OpenEval: Web Information Query Evaluation

Mehdi Samadi
Computer Science Department
Carnegie Mellon University
Pittsburgh, USA
msamadi@cs.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, USA
veloso@cs.cmu.edu

Manuel Blum
Computer Science Department
Carnegie Mellon University
Pittsburgh, USA
mblum@cs.cmu.edu

Abstract

In this paper, we investigate information validation tasks that are initiated as queries from either automated agents or humans. We introduce OpenEval, a new online information validation technique, which uses information on the web to automatically evaluate the truth of queries that are stated as multi-argument predicate instances (e.g., *DrugHasSideEffect(Aspirin, GI Bleeding))*). OpenEval gets a small number of instances of a predicate as seed positive examples and automatically learns how to evaluate the truth of a new predicate instance by querying the web and processing the retrieved unstructured web pages. We show that OpenEval is able to respond to the queries within a limited amount of time while also achieving high F1 score. In addition, we show that the accuracy of responses provided by OpenEval is increased as more time is given for evaluation. We have extensively tested our model and shown empirical results that illustrate the effectiveness of our approach compared to related techniques.

Introduction

A wide variety of complementary trends have started changing the transformation of the web, from being for humans only, to also providing machine-processable data that can be used by external applications. "Semantic Web" and "Web of Trust" are two examples of these trends that have attempted to reach this goal. Other significant contributions have been made by automatically constructing machine-readable knowledge bases by extracting relational facts and rules from large text corpora such as the Web. Open information extraction techniques (Etzioni et al. 2011; Fader, Soderland, and Etzioni 2011; Mausam et al. 2012; Carlson et al. 2010), and generic relation extraction/classification techniques (Cimiano, Ladwig, and Staab 2005; Zhang 2004; Fang and Chang 2011) are a few examples of these attempts. These techniques usually get an input initial ontology, which specifies the semantic categories and semantic relations, along with a set of unlabeled web pages. Given this input, they then apply large-scale learning techniques to learn to extract or classify new instances of these categories and relations.

Despite the wide success of the current Information Extraction (IE) techniques, most of them are designed for offline batch processing and to ensure high precision (i.e., few false positives) knowledge extraction. This high precision may result in low recall (i.e., many false negatives), making them less applicable to online applications that require high recall. The online aspect and high recall are especially important when a human or an external application, such as an automated agent, wants *to query* the knowledge base *within a limited time*. For example, a human or a pharmacological agent may need to know if a drug has a specific side effect. The knowledge acquisition technique should verify if a drug has a side effect within the time that the agent or the human can wait.

In this paper, we contribute a novel technique to respond to predicate-based queries within a limited amount of time while also achieving high recall (at a small cost of sacrificing precision). We focus on how to determine the value of a given proposition by devising and implementing a new learning approach, OpenEval. OpenEval evaluates the correctness of queries that are stated as multi-argument predicate instances (e.g., *DrugHasSideEffect(Aspirin, GI Bleeding))*). It trains a classifier by taking a small number of instances of the predicate as an input and converting them into a set of positive and negative Context-Based Instances (CBI), which are used as training data. Each CBI consists of a set of features constructed by querying the open Web and processing the retrieved unstructured web pages. To evaluate a new predicate instance, OpenEval follows a similar process but then gives the extracted CBIs to the trained classifier to compute the correctness probability of the input predicate instance. To navigate the diversity of information that exists on the Web, we have presented a novel exploration/exploitation search approach, which enables formulating effective search queries and increases the accuracy of responses provided by OpenEval.

We test OpenEval on a wide range of predicates chosen randomly from Freebase (Bollacker et al. 2008), a large semantic database of several thousand categories and relations. The baselines for our comparison are the Pointwise Mutual Information (PMI) technique, (Turney 2001), and weakly-supervised classification approach (Zhang 2004). We show that OpenEval significantly improves the F1 score on the test data compared to the baseline techniques. OpenEval is also in use by a real mobile service robot (Samadi, Kollar, and Veloso 2012; Kollar, Samadi, and Veloso 2012). When asked to get an object, of which the robot does not know the location, it autonomously generates queries to OpenEval as instances of a trained predicate *LocationHasObject* with the requested object and possible known locations. The robot is capable to then plan a route to different locations according to the corresponding confidences returned by OpenEval.

OpenEval

OpenEval evaluates the correctness of propositions that are stated as multi-argument predicate instances (e.g., *DrugHasSideEffect(Aspirin, GI Bleeding)*). A predicate such as $p(x_1,...,x_n)$ defines a relationship between entities $(x_1,...,x_n)$. We call $(x_1,...,x_n)$ an *instance* of predicate p where each x_i is an argument of such predicate instance. We describe next in detail the three main components of OpenEval, namely: CBI extractor, learning, and predicate instance evaluator. We now explain in detail the CBI extractor since it is used by both learning and evaluator components.

Context-Based Instance (CBI) Extractor

Algorithm 1 shows the procedure to extract a set of Context-Based Instances (CBIs). Each CBI consists of a set of features constructed by querying the open Web and processing the retrieved unstructured web pages. The input to the *CBI Extractor* is the tuple $\langle p, i_p, k\rangle$, where p is the input predicate (e.g., *DrugHasSideEffect*), i_p is an instance of predicate p (e.g., *(Aspirin, GI Bleeding)*), and k is a keyword that is used to formulate the search query (e.g., *side effects*). The output of the CBI extractor is a set of context-based instances which are extracted for the input instance i_p.

Algorithm 1 CBI Extractor

```
Input: $\langle p, i_p, k\rangle$ //$p$: predicate, $i_p$: predicate instance, $k$: keyword
1: Function: CBIExtractor ($\langle p, i_p, k\rangle$)
2: $B_{i_p} \leftarrow \phi$ //$B_{i_p}$ is the set of all the CBIs for $i_p$
3: $pArgs \leftarrow$ Arguments of $i_p$ separated by space
4: $Q \leftarrow$ "$pArgs\ k$" //$Q$ is the search query
5: $W \leftarrow$ Retrieve the first $N$ documents for query $Q$
6: $B_{W_i} \leftarrow \phi$ //$B_{W_i}$ is the list of all the bags-of-words that are extracted from Web page $W_i$
7: for all web pages $W_i \in W$ do
8:     for all occurrences of words in pArgs (close to each other) in $W_i$ do
9:         $\bar{t} \leftarrow$ extract text around pArgs,
10:        Remove stop words and words in pArgs from $\bar{t}$
11:        Add $\bar{t}$ as a bag-of-words to $B_{W_i}$
12:    end for
13:    Add bags-of-words in $B_{W_i}$ to $B_{i_p}$
14: end for
15: return $B_{i_p}$
```

Given the input tuple, the CBI Extractor first builds the search query (Lines 3-4). The search query Q is built from arguments of the input instance i_p, and input keyword k. For example, query $Q = \{$*"Aspirin" "GI Bleeding" side effects*$\}$ is built for the predicate instance *DrugHasSideEffect(Aspirin, GI Bleeding)*, where the keyword is *side effects*. The CBI extractor then searches the query Q in Google and downloads the first N web pages (Line 5). For each Web page W_i, which the search engine finds, the CBI extractor searches the content of W_i and finds all the positions in W_i whose words in i_p appear "close" to each other (Line 8). "Close" means we allow words in i_p to appear in any arbitrary order and up to a maximum of 15 words can be in between, before and after them. For each occurrence of words in i_p, consider $\bar{t}$ as the text that occurs around words in i_p (Line 9). All the stop words and the words in i_p are deleted from $\bar{t}$ and the remaining words in $\bar{t}$ are then added as a bag-of-words into the set B_{W_i} (Line 11). Finally all the bags-of-words that are extracted from different web pages are returned as a set of extracted CBIs for input instance i_p (Line 13).

Learning

As part of training, OpenEval is given the tuple $\langle P, I, R, t\rangle$ where P is a set of predicates, I is a set of seed examples for each predicate in P, R is the mutually-exclusive relationships between predicates P, and t is the time that the learning algorithm should spend for the training. The OpenEval learning algorithm is shown in Algorithm 2.

The learning algorithm first iterates over all the predicates $p \in P$ and extracts a set of CBIs (denoted by CBI_p) for each predicate p using the input seed examples (Line 2). It also constructs another set, CBI'_p, by randomly choosing and removing 10% of CBIs from set CBI_p (Line 3). CBI'_p is used as an *evaluation* set during the learning to estimate how OpenEval performs on the future test data. A SVM is then trained for each predicate p by considering the CBIs that are extracted for p as positive examples (Line 5). To build the set of negative examples, we sample from the instances that are extracted for all other predicates that are mutually exclusive from p (Line 6). Each of these CBI examples is then transformed into a *feature vector*, where each element corresponds to the frequency of a distinct word in the CBI. The dimension of the vector is equal to the total number of distinct words that occur in the training data. We train a classifier for each predicate p so it classifies the input feature vector f as positive if f belongs to p, and classifies it as negative otherwise (Line 7). The classifier that is trained for p is later used in evaluation to decide if a new CBI belongs to predicate p.

Ideally our learning algorithm should train a classifier that has minimum entropy value for each predicate. Having minimum entropy value means that the classifier $C_{p'}$ should classify all the CBIs that are extracted for predicate p' as positive and all CBIs that are extracted for other mutually-exclusive predicates as negative. It should also assign a high confidence value to its prediction. To achieve this goal, the learning algorithm iteratively finds a classifier $C_{p'}$ that has the maximum entropy value on the evaluation set CBI' (Line 10). The training data of predicate p' is then increased by extracting new CBIs (Lines 10-15). By increasing the number of training data and retraining the classifier, OpenEval tries to decrease the entropy value of a classifier and makes it more confident in its prediction.

Algorithm 2 Learning

```
Input: $\langle P, I, R, t \rangle$ //$P$: a set of predicates, $I$: seed examples, $R$: mutual-exclusive relationships of predicates in $P$,$t$: time for training (seconds)
1: Function: Learning_Classifiers ($\langle P, I, R, t \rangle$)
2: $CBI_p$ ← Call CBIExtractor to extract CBIs for all instances of $p$ that exists in $I$ (no keyword is used)
3: $CBI'_p$ ← Randomly sample and remove a set of CBIs from $CBI_p$ //used to estimate how OpenEval performs on future test data
4: for all $p \in P$ do
5:     pos ← $CBI_p$
6:     neg ← Sample instances from $CBI_i$ $\forall$ predicates $i$ that are mutually-exclusive to $p$
7:     $C_p$ ← Train SVM using pos and neg
8: end for
9: while elapsed time ≤ t do
10:    $C_{p'}$ ← Find classifier $C_{p'}$ that has maximum entropy on $CBI'$
11:    ref ← Extract reference set from $C_{p'}$
12:    $k$ ← Choose a keyword from ref
13:    NewCBIs ←Extract CBIs for predicate $p'$ using instances $I$ and keyword $k$
14:    $CBI_{p'}$ ← $CBI_{p'} \cup$ NewCBIs
15:    Retrain classifier for predicate $p'$
16: end while
17: return $C$, reference sets for all the predicates
```

To extract new CBIs for predicate p', the learning algorithm first chooses a keyword from the reference set of predicate p'. A reference set contains a set of keywords that represent some of the underlying semantic concepts of a predicate and mostly distinguish a semantic meaning of a predicate from others. For example, keywords such as {*drug, drug effects, adverse effects*} can be used as part of the reference set for predicate *DrugHasSideEffect(x,y)*. The reference set can be built by selecting a set of relevant phrases from CBIs that are already extracted for a predicate. Different feature extraction techniques that have been studied in the machine learning community can be used to construct the reference set (Blum and Langley 1997). Among these techniques, we use feature ranking using weights from linear SVM classifier which has been shown to be an effective approach for text classification tasks (Mladenić et al. 2004). Thus, the reference set is constructed by selecting the top K% keywords that have highest absolute weights in the trained SVM (i.e., the normal to the hyperplane that separates positive and negative classes).

By iteratively choosing keywords from a reference set, OpenEval automatically learns how to map a set of entities to an appropriate predicate (i.e., sense) to which they belong. For example, given a predicate instance such as *Fruit(Apple)*, OpenEval uses keywords that are chosen from the reference set of predicate *Fruit* as part of the search query which biases the search engine to return the documents that contain information about *apple* as a *fruit*. For example, by using a keyword such as {*vitamin*}, the search query for predicate *Fruit(Apple)* would be {*"Apple" vitamin*} which forces the search engine to return the results that are more likely to mention "apple" as a fruit rather than as a company. Using a keyword as part of the search query helps OpenEval to automatically extract CBIs that specify an appropriate sense of the input predicate instance. It is worth mentioning that in the first iteration of the learning algorithm (Lines 2-8), no keyword is used to extract CBIs. However, since CBIs are extracted for *a set* of seed examples (not only one), OpenEval eventually would be able to converge to extract relevant keywords for the reference set.

Predicate Instance Evaluator

To evaluate a new predicate instance, OpenEval follows a similar process to the learning algorithm by converting the input predicate instance to a set of CBIs, but gives the extracted CBIs to the trained classifier to compute the correctness probability of the input predicate instance. The input of predicate evaluator is predicate p, the candidate instance i_p, and time t (seconds). OpenEval outputs the probability of i_p to be an instance of predicate p.

To evaluate if i_p is an instance of predicate p, OpenEval iteratively selects a set of keywords from the reference set that is built for predicate p (the reference set is built during the training) and calls the CBI Extractor to extract a set of CBIs for i_p using the selected keywords. The goal is to select a set of keywords and formulate search queries that capture different semantic aspects of a predicate. For example, predicate *UniversityProfessor(x)* is related to different aspects such as *teaching professor* and *research professor*. Different aspects of this predicate can be captured by keywords such as {*research, teaching, paper, ...*}. To decide which keywords should be chosen, we define a utility function that measures the utility value of each individual keyword using the documents that are retrieved from the Web. Given this utility function, we propose a learning algorithm that iteratively explores/exploits different keywords from the reference set, retrieves documents from the Web, and extracts CBIs from the retrieved documents.

Let *ref* $= \{k_1, \ldots, k_n\}$ be a reference set that is extracted for predicate p, where each k_i is one of the keywords that is extracted during the training. Also consider set K as a subset of keywords that are selected from *ref* and CBI_K as a set of CBIs that are extracted for input predicate instance i_p using keywords in K. To construct CBI_K, we basically go through all the keywords in K and extract a set of CBIs for i_p using the selected keyword. We express the utility of the set of keywords K as following:

$$U(K) = \left| \sum_{c_i \in CBI_K^+} conf(c_i) - \sum_{c_i \in CBI_K^-} conf(c_i) \right| \quad (1)$$

where CBI_K^+ is a set of all CBIs that are classified as positive by the classifier of predicate p and CBI_K^- is a set of all CBIs that are classified as negative. $conf(c_i)$ is the confidence value that is assigned to c_i by the classifier. The utility $U(K)$ shows how keywords in K could help to classify i_p either as positive (true) or negative (false) classes. If predicate instance i_p belongs to predicate p, then we would like to find keywords K to extract CBIs that give highest confidence value on classifying i_p as positive (large value of

$U(K)$). On the other hands, if i_p does not belong to predicate p, then we would like to find keywords that help to classify i_p as negative where they give the highest confidence on the classification vote. Thereby, our goal is find a set of keywords K that maximizes the utility value $U(K)$.

To maximize the expected utility, one naive approach is to rank each of the keywords based on its utility and always choose the keyword that has highest utility value. We call it greedy approach. This approach may deprive the opportunity of finding the optimal set of keywords K. Therefore, we face a trade-off between exploitation and exploration to gain the optimal overall utility (Gittins and Jones 1974). To achieve both exploitation and exploration goals, we rank the keywords based on their expected utility plus a term related to their variances, instead of solely using the expected utility as in the greedy approach. The term related to the variance is known as the exploration bonus (Sutton 1990). We use similar algorithm to UCB1 algorithm (Auer 2003) which aims at obtaining a fair balance between exploration and exploitation in a K-Armed bandit problem, in which the player is given the option of selecting one of the K arms of a slot machine (i.e., the bandit). In the context of our work, each keyword in the reference list *ref* is treated as an arm which can be used to formulate a search query and extract the correspondence CBIs. The selection of keywords is directly proportional to the number of times that all the keywords have been selected and inversely proportional to the number of times that each keyword is selected. We also assume that an initial reward value is assigned to each keyword which is calculated from the initial weight assigned to each keyword during the training. More precisely, we assign $Q(k)$ value to each keyword k:

$$Q(k) = \bar{R}_k + C'\sqrt{\frac{2\log(n+1)}{N_k+1}} \tag{2}$$

The keyword k that maximizes $Q(k)$ is selected at each iteration. $\bar{R}$ is the average reward that we have obtained by selecting keyword k after n iterations, N_k is the number of times that keyword k is selected. C' controls the balance between exploration and exploitation.

Algorithm 3 shows the detail of our technique for evaluating correctness of an input predicate instance i_p. We first extract the reference set for the predicate p (Line 3). The average reward for each keyword is initialized by the weight that is assigned to it in the reference set divided by the sum of all the weights of keywords in the reference set (Line 5). The algorithm then iteratively updates Q values for all the keywords (Line 8) and extracts CBIs using the keyword that has maximum Q value (Lines 9-10). The reward for the selected keyword is then calculated (Line 12) and the average reward value ($\bar{R}_{k'}$) is updated (Lines 13-15). Finally, the algorithm classifies each of the extracted CBIs (Lines 17-18). The next step of the algorithm is to combine the classification results for all the extracted CBIs.

The simplest way to combine the classification results from multiple CBIs is within a voting framework. In this framework, the correctness probability of the input predicate instance i_p is calculated by dividing the number of instances that are classified as positive by the total number of instances that are returned by the CBI Extractor, where the classifier vote is weighted by its confidence value. This scheme is known as weighted majority (or plurality) voting which has been shown to be very robust compared to other more intelligent voting schemes that have been used in forecasting literature (Clemen 1989; Dietterich 1998). The correctness probability of i_p can be written as following:

Algorithm 3 Predicate Instance Evaluator

```
Input: $\langle i_p, C, t \rangle$ //$i_p$: a predicate instance to be evaluated, $C$: classifiers trained for predicates, $t$: time (seconds)
1:  Function: Evaluator ($\langle i_p, C, t \rangle$)
2:  CBI ← {}  //set of CBIs for predicate $p$
3:  ref ← Extract reference set from classifier $C_p$
4:  $N_k \leftarrow 0\ \forall k \in$ ref  //number of times that $k$ is used as a keyword
5:  $\bar{R}_k \leftarrow \frac{\text{weight of } k \text{ in } ref}{\text{sum of weights in ref}}\ \forall k \in$ref  //initializing average reward obtained by using keyword $k$
6:  $n \leftarrow 1$
7:  while elapsed time ≤ t do
8:      $Q_k \leftarrow Q(\bar{R}_k, N_k, n)\ \forall k \in$ref  //update Q values
9:      $k' \leftarrow$ Choose keyword with maximum $Q$ value
10:     $B \leftarrow$ CBIExtractor($\langle p, i_p, k' \rangle$)
11:     CBI ← CBI ∪ $B$
12:     $R \leftarrow \sum_{[c_i \in B+]} conf(c_i) - \sum_{[c_i \in B-]} conf(c_i)$
13:     $N_{k'} \leftarrow N_{k'} + 1$
14:     $\bar{R}_{k'} \leftarrow (\bar{R}_{k'} + |R|)/N_{k'}$
15:     $n \leftarrow n + 1$
16: end while
17: Classify instances in CBI
18: $eval \leftarrow$ calculate weighted majority vote for instances in CBI
19: return true if $eval \geq 0.5$; false otherwise
```

$$P(i_p = t|\text{CBIs}(i_p)) = \frac{1}{Z} \sum_{c \in CBIs(i_p)} W_{[C_p(c)=t]} \times \mathbb{1}\{C_p(c) = t\} \tag{3}$$

where Z is the normalization factor:

$$Z = \sum_{b=\{t,f\}} \sum_{c \in CBIs(i_p)} W_{[C_p(c)=b]} \times \mathbb{1}\{C_p(c) = b\} \tag{4}$$

and $W_{[C_p(c)=t]}$ is defined as the confidence of classifier C_p where it classifies c as positive.

Experimental Evaluation

OpenEval is tested on 50 predicates that are chosen randomly from Freebase knowledge base (Bollacker et al. 2008), a large semantic database. Freebase contains 360 million instances of different relations. These predicates contain both categories (predicates with one argument) and relations (predicates with two arguments). For each predicate, 25 instances are provided as training seed examples to train OpenEval and 50 instances are randomly chosen as the test data. The test data for each predicate p consists of 25 positive examples (i.e., instances of predicate p) and 25 negative examples. The negative examples are chosen from predicates that are mutually-exclusive to p and also from predicates that are not used as part of training in OpenEval.

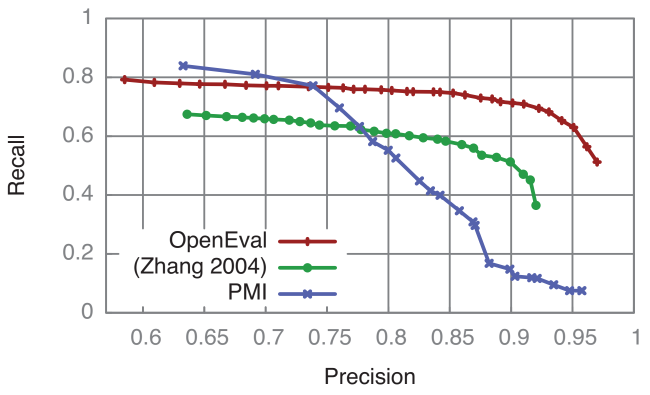

Figure 1: The precision/recall curve for test predicate instances. OpenEval uses 100 iterations for training and 5 iterations for evaluation. We also plot the curve for two baseline approaches.

We compare our technique to the baselines using standard performance metrics of precision, recall, and F1. The first baseline that we compare against is the weakly-supervised relation classification (Zhang 2004). In this work, a bootstrapping approach is used on top of SVM to classify each predicate instance to one of the predicates in the ontology. The input of SVM is a set of lexical and syntactic features extracted from the input corpus. To be able to do a fair comparison between Zhang's approach and OpenEval, for each predicate instance in the training and test data, we search the arguments of such predicate instance in Google and crawl the first N returned web pages. N is set to be the same as the number of web pages that are used in OpenEval. The list of predicates, seed examples for each predicate, and web pages crawled from Google are given as an input to Zhang's self bootstrapping approach.

The other baseline for our comparison is the PMI technique. Given a predicate p and a predicate instance i_p, the PMI score is calculated as the following (Turney 2001):

$$PMI(i_p, p) = \frac{|Hits(i_p, p)|}{|Hits(p)| \times |Hit(i_p)|} \quad (5)$$

where $|Hits(p)|$ and $|Hit(i_p)|$ are the number of search engine hits for query p and i_p, respectively, and $|Hits(i_p, p)|$ is the number of hits when both i_p and p appear in the search query. To evaluate correctness of an instance i_p of predicate p using the PMI technique, we calculate $PMI(p, i_p)$ and if $PMI(p, i_p) > Threshold$, then we accept i_p as an instance of p.

Figure 1 shows the precision/recall curve of OpenEval, PMI, and Zhang's self bootstrapping approach (Zhang 2004) for 50 predicates chosen randomly from Freebase knowledge base. The experiments are obtained when OpenEval is trained with 100 iterations and CBIs are extracted by crawling the first 5 web pages from Google. Each iteration of training is one time step in Algorithm 2 (Lines 9-16) and on average takes 43 seconds in a 8-core 2.67 GHz CPU workstation. Also the value of parameter C' in predicate evaluator is set to 3. OpenEval uses 5 iterations (Lines 8-17 in algorithm 3) to evaluate a new predicate instance. The graphs show that for most of the different recall values, OpenEval achieves significantly higher precision compared to PMI. Comparing the result of OpenEval and baselines, OpenEval has achieved the best F1 score of 80% while the best F1 scores of PMI and Zhang's approach are around 74% and 70%, respectively. One should note that the F1 score of OpenEval can be improved by using more iterations in the training and evaluation (as will be shown in the next results).

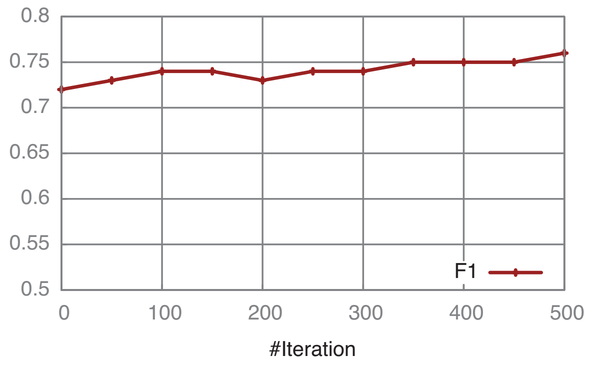

Figure 2: F1-score of OpenEval when it uses different number of iterations for training.

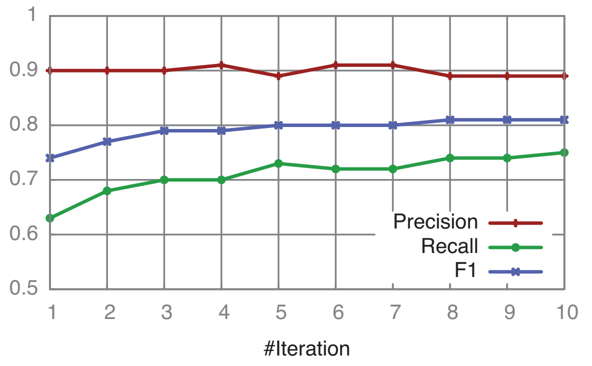

Figure 3: Precision, recall, and F1-score of OpenEval when it uses different number of iterations for evaluation.

The accuracy of OpenEval depends on the parameters such as the number of iterations that are used in the training. A comparison of different values of these parameters is important for understanding how the result of OpenEval can be improved when more time is given for training. Figure 2 shows the results of OpenEval where the x-axis represents the number of iterations that are used for training. These experiments are done when OpenEval uses only one iteration for evaluation. The result shows that OpenEval achieves a high F1 value even when it uses a few iterations for the training.

Figure 3 shows the F1-score of OpenEval when different number of iterations is used for evaluation (OpenEval is trained by using 100 iterations). Interestingly, although the number of training iteration is fixed for all different data points in the figure, the result shows that OpenEval is able to improve its accuracy as more time is given for evaluation. OpenEval uses the trained classifier to measure the effectiveness of each keyword and iteratively finds the set of keywords that are most relevant to the input query. At iteration one, OpenEval achieves F1 score of 74% while the F1 score is increased to 80% at iteration 5. It is worth mentioning that each iteration of predicate evaluator only takes about 1.2 second to be completed.

Related Work

Our work in this paper leverages and extends some of the ideas explored by previous work such as use of co-occurrence statistics in information extraction, machine reading, generic relation extraction and classification. In this section, we briefly describe related research from each of these areas.

Use of co-occurrence statistics: Co-occurrence statistics computed from a collection of documents, such as the Web corpus, has been widely used for information extraction. For example, Pointwise Mutual Information (PMI) is an unsupervised technique that uses search engine hit counts to measure the semantic similarity of pairs of words (Turney 2001). Other researchers have also used PMI to validate information extraction (Soderl et al. July 2004), to validate a candidate answer in a question answering system (Magnini et al. 2002), or to extract a relation instance (Cimiano and Staab 2004; Nakov and Hearst 2005). The performance of all of the above techniques depends on the accuracy of the search engine hit counts, which is shown to be inaccurate in estimating the popularity of input queries (Uyar 2009). Our approach does not rely on the search hit count; rather it uses the content of the web pages that are returned by the search engines for the evaluation.

Machine reading: The main goal of machine reading is to automatically extract knowledge from an unstructured text corpus (e.g., Web) and represent the knowledge in a structured format. KnowItAll (Etzioni et al. 2004; Downey, Etzioni, and Soderland 2010) is one of the first web-scaled machine reading systems that uses generic syntactic patterns to extract relation instances. TextRuuner (Etzioni et al. 2008), ReVerb (Etzioni et al. 2011), and OLLIE (Mausam et al. 2012) are examples of other IE techniques which have shown significant improvements compared to KnowItAll system in both precision and recall. Similar to these works, NELL (Carlson et al. 2010) is a semi-supervised learning technique that extracts structured information from the unstructured web pages using the initial ontology. Weakly supervised techniques are also used for large-scale IE techniques (Hoffmann, Zhang, and Weld 2010; Mintz et al. 2009), where they use an existing ontology to generate a set of training data. Preemptive IE (Shinyama and Sekine 2006) and OnDemand IE (Sekine 2006) are two examples of IE techniques that avoid relation specific extractors and do not require any input ontology, but they rely on document and entity clustering. These dependencies make them too costly to be used in anytime and online applications.

Generic relation extraction and classification: There has been much research in the community of information extraction focusing on supervised techniques to learn relation extractors (Califf and Mooney 2003; Ciravegna 2001). These techniques rely on strong assumptions, such as a large number of manually annotated examples, which make them unable to scale up to hundreds of relations. To deal with this weakness, others have developed unsupervised (Poon et al. 2010; Carlson et al. 2010), weakly-supervised (Zhang 2004), or bootstrap relation extraction/classification techniques (Agichtein and Gravano 2000; Pantel and Pennacchiotti 2006; Freedman et al. 2010) which start from an initial set of seed examples and automatically generate a set of surface patterns (or a set of rules) that are later used to identify new relation instances. However, most of these works have been focused on precision or are dependent on a set of handwritten patterns. Hence the applicability of their techniques is limited for any generic relations. Ontology-driven relation extraction techniques use predefined ontologies (Cimiano, Ladwig, and Staab 2005; Cimiano and Volker 2005; McDowell and Cafarella 2006), which limits their scalability.

There are several main differences between OpenEval and the above IE systems. Most of the current IE techniques are designed for offline batch processing (usually require a long time for training) and to ensure high precision. OpenEval only requires a few minutes to be trained for each predicate and is designed to achieve higher recall (at a small cost of sacrificing precision), making it applicable to online and anytime applications. Second, unlike most of the current IE techniques that require to have a full hierarchy of relationships between predicates as an input ontology, OpenEval only takes mutually-exclusive relationship between predicates. Third, OpenEval is designed to evaluate correctness of a predicate instance, which is generally considered as a more specific and possibly an easier task in IE compared to extracting instances of predicates.

Conclusion

This paper introduced OpenEval, a novel online information validation approach that automatically evaluates the correctness of a predicate instance using the Web. We described two main components of OpenEval in detail: *learning* and *evaluation*, and experimentally showed that OpenEval massively outperforms the related techniques such as PMI and weakly-supervised relation classification technique (Zhang 2004). We presented an online algorithm that explores/exploits information on the Web by extracting a set of context-based instances from the content of Web pages that are returned by the search engine. We showed that our online algorithm is able to improve the accuracy of its prediction as more time is given for the evaluation. OpenEval is a general online technique, requires minimum supervision (in terms of seed examples and input ontology), and improves its accuracy as more time is given for evaluation and training.

Acknowledgments

The authors would like to thank Tom Mitchell, Craig Knoblock, and Thomas Kollar for their valuable discussions and suggestions.

References

Agichtein, E., and Gravano, L. 2000. Snowball: extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, 85–94.

Auer, P. 2003. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.* 3:397–422.

Blum, A. L., and Langley, P. 1997. Selection of relevant features and examples in machine learning. *Artif. Intell.* 97(1-2):245–271.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08*, 1247–1250.

Califf, M. E., and Mooney, R. J. 2003. Bottom-up relational learning of pattern matching rules for information extraction. *J. Mach. Learn. Res.* 4:177–210.

Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Jr., E. R. H.; and Mitchell, T. M. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Cimiano, P., and Staab, S. 2004. Learning by googling. *SIGKDD Explor. Newsl.* 6(2):24–33.

Cimiano, P., and Volker, J. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *RANLP*, 166–172.

Cimiano, P.; Ladwig, G.; and Staab, S. 2005. Gimme' the context: context-driven automatic semantic annotation with c-pankow. In *WWW'05*, 332–341. New York, NY, USA: ACM.

Ciravegna, F. 2001. Adaptive information extraction from text by rule induction and generalisation. In *IJCAI'01*, 1251–1256. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Clemen, R. T. 1989. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting* 5(4):559–583.

Dietterich, T. G. 1998. Machine-learning research: Four current directions. *The AI Magazine* 18(4):97–136.

Downey, D.; Etzioni, O.; and Soderland, S. 2010. Analysis of a probabilistic model of redundancy in unsupervised information extraction. *Artif. Intell.* 174(11):726–748.

Etzioni, O.; Cafarella, M.; Downey, D.; Kok, S.; Popescu, A.-M.; Shaked, T.; Soderland, S.; Weld, D. S.; and Yates, A. 2004. Web-scale information extraction in knowitall: (preliminary results). In *WWW '04*, 100–110. New York, NY, USA: ACM.

Etzioni, O.; Banko, M.; Soderland, S.; and Weld, D. S. 2008. Open information extraction from the web. *Commun. ACM* 51(12):68–74.

Etzioni, O.; Fader, A.; Christensen, J.; Soderland, S.; and Mausam. 2011. Open information extraction: The second generation. In Walsh, T., ed., *IJCAI*, 3–10. IJCAI/AAAI.

Fader, A.; Soderland, S.; and Etzioni, O. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, 1535–1545. Stroudsburg, PA, USA: Association for Computational Linguistics.

Fang, Y., and Chang, K. C.-C. 2011. Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality. In *WSDM '11*, 825–834. New York, NY, USA: ACM.

Freedman, M.; Loper, E.; Boschee, E.; and Weischedel, R. 2010. Empirical studies in learning to read. In *Proceedings of the NAACL HLT 2010*, FAM-LbR '10, 61–69.

Gittins, J. C., and Jones, D. M. 1974. A Dynamic Allocation Index for the Sequential Design of Experiments. In Gani, J., ed., *Progress in Statistics*. Amsterdam, NL: North-Holland. 241–266.

Hoffmann, R.; Zhang, C.; and Weld, D. S. 2010. Learning 5000 relational extractors. In *ACL '10*, 286–295. Stroudsburg, PA, USA: Association for Computational Linguistics.

Kollar, T.; Samadi, M.; and Veloso, M. 2012. Enabling robots to find and fetch objects by querying the web. In *AAMAS '12*, 1217–1218.

Magnini, B.; Negri, M.; Prevete, R.; and Tanev, H. 2002. Is it the right answer? exploiting web redundancy for answer validation. In *ACL-40*, 425–432.

Mausam; Schmitz, M.; Soderland, S.; Bart, R.; and Etzioni, O. 2012. Open language learning for information extraction. In *EMNLP-CoNLL*, 523–534. ACL.

McDowell, L. K., and Cafarella, M. 2006. Ontology-driven information extraction with ontosyphon. In *ISWC'06*, 428–444. Berlin, Heidelberg: Springer-Verlag.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *ACL '09*, 1003–1011.

Mladenić, D.; Brank, J.; Grobelnik, M.; and Milic-Frayling, N. 2004. Feature selection using linear classifier weights: interaction with classification models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, 234–241. New York, NY, USA: ACM.

Nakov, P., and Hearst, M. 2005. Search engine statistics beyond the n-gram: application to noun compound bracketing. In *CONLL '05*, 17–24. Stroudsburg, PA, USA: Association for Computational Linguistics.

Pantel, P., and Pennacchiotti, M. 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *ACL-44*, 113–120.

Poon, H.; Christensen, J.; Domingos, P.; Etzioni, O.; Hoffmann, R.; Kiddon, C.; Lin, T.; Ling, X.; Mausam; Ritter, A.; Schoenmackers, S.; Soderland, S.; Weld, D.; Wu, F.; and Zhang, C. 2010. Machine reading at the university of washington. In *Proceedings of the NAACL HLT 2010*, FAM-LbR '10, 87–95.

Samadi, M.; Kollar, T.; and Veloso, M. M. 2012. Using the web to interactively learn to find objects. In Hoffmann, J., and Selman, B., eds., *AAAI*. AAAI Press.

Sekine, S. 2006. On-demand information extraction. In *COLING-ACL '06*, 731–738.

Shinyama, Y., and Sekine, S. 2006. Preemptive information extraction using unrestricted relation discovery. In *HLT-NAACL '06*, 304–311.

Soderl, S.; Etzioni, O.; Shaked, T.; and Weld, D. S. July 2004. The use of web-based statistics to validate information extraction. In *AAAI Workshop on Adaptive Text Extraction and Mining*.

Sutton, R. S. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, 216–224. Morgan Kaufmann.

Turney, P. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl.

Uyar, A. 2009. Investigation of the accuracy of search engine hit counts. *J. Inf. Sci.* 35:469–480.

Zhang, Z. 2004. Weakly-supervised relation classification for information extraction. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, 581–588. New York, NY, USA: ACM.