL should review (self-revision) if this relation is a wrong knowledge or not. To do this is necessary a human inspection or an automatic processes.

6.1.2 Misplaced Connections and Conversing Learning

As mentioned in the previous section, misplaced edges are incorrect connections (relations) between two nodes in *rtwgraph*. When Prophet create a new relation based on at least one misplaced edge (forming the open triangle), that new relation (edge) is identified as an outlier. Anomalies and outliers are two terms used most commonly in the context of anomaly detection, some-times interchangeably[6]. Machine Learning systems usually need some kind of validation to keep the knowledge acquired in good direction. The misplaced connections, identified by Prophet are valuable information to be used as guidance for NELL's health-check.

Although Prophet knows that these outliers are the result of mismatched instances, several different reasons might be candidates to be the root cause of the mismatched connections. Thus, finding the root cause of these misplaced connections favors fixing the real issue in the future.

Conversing Learning can be used to provide an automatic interaction with Prophet and Web communities to perform outlier verification and help in self-supervision, which would be great advantage to NELL. Outliers are composed by an open triangle in *rtwgraph* (two edges connected by a single category instance). There are two possible scenarios for the outliers. The first one, is that at least one edge is a mistake. The second, is that the two edges are correct, but because of combination made by Prophet the predicted relation is a mistake. The Conversing Learning task here proposes a way to identify the root of the anomalies. Thereby, the Web community will be asked about each misplaced edge. They are going to answer whether or not the two relations that composes the open triangle fits the real world.

Prophet can provide a set of details about the edges that compose the anomalies. These details were used with the SS-Crowd algorithm as a conversing agent to ask questions to the Yahoo! Answers community. The same agent gathers all obtained answers and returns the overall opinion of the users. It's intended to use this result for further evaluation on Prophet and NELL's KB.

6.1.3 Putting the Outlier Detection System to the Test

This experiment is focused on exploring possibilities of combining the abilities of Prophet and Conversing Learning. The method counts on the Web community common sense to investigate a set of outliers identified by Prophet. It is expected to discover novel knowledge about NELL's KB and Prophet's prediction ability.

With the Web community opinions about misplaced edges in hands, it will be possible to put to the test Prophet's ability to identify anomalies. In addition, with good confirmation of Prophet's link prediction accuracy, the attention can be changed to NELL in order to explore the beliefs of the outliers and try to understand why an edge that matches most of the knowledge does not work for a few instances.

To run the experiments, a set of previously generated (by Prophet as shown in [1]) outliers were used to create a dataset of misplaced edges. The outliers are displayed as structured information. Each instance of this outlier dataset represents a misplaced link containing the two edges and three nodes that composes each outlier (open triangle), as shown in Figure 6.1 (shown in section 6.1.1). By looking at it, it is possible to notice that NELL's KB already supports the relations `Players` and `AthletePlaysInLeague`. CL task is being used, in this case, to know about a third relation connecting entities `Soccer` and `NBA`, which would be a transitive relation through entity `Cristiano Ronaldo`.

For the experiments presented in this work, it was used NELL's KB at the 100th iteration to create the undirected graph rtwgraph with 9,419 nodes and 24,132 edges. Then, Prophet was run and found new relations, instances and misplace edges. In the next step, all misplaced edges fed the CL task to start the human assessment process that will bring the common-sense opinion about Prophet accuracy when identifying misplaced concepts. Table 6.1 shows three open triangles identified by Prophet and chosen to be submitted to the conversing agent, the number of instances flagged as misplaced edges and the number of answers obtained for each pair of rule.

Table 6.1: Distribution of the relations considered in our tests

Relations	# of outliers	# of answers
`AthletePlaysInLeague & Players`	9	72
`TeamPlaysSport & TeamPlaysInLeague`	20	144
`TeamPlaysSport & TeamWonTrophy`	53	386

Converting outliers into questions for Yahoo! Answers

To illustrate the process of converting detected outliers into questions for Yahoo! Answers, consider for example that, Prophet inferred from NELL's KB that *Barry Zito* is a Baseball player and also that *Barry Zito* is an athlete that plays in NFL league. If an athlete plays an sport and also plays in a league then it's very likely that other players of the same sport would also play in the same league. Prophet uses this information amongst others to instantiate a new relation, which will be a relation between players and leagues. Without the anomaly identification, Prophet might imply that baseball players plays in NFL league. However the algorithm finds that the singular instance (Barry Zito) of our example does not fit the relation just created.

The intention here is to explore this misplaced edge, measure Prophet's accuracy when identifying anomalies and also understand why the new relation does not work for a minor set of instances.

This example deals with two relations (`Players` and `AthletePlaysInLeague`) and these relations are exactly those who needs to be explored. The Web community is going to answer whether or not these relations suit well the real world. As explained in section 6.1.1, it is expected that at least one of these relations is wrong, which would indicate a good accuracy of the algorithm to identify misplaced connections. The CL task will individually examine these relations, so that there will be one question for each relation. The SS-Crowd algorithm converts relations into questions as follows:

- Is *Barry Zito* a baseball player?
- Is *Barry Zito* an athlete that plays in NFL league?

SS-Crowd sends these questions to Yahoo! Answers and since the method depends on human interaction (guidelines from the Web community), it waits for the answers.

For this example, SS-Crowd found that, through the eyes of the community about this instance, the relation `Players` is compliant to the real world and the relation `AthletePlaysInLeague` is not, as expected. If both relations were classified as compliant then this unexpected result would be an advice to check on the predicted edge, because the unexpected result might be the consequence of another problem than the outlier identification (as mentioned in the *Cardinals* example).

Extracting information from outliers

To show that this approach can significantly enhance the benefits of the misplaced connections identification in a learning system like NELL, it was taken a set of 82 outliers from Prophet and ran SS-Crowd expecting to find at least one edge classified as wrong for each anomaly detected.

Table 6.2: Numbers for edges evaluated as suitable or not to the real world through the Web community eyes.

Outliers	
at least one wrong edge	39 (47.56%)
both edges correct	40 (48.19%)
unresolved edges	3 (03.65%)

As seen in Table 6.2, 48% of the verified outliers have two edges classified as right by the Web community.These experiments, deal only with outliers instances previously know to be real outliers, it's expected that from the total amount of outliers, at least one edge per outlier instance should be flagged as wrong by the Web community. The rate of outliers with at least one wrong edge indicates the health of the anomalies detection algorithm. Until this point, it is not possible to know what caused the creation of the misplaced edges, but which of the edges from the outlier is wrong is known. This is valuable information for the ML system because it tells developers to focus attention on the knowledge that generated the wrong edge and reduce the costs to harvest the database looking for less relevant bits of knowledge. As explained in section 6.1.1, the amount of outliers with two correct edges indicates that the detection of outliers is inaccurate. If both edges are correct, then this particular instance should not have been identified as an anomaly in the first place.

Identifying ambiguities in the KB

Since lower rates of outliers with both edges right are expected, at first, the focus was on the predicting algorithm, which expected to find problems in the outlier detecting algorithm. With deeper look, the surprisingly high total number of misplaced connections with both edges classified changed the attention of this work to this specific subset of outlier instances. Results like these might indicate, for example, that the SS-Crowd algorithm is not accurately identifying the opinion from the Web community or even that the Web community sent flawed information. A closer look on these edges revealed that most of them have a particular feature, as shown in the example below involving the relations `TeamPlaysSport` and `TeamWonTrophy`.

1. Manchester United is a team that plays sport basketball.

2. Manchester United is a team that won trophy UEFA Champions League.

The users answered that both relations are right, and instead of finding noise on Prophet's prediction it was found that Manchester United is a basketball team and also a soccer team. In this case, Prophet prediction is right as well as the validation from users. From the example above, Prophet could imply that UEFA champions league teams winners plays basketball. However, Prophet has weak evidence about that since there are few UEFA winners teams that also plays basketball and the instance was identified as a misplaced connection.

This kind of problem have a tendency to happen every time an entity is used to describe more than one meaning in the KB. In our example, NELL was not successful to decide whether Manchester United is a basketball team or a soccer team. Since NELL learns forever, eventually it will learn about more basketball teams, find more evidence about Manchester United team, and problems like this have a tendency to decrease. However, NELL intends to learn better every day, and new possibilities of misunderstandings like our example could be expected. Conversing Learning combined with Prophet can drive NELL to focus on a specific problem and enhance its knowledge supervision and verification.

Conversing Learning combines the benefits of both Prophet and NELL. It could be used by Prophet as an indicator of it's accuracy and it could be used by NELL as an indicator of the beliefs that needs attention. These algorithms have a way to provide self-supervision for a system like NELL that can take advantage of human supervision. This approach also opens doors for new experiences with Never Ending Learning systems such as self-revision through the refactoring of knowledge flagged as wrong.

6.1.4 Discussion

Link prediction has been successfully used (in previous works) with Prophet, a component to infer new relations for NELL with graph mining techniques. Prophet has a well-defined process to create new relations for NELL based on link prediction. Some of these predicted links, however, do not attend all the requirements to have enough confidence and to be promoted as knowledge. Therefore, Prophet flags these links as misplace edges which need to be revised and validated. This validation was previously done by human inspection, but in this work it was proposed a methodology that allows performing it automatically through a Conversing Learning approach. The proposed methodology allows Prophet to assess human opinion through Web communities thus configuring a self-supervision approach. The CL task in this work gathers knowledge from Yahoo! Answers, thus, allowing supervision by using the common-sense of Web users to validate learning systems.

Results obtained in the performed experiments showed that the combination of Prophet and

Conversing Learning allows a never-ending learning system (such as NELL) to identify which edges are really wrong and which edges needs more time (NELL iterations) to fill the gaps on information to be considered valid. Experiments also revealed that Prophet's outlier detection has high accuracy. In addition, it was possible to observe that most of the combination of edges that produced misplaced connections were related to a co-reference problem (noise in NELL's KB) and is not related to Prophet misbehavior itself. This analysis may be followed up to system developers, not only as set of anomalies but also as good indication of what knowledge should be verified and what could be improved in the learning system. Thus, the validation of machine learning tasks with CL is an useful approach to help self-supervision and self-revision.

6.2 Exploring NELL's Metadata with Conversing Learning

In this preliminary experiment is presented how Conversing Learning can be used to execute tasks that go beyond validation/verification for learning systems. When CL is applied to validation tasks, the conversing agent plays its role after the ML process, collecting information from a previously built KB and sending it to oracles appreciation. In that case, the Conversing Learning task and the Machine Learning tasks are processed in different time frames and are connected by a serialized chain of asking for feedback and receiving it.

In the experiments presented in this section 6.2, a different approach is explored. As already discussed, ML tasks depends on data to learn from. This data represents a real world problem, therefore, it models the mechanics of the learning task. When learning systems grows in complexity, the analysis performed over the training instances are deeper and it might be job of more than a single learning step. Applying changes in the training instances affects the mechanics of the learning method, thus it may cause significant changes to the learning process. These changes could be measured, for example, in accuracy and volume of knowledge generated. In this section is presented what we mean by *metadata* of a learning task and how is it related to the training set. Furthermore, preliminary tests on how to use CL to acquire metadata for a ML task are shown.

6.2.1 Understanding the Promotion of Concepts for Categories on NELL

For NELL, categories are a class of instances of any domain. For an example, food and bedroomItem are categories. “hamburger” and “cheese” are instances of food, while “bed” and “closet” are instances of bedroomItem. In fact, there are different kinds of bed such as ”box beds” and ”wall beds”, so bed can also be a category. In the same manner, bedroomItem is a instance of householdItem. In summary, categories are organized as a tree of classes.

In a high-level analysis, the learning process for new category instances consists in defining new categories, build a training set by assigning *a priori* values for each attribute, build a dataset representing each category based on the information of the training set, and then, let NELL run and learn new instances. For the experiments shown in the sequence, a training set with 19 attributes was used. The most important of these have significant impact into the ML and CL process and are described below. In this example, the category actor is going to be used for demonstration.

- **categoryName**: works as a category identifier. *(e.g. actor)*
- **englishName**: describes in English (natural language) what can be considered an instance of a specific category *(e.g. actors)*
- **generalizations**: a category that has actor as an instance *(e.g. person)*
- **knownNegatives**: instances that does not belong to a category *(e.g. “Steven Spielberg”, “Walt Disney” are not actors)*
- **mutexExceptions**: determines the categories that are not mutually exclusive (mutually exclusive exception) with a category *(e.g. celebrities and comedians can also be actors)*
- **seedInstances**: examples of instances for a category *(e.g. ”Clint Eastwood”, ”Jack Nicholson” are actors)*

The next step is to process the training set and build a dataset for each category. Each dataset was built through some process over the information of the training set, therefore they are a representation of a category. The dataset includes some of the attributes of the training set and provides more detailed representation of the categories such as:

- **extractionPatterns**: patterns used to recognize instances of a category within a sentence *(e.g. "actor named _", "all-star cast including _")*
- **mutexPredicates**: determines the categories that cannot be mutual with a category *(e.g. animal, emotion)*

The Training set contains data that models a category thus, it is the metadata of a category. The metadata is going to feed CPL, SEAL and the other components of NELL. With that information, NELL is ready to query the Web and match the results of the Web pages with algorithms based on the metadata. The queries to the Web are performed iteratively and after each iteration, NELL updates the metadata based on new information it collected from the Web, thus using its own knowledge to acquire more knowledge.

6.2.2 Designing a Conversing Learning Task to Acquire Metadata

This experiment explores the CL model to acquire metadata for NELL. The motivation for this comes from NELL's learning forever capability. An analysis of NELL's KB can reveal that it is possible that some categories does not have enough metadata to keep the learning process happening. When that is the case, it might be difficult to retrieve information about that category from the Web, thus leading the category instances promotion process to stall. The addition of new metadata can put that category back on rails of learning and new information should lead NELL to new instances and the knowledge acquisition process can be back to normal.

The metadata that is going to be explored in this experiment are the seedInstances. This metadata is also present in the training set and have great influence in the knowledge acquisition process. The performed experiments intend to query the Web communities for new seeds and to see how it can assist the learning process in specific categories.

6.2.3 Creating Questions for Metadata Acquisition

In all the experiments here presented, the conversion of machine knowledge into human knowledge was the same. The conversion is based on the Reversed Macro-QA mentioned in Section 3.3. The method also depends on the training set for the categories. First, **englishName** attribute is taken, and then, it is added on the preset question format previously defined by CL developers. Preset question and final question are demonstrated below:

Do you want to help NELL? Give us examples of %! Do you have examples of %? Please, help us with some!"

The final question is a simple substitution with the englishName. Below is a demonstration with the bakedGood category.

Do you want to help NELL? Give us examples of baked goods! Do you have examples of baked goods? Please, help us with some!"

These question are going to be posted on Twitter. Twitter community was chosen because users that decided to follow NELL's tweets already know that a machine is sending tweets. This is important in this task because the extraction of knowledge from the answers are more complex than the validation tasks.

6.2.4 Defining the Conversing Agent

The SS-Crowd algorithm just as defined in Section 4.1 was used for the experiments with a few changes the extraction of metadata from the answers instead of extracting opinions. The tweets were posted periodically every 4 hours which is reasonable time based on previous experiments with SS-Crowd.

Seeds extraction (from twitter replies) is performed in a 3-step shallow parser. See the example below with the category clothing.

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

Answer 01: Snowshoes, rain ponchos, galoshes, sunhats, visors, scarves, mittens, and wellies are all examples of weather specific clothing!

Answer 02: pants

Answer 03: Training shoes can be worn by anyone for any purpose, but the term means to train in sports

To help with the extraction, Stanford Log-linear Part-Of-Speech Tagger developed by The Stanford NLP Group[3] was used. Answer 01 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 only get tokens where all the words in the phrase are names. Thus, wellies is not considered a seed because it is mixed with other elements that are difficult to read. Because of the same reason, the whole Answer 03 is not being considered. As explained in Section 5, sometimes reading is too difficult and a few collaborations are discarded to prevent propagating bad responses (MacroQA approach). In this experiment, complex answers such as Answer 03 are rare. This is related to the scope of the Web users. They know their answers are going to be used by NELL somehow, so users providing readable answers is expected.

In this experiment, feedback, that is, seeds after the extraction, are used to create a new training set (with the new seeds), but it could be applied directly to NELL's metadata.

6.2.5 Preliminary Results & Discussion

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 the developers ofNELL and put it to run without changing any of the seeds. NELL ran 2 iterations and created a total of 11042 new instances for the categories, also called promotions. The reason to run small number of iterations is that

[3]http://nlp.stanford.edu/software/tagger.shtml

the seeds does not have any leverage over the other metadata during knowledge acquisition process, meaning that since NELL uses its knowledge to keep learning, the influence of the seeds in the promotions have a tendency to decrease.

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, which means, not all questions were answered since the extraction algorithm faced rare unsolvable answers. The 552 new seeds helped NELL to promote 5300 concepts. Seeds defined by NELL's developers would promote 5900 concepts.

The preliminary results shows that although there are several subtleties of NELL and CL open for more deep research, using CL for metadata acquisition can be a successful task. After 2 iterations, seeds from CL helped new to promote concepts with only 10% gap from the seeds proposed by the developers of NELL. With some extension of this particular experiment, CL could be used as a surrogate of manual seed definition if the seeds sent by the Web users are accurate.

Chapter 7

Overall Discussion

This work presented Conversing Learning. A model of learning whose tasks should be capable of autonomously look for human collaboration to assist a ML task. That collaboration can be used to perform self-supervision and self-reflection tasks and maybe go beyond the idea of acquiring metadata for learning systems. One key aspect when designing a CL task is finding a way to reach external oracles, which in this work, were represented by Web community users. Researchers in Human-Computer Interaction, have worked on how users interact with machines, focusing mainly on making the user's experience more useful and friendly. In these cases, most of communication improvements targets human users. In Conversing Learning instead, humans help to improve machine tasks, which means, the application of Reversed Human-Computer Interaction[17]. In CL, the communication improvements target the machine and not the human user. To autonomously improve Machine Learning tasks based on human supervision, this work focused on an environment where the computer can autonomously get help from humans. Thus, it is important that a Conversing Learning system identifies the following questions: (i) which knowledge should be put to humans attention? (ii) Who are the humans that the machine should look for help? (iii) How to reach the humans that will assist the machine (iv) How to adapt the machine knowledge so humans can understand the KB (v) How to understand human answers? (vi) How to infer knowledge from human answers?

When defining which knowledge from a glskb to send to the CL task, one have to consider that the amount of available data and the potential of ML algorithms and techniques favors knowledge base that have a tendency to be infinite, and NELL's KB is an example of it. Therefore, it would be difficult to use the wisdom of the crowds (in a CL approach) to validate every bit of knowledge in such huge knowledge bases. Although there are millions of people to ask, focusing on more critical piece of knowledge might bring more accurate collaboration faster. Communities on the Web may change focus quickly and it might be difficult to reach human collaboration unless it is a community specially designed for Machine Learning purposes. One way to approach this matter could be the application of Active Learning tasks to select the subset of knowledge that, when put into human consideration would make greater impact on the knowledge base. This subset can be considered critical because of its potential to improve ML systems accuracy substantially. The same idea can be applied when choosing a specific subset of channels (specific Web communities). Those are examples of the principles behind CL. Also, it is important to consider that Web users are sensitive to the awareness of the intentions of the query they are being prompted. Users tend to behave differently when they know a ML research is going to use the data generated from them. We noticed that the awareness of users influences the plainness and detailing of their answers and this difference could be used in the favor of CL systems while being developed.

Initially, we have compared the opinion of Yahoo! Answers and Twitter with opinions previously annotated by NELL's specialists during the human supervision procedure (see Section 4.2). The results from that comparison, were not enough accurate to attest the use of Conversing Learning methods as surrogate of NELL's current verification procedure. It could be used, however, to point out subsets of knowledge that needs attention, thus performing a self-supervision task. Such characteristic could also be extended to a self-revision task. Considering, for instance, adding a new step to CL, in which Web users are asked to give their help to correct knowledge flagged as wrong, in this sense, the wisdom of the crowds could be used as a reviewing process.

Moreover, when working with different sources of human generated content, if the channel of communication and its limits and features are understood , then specific issues can be directed to specific communities. Following along these lines, it was noticed that Yahoo!Answers users give more attention to details and provide more complex answers. Twitter users, on the other hand, are more straightforward (mainly because they are NELL followers, thus, they are motivated to help NELL to learn better). With a deep understanding of these capabilities, it is possible to CL systems to know where to look for human assistance depending on the problem being assessed, thus making possible a future raise of self-reflection tasks in the CL system itself. Applying Conversing Learning to improve learning tasks can bring more challenges to crowdsourcing research. The CL capability of actively asking for human collaboration allows the creation of autonomous tasks such as validation and knowledge acquisition which are critical for systems that learns continuously.

NELL's architecture allows the system to generate constraints that will guide the learning process, thus, avoiding wrong concepts to be inserted in the KB. To be even more precise, and also, to review and verify whether the facts stored in its KB are correct or not, currently the system can take advantage of four different approaches: i) ***Human Supervision***: RTW[1] group members can spend 5 minutes per day validating NELL's extractions; ii) ***Web Querying***[22]: NELL can query the Web on specific facts to verify correctness, or to predict the validity of a new fact; iii) ***Conversing Learning***[18]: NELL can autonomously talk to people (have conversations) in Web communities and ask for help on validating specific facts, rules or meta-data (features, mutually exclusiveness relationships, etc.); and iv) ***Hiring Labelers***: NELL can autonomously hire people (using Web services such as Mechanical Turk, by Amazon) to label data and help the system to validate acquired knowledge.

[1]http://rtw.ml.cmu.edu

Chapter 8

Future Work

Conversing Learning is a learning system that is inspired in other models of learning and it has just began to be developed. Designing CL tasks goes through AL and question answering analysis and maybe NLP and Never Ending Learning steps. Because of such interaction with many areas of AI and ML, there is room for improvement in several aspects. This section describes some of the uncovered tasks that are part of the plan of the author of this work.

So far, the Conversing Agent described in Chapter 5 has worked with SS-Crowd as a component to ask questions and retrieve answers. In Section 4.2 are described efforts to turn the Conversing Agent into a learning system itself. That work is intended to be extended and to map more attributes from the communities to improve the way the Conversing Agent interact with users. From preliminary experiments it was possible to note that recall is affected by the following factors: when questions are asked in daytime; questions are made to users that are more responsive and users that has better quality answers; when unresolved questions are reformulated and asked again. These factors can be mapped into attributes for the CL Agent and configure a classifier, for instance. Furthermore, depending on the quality of the mappings it might be possible to turn the CL Agent into a Never Ending Learning system.

The Conversing Agent performs an Active Learning task to reduce the portion of the KB that will be asked for the users. So far this activity has been focused in some static inference over the KB, that is, the AL task is fixed and always performed the same way. This work performed shallow analysis of the scenario where the CL task can check on the learning system and autonomously identify which are KB needs and then use its learning model to identify which are the best question, the best community and the best users to ask. A system with such a capability would be able to automatically understand its gaps and then look for help through a CL task, thus configuring a self-reflection task.

Another step that is important and uncovered for Conversing Learning are deep validation with NELL. From the experiments performed in this work, it is possible to understand that CL can be used to assist ML tasks, but how much that assistance impacts the ML system has not been analyzed. This is because this work is focused in presenting CL as a model. Therefore in the future, the author intend to produce work focusing on the application of CL on NELL and other ML tasks.

Bibliography

[1] A.P. Appel and ER Hruschka. Prophet–a link-predictor to learn new rules on nell. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 917–924. IEEE, 2011.

[2] R.M. Baecker. *Readings in Human-Computer Interaction: toward the year 2000*. Morgan Kaufmann, 1995.

[3] E. Breck, J.D. Burger, L. Ferro, L. Hirschman, D. House, M. Light, and I. Mani. How to evaluate your question answering system every day and still get real work done. *Arxiv preprint cs/0004008*, 2000.

[4] R.D. Burke, K.J. Hammond, V. Kulyukin, S.L. Lytinen, N. Tomuro, and S. Schoenberg. Question answering from frequently asked question files: Experiences with the faq finder system. *AI magazine*, 18(2):57, 1997.

[5] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41:15:1–15:58, July 2009.

[7] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[8] I. Dagan and S.P. Engelson. Committee-based sampling for training probabilistic classifiers. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 150–157. Citeseer, 1995.

[9] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298. ACM, 2002.

[10] T. Gruber. Collective knowledge systems: Where the social web meets the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):4–13, 2008.

[11] J.A. Jacko and A. Sears. *The human-computer interaction handbook*. L. Erlbaum Associates.

[12] N. Lao, T. Mitchell, and W.W. Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. Edinburgh, Scotland, UK.: Association for Computational Linguistics*, pages 529–539, 2011.

[13] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, New York, NY, USA, 2003. ACM.

[14] R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. *Machine learning: An artificial intelligence approach*, volume 1. Morgan Kaufmann, 1985.

[15] T. Mitchell, J. Betteridge, A. Carlson, E. Hruschka, and R. Wang. Populating the semantic web by macro-reading internet text. *The Semantic Web-ISWC 2009*, pages 998–1002, 2009.

[16] S.D.S. Pedro, Ana Paula Appel, and E.R. Hruschka Jr. Autonomously reviewing and validating the knowledge base of a never-ending learning system. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 1195–1204. International World Wide Web Conferences Steering Committee, 2013.

[17] S.D.S. Pedro and E.R. Hruschka Jr. Collective intelligence as a source for machine learning self-supervision. In *Proceedings of the 4th International Workshop on Web Intelligence & Communities. In conjunction with WWW2012*, page 5. ACM, 2012.

[18] S.D.S. Pedro and E.R. Hruschka Jr. Conversing learning: active learning and active social interaction for human supervision in never-ending learning systems. In *Submitted to IBERAMIA2012*, page 10, 2012.

[19] J. Quinlan and R. Cameron-Jones. Foil: A midterm report. In *Machine Learning: ECML-93*, pages 1–20. Springer, 1993.

[20] J. R. Quinlan and R. M. Cameron-Jones. Foil: A midterm report. In *Proc. of ECML*, 1993.

[21] A. Ritter, C. Cherry, and W.B. Dolan. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 583–593. Association for Computational Linguistics, 2011.

[22] Mehdi Samadi, Manuela Veloso, and Manuel Blum. Openeval: Web information query evaluation. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI-13)*, 2013.

[23] B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1069–1078. Citeseer, 2008.

[24] Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proc. of the EMNLP'11*, pages 1467–1478, Edinburgh, 2011. ACL.

[25] Burr Settles. *Active Learning.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool, 2012.

[26] C. Wagner and M. Strohmaier. The wisdom in tweetonomies: Acquiring latent conceptual structures from social awareness streams. In *Proceedings of the 3rd International Semantic Search Workshop*, page 6. ACM, 2010.

[27] Richard C. Wang and William W. Cohen. Iterative set expansion of named entities using the web. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 1091–1096, Washington, DC, USA, 2008. IEEE Computer Society.

[28] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2:3.