Project Report
CS 290D

A Survey of Open Information Extraction Systems

Sina Miran
06/11/14
List of Contents:

1. Introduction to IE, Open IE, and Machine Reading .............. P3
2. TextRunner .............................................................................. P4
3. Reverb ..................................................................................... P6
4. OLLIE ....................................................................................... P7
5. Performance Comparison ....................................................... P9
6. Conclusion ............................................................................... P11
7. References ............................................................................... P11
1. Introduction to IE, Open IE, and Machine Reading

Information Extraction (IE) [1] has always been one of the active research fields in computer science. Basically, the goal of IE systems is to distill semantic relations from natural language text. In traditional IE systems the major goal of the system was to extract some specific relations in a set of hand-labeled documents. For example, in a set of documents on scientific inventions of the past century we might be looking for relations such as “who invented what”. In this case, if we have the sentence “Edison was the inventor of the light bulb”, the output of the IE system should be tuples like (Edison (entry 1), invented (relation), light bulb (entry 2)). Also, since all of the pieces of information in text are not necessarily correct, the IE system should assign a probability to each of its extractions. For instance, if the assigned probability to this tuple is 0.9, it means that the IE system assures the correctness of the information in this extraction by %90.

There are two fundamental problems with these IE systems that prevent us to use these same systems in information extraction tasks nowadays. First, we prefer to extract all of the existing relations in a set of documents automatically rather than looking for pre-specified relations. As mentioned earlier, traditional IE systems look for specific and pre-defined relations. In other words, instead of performing a supervised learning on the possible relations, we want our system to extract all of the relations in an unsupervised manner. Second, traditional IE systems should have hand-labeled documents which are on a specific topic. Hand labeling documents is somehow impractical nowadays because of the huge number of documents present, for example all of the webpages. Therefore, there has been a need towards the next generation of IE systems so that they can scale to the World Wide Web. Table 1 summarizes the differences of traditional IE systems and the new information extraction systems which are called Open IE [1].

<table>
<thead>
<tr>
<th></th>
<th>Traditional IE</th>
<th>Open IE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>corpus + hand-labeled data</td>
<td>corpus</td>
</tr>
<tr>
<td><strong>Relations</strong></td>
<td>specified in advance</td>
<td>discovered automatically</td>
</tr>
<tr>
<td><strong>Extractor</strong></td>
<td>relation specific</td>
<td>relation independent</td>
</tr>
</tbody>
</table>

Table 1: IE vs. Open-IE
Another task we are interested in because of the nature of current information processing tasks, is to do some simple or complex inference over the extracted information (extracted tuples). Machine Reading [1] is defined as the combination of Open-Information Extraction (Open-IE) and doing an inference over the original tuples. As mentioned, this inference does not have to be very complex for the first generation of machine reading systems.

With this introduction, we can come up with a chain in information processing which also shows the future research direction on this subject. As shown in Figure 1, we have some raw information like all the information embedded in World Wide Web. On the top of this raw information, there are powerful search engines like Google that look for specific key words using a knowledge graph. After Google, our goal is to build machine reading systems that can actually answer the questions of the users by doing some inference to come up with several statistics and features if the question is on a specific product for instance.

As my project, I have chosen to do a survey over three Open-IE systems that have fundamental differences and look at this information extraction task from different perspectives namely TextRunner, Reverb, and OLLIE.

![Figure 1: Chain of information processing](image)


TextRunner is one of the first Open-IE systems developed by the University of Washington researchers. The architecture of the system is displayed below in Figure 2. The goal of this system is to first extract some tuples from the data which are in the general form of \((e_i, r_{ij}, e_j)\). \(e_i\) and \(e_j\) are two noun phrases, and \(r_{ij}\) is the relationship between these two. After that, the system does some inference on these tuples to assign a correctness (likelihood) probability with each tuple. One important assumption in this model is that we have a lot of redundant information, so that
we can use this redundancy and co-occurrence of the words to determine the correctness of a tuple. Next, I explain the various components of TextRunner.

**Extractor:** The role of the extractor is to find the raw tuples from the data. Extractor uses various learning methods like conditional random fields to extract the tuples. In this task, the extractor also uses parsers and some heuristics to find the main noun phrases in a sentence and the relationship between them.

**Assessor:** The assessor does some inference over the extracted tuples. For instance, if two entries (noun phrases) \( X \) and \( Y \) have similar tuples, we can say that with a high probability \( X \) and \( Y \) are the same. As an example of this kind of inference, TextRunner may find that “A. Einstein” is the same as “Albert Einstein” and these are not the same as “Einstein Bros.” which is a corporation. Also, if two relations \( R_1 \) and \( R_2 \) share the same tuples (their entries are similar), again we can say that with a high probability these two relations mean the same thing. The next and the more critical part of the inference is how to assign a likelihood probability with a certain tuple. Basically, in task we are looking for the answer to this general problem:

![Figure 2: Architecture of TextRunner](image)
“If an extraction \( x \) appears \( k \) times in a set of \( n \) distinct sentences each suggesting that \( x \) belongs to \( C \), what is the probability that \( x \in C \)?”, where \( C \) is a class such as “cities” or a relation such as “mayor of”.

According to [6], this probability is given by:

\[
P(x \in C | x \text{ appears } k \text{ times in } n \text{ draws}) = \frac{\sum_{r \in \text{num}(C)} \left( \frac{r}{S} \right)^k \left( 1 - \frac{r}{S} \right)^{n-k}}{\sum_{r' \in \text{num}(CUE)} \left( \frac{r'}{S} \right)^k \left( 1 - \frac{r'}{S} \right)^{n-k}}
\]

One intuitive fact behind this probability is that if for fixed \( n \) we increase the number of observations \( k \), then this probability becomes larger, and if for fixed \( k \) we increase the total number of distinct sentences \( n \), the probability becomes smaller. Also note that we are only using the number of “distinct” sentences in our argument. The reason for this is that there is a lot of copying over the web, and we don’t want our probability estimate to be influenced by this fact.

3. ReVerb [7] [8]

TextRunner was among the first Open-IE systems, and the flaws in the extracted tuples by TextRunner motivated the next Open-IE system developed by UW researchers namely ReVerb. Table 2 displays some common problems with the extracted tuples of TextRunner. As we see, broadly speaking, the problem with TextRunner is that many times it does not extract the full relation between two noun phrases, and only extracts a portion of the relation which is ambiguous. For instance, where it should extract the relation “is an album by”, it only extracts “is” as the relation.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Incoherent Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The guide contains dead links and omits sites.</td>
<td>contains omits</td>
</tr>
<tr>
<td>The Mark 14 was central to the torpedo scandal of the fleet.</td>
<td>was central torpedo</td>
</tr>
<tr>
<td>They recalled that Nungesser began his career as a precinct leader.</td>
<td>recalled began</td>
</tr>
</tbody>
</table>

Table 2: Common problems with TextRunner
Based on these flaws, ReVerb, which stands for Relation from Verbs, was developed. The idea behind reverb is that since we know the grammatical format of common relations, instead of learning the format of the relations in an unsupervised manner, we enforce this common grammatical structure on the relations. In ReVerb, three grammatical structures are considered for relations, and, in each sentence, the longest phrase matching one of the three formats below is considered as the relation in that sentence.

1) V
2) VP
3) VPW

\[ V = \text{verb}, \ W = (\text{noun} | \text{adj} | \text{adv} | \text{pron} | \text{det}), \ P = (\text{prep} | \text{particle} | \text{inf. marker}) \]

Generally, ReVerb assumes that all relations are mediated by verbs and tries to find such relations. The other condition imposed on relations in ReVerb is that each valid relation must at least get repeated \( k \) times in the whole corpus (\( k \sim 20 \)). This condition is for not acknowledging relations like “is offering only modest greenhouse gas reduction at” as a correct relation since relations such as this are valid based on the three grammatical structures.

As a recap, we can say the basic idea behind ReVerb is if we know the common structure of the relations, then instead of using machine learning methods to capture the format of the relations, we apply this common format to our relations manually. Since there are a lot of redundant information on the web, in using ReVerb, we hope that any tuple and relation not mediated by verbs will have an equivalent verb-mediated relation somewhere in the web which can be extracted using ReVerb.

4. OLLIE [9]

As mentioned earlier, ReVerb only extracts the relations mediated by verbs. However, our ultimate goal is that given a sentence find all of the available relations whether they are mediated by verbs or not. Table 3 shows some defects of ReVerb that prevents it from extracting all of the available information in a sentence, and these defects motivated the development of OLLIE, which stands for Open Language Learning for IE. As we see, in 1, 2, and 3, the relations extracted by OLLIE won’t be extracted using ReVerb. Another problem with ReVerb is that somehow ignores the context of the relation. Therefore, sometimes, it might give false and incomplete relations. OLLIE has solved this problem by adding new elements to the tuples namely AttributedTo and ClauseModifier as seen in 4 and 5.
Table 3: Problems with ReVerb (O:OLLIE, R:ReVerb)

Figure 3 shows the general architecture of OLLIE. The basic idea behind each of the building blocks of OLLIE will be described next.

**ReVerb:** OLLIE assumes that all the possible relations have equivalent verb-mediated relations which can be found over the web. Based on this assumption, OLLIE tries to learn the patterns of relations using their equivalent relation found by ReVerb. Thus, in the first step, ReVerb is applied on the raw data (WWW) and among the extracted tuples the most reliable ones are chosen.

**Bootstrapper:** For each tuple found using ReVerb, the bootstrapper looks for all the sentences in raw data having the same words as in that extracted tuple. The assumption is that
all these sentences contain equivalent relations to the original verb-mediated tuple extracted. Therefore, the next step would be to learn the new patterns of relations using these sentences for each of the ReVerb’s extracted tuples. The extracted sentences for each of the ReVerb’s tuples are the training data marked in Figure 3.

**Open Pattern Learning:** Using the dependency parsers, for instance [10], this block learns and specifies new grammatical formats for relations using the training data which are called pattern templates in Figure 3.

**Pattern Matching:** Now that we have the learned pattern templates, for each sentence, we use the output of a dependency parser on the sentence to match it with one or multiple learned pattern templates.

**Context Analysis:** Again, using the output of a dependency parser on the sentence, OLLIE tries to change some of the incomplete tuples such that the information within them exactly reflects what was meant by the sentence. For instance, if there is a *if clause* in the sentence the context analysis block adds it as a ClauseModifier to the tuple. This block exactly strives to solve the problem with ReVerb that was shown in parts 4 and 5 of Table 3.

5. **Performance Comparison**

In this section, I give a brief comparison of these three systems.

1) **ReVerb vs. TextRunner:**

Two important criteria that can be used in assessing the performance of Open-IE systems are precision and recall. In other words, we are interested in the ratio of the tuples extracted by the system to the total number of tuples available in the documents and the ratio of the correct tuples extracted by the system to the total number of extracted tuples. As introduced earlier, for each tuple there is a probability assigned to it. If we put a threshold on these probabilities to denote the ones with the probability greater than the threshold as the final extracted tuples and vary this threshold, we can plot a precision vs. recall curve for each of the algorithms. In other words, for each value of the threshold, we can calculate the corresponding precision and recall using the final extracted topics after applying the threshold. Obviously, the higher a precision-recall curve is, the better the performance of the corresponding algorithm will be.

The dataset used in Figure 4 consists of 500 sentences sampled randomly from the web using Yahoo!’s random link services. Two human judges evaluated the extractions as correct or false and determined all the possible tuples that can be extracted using the sentences. As we said, ReVerb has two constraints. ReVerb-lex is ReVerb with only the first constraint on the grammatical structure of the sentences. As we see, TextRunner is outperformed by ReVerb, and second constraint on ReVerb (at least $k$ repetitions for each relation) improves its performance.
2) OLLIE vs. ReVerb:

The dataset used for Figure 5 contains 300 random sentences from three sources such as Wikipedia, and the 2000-most reliable extractions were considered for each of the algorithms. Again, extractions were annotated manually as correct or false by human judges. WOE is a Wikipedia based parser which was not examined in this survey. Yield is defined as the number of correct extractions for each value of precision as the total number of extractions is fixed at 2000 in this experiment. As we see, for most values of precision, OLLIE is superior to ReVerb in terms of the corresponding number of relations found. The reason why ReVerb seems to give higher precision for small number of extractions is that it only considers verb-mediated relations which are most likely correct (high precision for very high confidence tuples) but the problem is that these do not cover all of the possible relation in a sentence!
Another experiment that can be done is to see for a specific relation how many extractions OLLIE and Reverb can find in a set of documents. Table 4 shows the total number of extractions found by OLLIE and Reverb corresponding to a specific verb-mediated relation. The dataset used in this experiment contains 100M sentences from the ClueWeb corpus available at [11]. Since OLLIE is not limited to finding solely the verb-mediated relations and also extracts the equivalent non-verb-mediated relations, we see that it finds a far larger number of tuples compared to ReVerb for a specific relation.

<table>
<thead>
<tr>
<th>Relation</th>
<th>OLLIE</th>
<th>REVERB</th>
<th>incr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>is capital of</td>
<td>8,566</td>
<td>146</td>
<td>59x</td>
</tr>
<tr>
<td>is president of</td>
<td>21,306</td>
<td>1,970</td>
<td>11x</td>
</tr>
<tr>
<td>is professor at</td>
<td>8,334</td>
<td>400</td>
<td>21x</td>
</tr>
<tr>
<td>is scientist of</td>
<td>730</td>
<td>5</td>
<td>146x</td>
</tr>
</tbody>
</table>

Table 4: OLLIE vs. ReVerb

6. Conclusion

In this survey, information extraction and open information extraction were introduced and the motivation behind them were explained. Three different Open-IE systems, namely TextRunner, ReVerb, and OLLIE, which looked at this task from different views were introduced and explained. Finally, a concise comparison of their performance was given to see how each system tries to overcome the difficulties of its predecessor.

7. References


