

# Information Extraction for Ontology Learning

Fabian SUCHANEK

*Max Planck Institute for Informatics, Germany*

**Abstract.** In this chapter, we discuss how ontologies can be constructed by extracting information from Web documents. This is a challenging task, because information extraction is usually a noisy endeavor, whereas ontologies usually require clean and crisp data. This means that the extracted information has to be cleaned, disambiguated, and made logically consistent to some degree. We will discuss three approaches that extract an ontology in this spirit from Wikipedia (DBpedia, YAGO, and KOG). We will also present approaches that aim to extract an ontology from natural language documents or, by extension, from the entire Web (OntoUSP, NELL and SOFIE). We will show that information extraction and ontology construction can enter into a fruitful reinforcement loop, where more extracted information leads to a larger ontology, and a larger ontology helps extracting more information.

**Keywords.** Information Extraction, Reasoning, Consistency, Taxonomy, Disambiguation

## 1. Introduction

### 1.1. Motivation

The World Wide Web provides an enormous source of knowledge. News articles, university home pages, scientific papers, personal blogs, commercial Web sites, catalogs such as the Internet Movie Database, and entire encyclopedias such as Wikipedia are available online. These sites host information on a wealth of topics, from scientific theories to current political events, from digital cameras to endangered wildlife species and from train schedules to cooking recipes. This information is available as digital documents. This means that machines can store, display and index these documents. However, machines do not have a semantic representation of the content of the documents. This means that machines cannot figure out whether a document contains contradictory information. It also means that we cannot ask SPARQL queries against the documents. Thus, there is a gap between the syntactical way in which data is stored on the Web and the semantic way in which information is stored in RDF/OWL ontologies.

Information Extraction (IE) steps in to bridge that gap. IE is the science of extracting structured data from unstructured or semi-structured documents. For example, given a natural language document with the sentence “Elvis Presley was born in Tupelo”, the goal of an IE system could be to extract the triple  $\langle \textit{Elvis}, \textit{bornIn}, \textit{Tupelo} \rangle$ . This fact could then become part of an ontology. If this process could be applied to all the documents of the Web, then the syntactic information of the Web could become semantically

accessible to machines. This chapter will present recent promising work in this direction. The remainder of this section will first discuss IE in general and for ontologies in particular. Then, Section 2 will present approaches that extract ontological information from Wikipedia. Section 3 discusses approaches that target unstructured Web pages, or the entire Web, before Section 4 concludes.

### 1.2. Information Extraction

IE is the process of extracting structured information from one or multiple given source documents. There are literally hundreds of different IE techniques. Some of these techniques have become particularly popular. At first, the document is usually preprocessed. This may include character set detection or conversion, the removal of special characters, DOM tree construction for HTML documents and general cleaning, such as the removal of irrelevant portions of the document. Next, the document is usually segmented into words or tokens (cf. Maynard and Bontcheva [13], this volume). At the level of tokens, we can already apply techniques of Named Entity Recognition. NER will identify the tokens that are dates, numbers and proper names. Some NER techniques can even distinguish location names, organization names and person names. All approaches that we will see in this chapter use basic preprocessing of this kind.

After NER, first methods of fact extraction can be applied. If the document is structured (such as HTML tables, lists, or Web sites that follow a template), we can employ techniques such as wrapper induction [9] or table annotation [12]. These techniques can extract tabular information from Web pages. If the document is unstructured (i.e., if it consists of natural language text), we can apply approaches that seek patterns in the sentences of the text [10]. For example, the pattern “X is a Y” indicates that X could be an instance of the concept Y (such as in the sentence “Elvis Presley is a rock singer”). The patterns can either be predefined or learned from examples. Other techniques rely on a co-occurrence analysis of terms. For example, if *Elvis Presley* always appears together with the words *guitar* and *sing*, then this may indicate that he is a musician.

If the document contains natural language text, it can also be linguistically analyzed, as explained by Maynard and Bontcheva [13] (this volume). This may include part-of-speech tagging and syntactic analysis, such as dependency parsing. Then, linguistic techniques can be applied to deduce the meaning of a sentence and distill the facts it expresses. There are many more approaches, which blend, adapt or use the previously mentioned approaches or which take a new perspective altogether. The reader is referred to [18] for a general survey of techniques.

### 1.3. IE for Ontologies

Traditional IE concentrated on entity extraction, relation extraction, and fact extraction. Given a sentence “Elvis married Priscilla Beaulieu”, an IE system aimed to extract that the relation *married* holds between *Elvis* and *Priscilla Beaulieu*. If this fact is to be used in an ontology, however, it is not sufficient to extract just these three tokens. Rather, three more desiderata have to be fulfilled:

1. **Canonicity:** The entity names have to be disambiguated and mapped to existing entities of the ontology. In the example, the system has to determine that “Elvis” refers to the entity *Elvis Presley* (instead of, say, the singer *Elvis Costello*). The

system may also decide to introduce a new entity if the entity does not yet exist in the knowledge base. The same applies to class names and relation names.

2. **Taxonomic organization:** The extracted entities have to be placed in a taxonomy of classes. In the example, the system has to know that Elvis is a singer and that this implies that he is a person.
3. **Consistency:** The extracted facts have to be logically consistent with the ontology. In the example, the system may have to check that Priscilla was born before Elvis died.

These desiderata put an additional burden on the IE system. At the same time, the ontology can also help the IE process. This can happen on multiple levels. First, the data that is already present in the ontology can serve as seed data for the IE process. For example, assume that the ontology already knows that Einstein was married to Mileva Maric. If the system comes across the sentence “Einstein married Mileva Maric”, the system can guess the meaning of the pattern “X married Y”. This, in turn, will allow it to understand the sentence “Elvis married Priscilla Beaulieu”. The ontology can also help in another way: If the ontology has to be logically consistent, then certain interpretations of the source document can be excluded. In our sample sentence, the word “Elvis” cannot refer to Saint Elvis, because this saint lived in the 5th century and cannot have married Priscilla.

Thus, the more knowledge has already been extracted, the better the system will be able to extract new information. The better it extracts new information, the more knowledge will be extracted. Much like humans, who learn faster when they know more, and know more when they learn faster, IE and ontologies could enter a fruitful cycle of knowledge accumulation. We will look at two classes of ontological knowledge extraction systems that go into this direction. The first class extracts information from Wikipedia. The second class ventures beyond Wikipedia and extracts information from arbitrary documents – or targets even the whole Web.

## 2. IE from Wikipedia

Wikipedia is a multilingual, Web-based encyclopedia. It is written collaboratively by volunteers and is available for free under the terms of the Creative Commons Attribution-ShareAlike License<sup>1</sup>. As of February 2011, the English Wikipedia contained more than 3.5 million articles<sup>2</sup>. Wikipedia suggests itself for ontological information for multiple reasons. First, Wikipedia articles follow a number of conventions concerning the formatting and the content. This makes Wikipedia much more homogeneous than a set of random Web pages and facilitates IE. Second, Wikipedia is constantly updated and improved by thousands of volunteers, so that it is widely regarded as credible, correct and authoritative. Wikipedia also covers many different aspects of knowledge, from science to literature or politics, which makes it useful as a source for a general-purpose ontology. Lastly, Wikipedia is a highly structured resource, which contains not just natural language texts, but also standardized tabular information, a category system and meta-

---

<sup>1</sup><http://creativecommons.org/>


<sup>2</sup><http://en.wikipedia.org/wiki/Wikipedia:Statistics>

## Elvis Presley

---

**Elvis Aaron Presley** ([January 8, 1935](#) – [August 16, 1977](#)), middle name sometimes written Aron) was an [American singer](#), [musician](#) and [actor](#).  
*etc.*

[Categories](#): [1935 births](#) | [1977 deaths](#) | [American rock singers](#) | [Rock and roll](#) | [People with diabetes](#)  
*etc.*



Born: January 8, 1935  
Occupations: [singer](#), [actor](#)  
*etc.*

**Figure 1.** A Wikipedia Article

information such as inter-article links and user information. This allows structured IE techniques to work, which often have a higher precision than unstructured techniques.

For these reasons, it is not surprising that several ontological IE approaches have targeted Wikipedia. This section will present three of them. For this purpose, let us first discuss the technical markup of Wikipedia.

### 2.1. Wikipedia

Each Wikipedia article is a single Web page and usually describes a single topic or entity, the *article entity*. Figure 1 shows an excerpt of the article about Elvis Presley. As an online encyclopedia, Wikipedia has several characteristics: First, each article is highly inter-linked to other articles. In the example, a click on the word *singer* leads to the Wikipedia article about singers. Each Wikipedia article has an unstructured, natural language part (on the left hand side in Figure 1). In addition to that, some Wikipedia articles also have an *infobox* (pictured on the right hand side). An infobox is a standardized table with information about the article entity. For example, there is an infobox template for people, which contains the birth date, the profession, and the nationality. Other widely used infobox templates exist for cities, music bands, companies etc. Each row in the infobox contains an attribute and a value. For example, our infobox contains the attribute *Born* with the value *January 8, 1935*. An infobox attribute may also have multiple values, as exemplified by the *Occupations* attribute. The majority of Wikipedia articles have been manually assigned to one or multiple *categories*. Our sample article is in the categories *American rock singers*, *1935 births*, and 34 more. Figure 1 shows an excerpt at the bottom.

Wikipedia is rendered as HTML pages, but is written in a special markup language, the *Wiki markup language*. Figure 2 shows an excerpt of the article on Elvis Presley in this language. The XML dump of Wikipedia (as of 2010) is approximately 27 Gigabytes large and stores the articles in the Wiki markup language.

We will now look at three systems that exploit this structured nature of Wikipedia: DBpedia, YAGO and KOG. Since these systems build an ontology, they can use automated reasoning to check and control the knowledge they are extracting. While DBpedia uses fewer reasoning, YAGO uses simple deductive reasoning and KOG uses the more sophisticated Markov logic [17].

```

{{Infobox musical artist
| Name = Elvis Presley
| Img = Elvis Presley 1970.jpg
| Born = {{birth date|1935|1|8|}}
| Occupation = [[singer]], [[actor]]
etc.
}}

'''Elvis Aaron Presley''' ([[January 8]], [[1935]] - [[August
16]], [[1977]]), middle name sometimes written '''Aron''', was
an [[United States|American]] [[singer]], [[musician]] and
[[actor]]. etc.

[[Category:1935 births]] [[Category:1977 deaths]] etc.

```

**Figure 2.** The Wikipedia Markup Language

## 2.2. DBpedia

DBpedia [2] is a system that extracts an RDF ontology from Wikipedia<sup>3</sup>. The project is a community effort, coordinated by the University of Leipzig, the Free University of Berlin and the OpenLink Software company. DBpedia was among the first projects to harvest Wikipedia at a large scale for ontology construction.

### 2.2.1. Fact Extraction

The DBpedia system takes as input a dump of Wikipedia. The system creates one resource for each article in Wikipedia. For example, the Wikipedia page named “Elvis Presley” gives rise to the DBpedia resource [http://dbpedia.org/resource/Elvis\\_Presley](http://dbpedia.org/resource/Elvis_Presley). The DBpedia extractors create basic facts about the resource by exploiting, among other things, the title of the article (which becomes the *rdfs:label* of the resource), the image of the article (which creates a triple linking the resource and the image URL) and the external links of the article (which each generate one triple). DBpedia applies this process to all language pages of Wikipedia, so that each resource in DBpedia has labels and abstracts in multiple languages (if Wikipedia has them).

DBpedia uses two approaches to harvest the infoboxes of Wikipedia. The first approach is *recall-oriented*. This means that it tries to harvest as much information as possible, even if this information may not be correctly extracted or contradictory. In this approach, every value of the infobox gives rise to one triple. The subject of the triple is the article resource. The relation of the triple is the infobox attribute, prefixed by <http://dbpedia.org/property/>. The object of the triple is the value in the infobox. For example, the line “Occupation = [[singer]]” in Elvis’ infobox translates into the triple

```

<http://dbpedia.org/resource/Elvis\_Presley,
http://dbpedia.org/property/occupation,
http://dbpedia.org/resource/singer>

```

<sup>3</sup><http://dbpedia.org>

The extractors can detect and handle lists, dates, numbers, and units as data types. This extraction technique covers all infoboxes and all infobox attributes. Unfortunately, Wikipedia sometimes uses different identifiers for the same relation. For example, some Wikipedia templates contain the attribute *occupation* while others contain *profession*. Therefore, the extraction algorithm creates many synonymous relations without an indication that they are synonymous.

Hence DBpedia also pursues a second approach to infobox harvesting [4]. This approach is *precision-oriented*, meaning that it extracts fewer information, but with higher quality. For this purpose, the DBpedia community has begun to map Wikipedia templates into an ontology. This ontology was created by manually arranging the 350 most commonly used infobox templates within the English edition of Wikipedia into a subsumption hierarchy. This hierarchy consists of 170 classes. The community then mapped the 2350 attributes of these templates to 720 manually defined ontology properties. The property mappings include fine-grained rules on how to parse infobox values and define target data types, which help the parsers to process attribute values. For instance, if a mapping defines the target data type to be a list of links, the parser will ignore additional text that might be present in the attribute values. This extraction provides cleaner data, but works only on the templates that have been manually identified. This applies to roughly half of the resources already.

### 2.2.2. Taxonomies

DBpedia arranges its resources into 4 taxonomies. The first taxonomy is derived from the **Wikipedia Category** system. Every category of Wikipedia becomes a class, structured by *skos:broader* and *skos:narrower* relationships. The main advantage of the category system is that it is collaboratively extended and kept up-to-date by thousands of Wikipedia editors. A disadvantage is that categories do not form a proper topical hierarchy, as there are cycles in the category system and as categories often only represent a rather loose relatedness between articles. The second taxonomy is imported from the **YAGO** ontology. This taxonomy is described in detail in Section 2.3. While YAGO achieves a high accuracy in general, there are a few errors and omissions due to its automatic generation. A third taxonomy has been taken from the Upper Mapping and Binding Exchange Layer **UMBEL**. The UMBEL taxonomy was derived from OpenCyc. The fourth taxonomy is being developed manually by the **DBpedia community** from the most commonly used infobox templates within the English edition of Wikipedia. It consists of 170 classes that form a shallow subsumption hierarchy. It includes 720 properties with domain and range definitions. As all taxonomies use Wikipedia identifiers for the instances, they can all co-exist in parallel in DBpedia. See the PARIS project [19] for a comparison of the class and relation hierarchies of YAGO and DBpedia<sup>4</sup>.

### 2.2.3. Summary

As of February 2011, DBpedia contains 3.5 million resources and 670 million triples. The DBpedia data can be downloaded from the Web, but it can also be accessed through a SPARQL interface and through the Linked Data protocol [3]. This has made DBpedia a hub in the world of linked data. One of the main technical contributions of DBpedia is the systematic harvesting of infoboxes for fact extraction. This techniques give DBpedia

---

<sup>4</sup><http://webdam.inria.fr/paris>

large coverage of practically all structured facts of Wikipedia. The entities of DBpedia are canonic, because DBpedia is based on Wikipedia, which contains very few duplicate entities. The community-based definition of extraction patterns ensures that more and more of the properties of DBpedia are also canonic. Through its 4 class hierarchies, DBpedia also gives an extensive taxonomic structure to its data.

DBpedia does not use reasoning to check its data for consistency. The YAGO project makes first steps in this direction.

### 2.3. YAGO

YAGO[20] is an ontology<sup>5</sup> that has been automatically constructed from Wikipedia, WordNet [8], the Universal WordNet [7] and Geonames<sup>6</sup>. The project is being developed by the Max Planck-Institute for Informatics in Germany. YAGO was one of the first projects that used the extracted knowledge for consistency checks on other extracted knowledge.

#### 2.3.1. Fact Extraction

Much like DBpedia, YAGO starts out from Wikipedia and makes every article a resource. For every article, YAGO also collects the categories of the article. It determines which of the categories identify a class to which the article entity belongs. For example, the category *American rock singers* identifies a class for the article entity *Elvis Presley*, while the category *Rock music* does not. YAGO creates a class for each of these category names and makes the article entity an instance of these classes. Then, YAGO links the classes to the corresponding classes in the WordNet taxonomy [8]. The class *American rock singers*, e.g., becomes a subclass of the WordNet class *singer*. YAGO harvests the infoboxes by help of a manual mapping from infobox attributes to properties [21]. YAGO also contains the Universal WordNet UWN [7]. UWN was derived from WordNet and translates its class names into up to 200 different languages. The YAGO extractors add in the classes and instances from Geonames, a dataset of geographical entities. Careful matching algorithms make sure that each Geonames class finds its place in the existing taxonomy and that each Geonames instance is mapped to the matching entity from Wikipedia, if it already exists. Thereby, YAGO is virtually free of duplicates and achieves canonicity of entities, classes and relations.

#### 2.3.2. Fact Checking

In YAGO, every entity is assigned to at least one class. If YAGO cannot find a class for an entity, the entity is purged. This applies also to literals. YAGO contains a manually designed taxonomy for literals (the class *ThreeLetterLanguageCode*, e.g., is a subclass of *String*). Each literal class comes with a regular expression that identifies lexical forms of the data type. Furthermore, the relationships in YAGO are defined manually and have ranges and domains. This forces the extracted data into a rather rigid framework of classes and constraints. Within this framework, the YAGO extractors can perform a limited type of reasoning while extracting the facts:

---

<sup>5</sup><http://yago-knowledge.org>

<sup>6</sup><http://geonames.org>

1. **Functional Dependencies:** If a relation is known to be a function, only one object will be accepted for each subject
2. **Reductive Type Checking:** If a value extracted from an infobox does not meet the domain and range conditions of the relation, it is rejected. The type checking applies to resources (which have a type given by the taxonomy) as well as to literals (which can be type checked by the regular expression from the data type taxonomy).
3. **Type Coherence Checking:** At the top-level, the taxonomy of YAGO is partitioned into 5 different branches (locations, artifacts, people, other physical entities, and abstract entities). If an entity is an instance in multiple branches, a voting procedure is used to determine the only branch that will be accepted for that entity.

These mechanisms ensure that potentially erroneous triples are not accepted into YAGO.

### 2.3.3. Fact Deduction

The YAGO extractors use reasoning not just to eliminate triples, but also to deduce new triples. YAGO implements a deduction mechanism for simple Horn rules. These rules are manually defined. The following rule, e.g., deduces that if an entity has an academic supervisor, then that entity must be a person:

$$\frac{\langle X, :hasAcademicSupervisor, Y \rangle}{\langle X, rdf:type, :person \rangle}$$

This mechanism, *Inductive Type Checking*, works for certain relations that identify the type of its argument unambiguously. Other rules deduce temporal and spatial information for facts. For example, YAGO knows that the relation *wasBornOnDate* indicates the temporal starting point of an entity. YAGO also knows that the relation *wasBornInPlace* happened at the time of the birth of the entity. Therefore, YAGO deduces that the time of the *wasBornInPlace* fact must be the date that comes with the *wasBornOnDate* fact. Horn rules are used to propagate time and space information from one fact to the other. As a result, YAGO can attach time intervals and geographic locations to many of its entities and facts, thus giving the data a temporal and a spatial dimension [11].

### 2.3.4. Summary

YAGO contains 80 million triples about 10 million resources. This includes temporal and spatial information for both facts and entities. YAGO has been evaluated manually and shown to have an accuracy of 95% (with respect to Wikipedia as the ground truth). The main technical contribution of the project is the use of a basic reasoning framework to check extracted facts for consistency and to derive new facts. Through the manual definitions of properties and the matching algorithms, YAGO achieves canonicity of entities, classes, and relations. Through the fact checking mechanisms, YAGO achieves a certain consistency of the extracted data. By mapping Wikipedia categories to the classes of WordNet, YAGO gives its data a taxonomic backbone.

YAGO does not exploit that infobox templates can also yield class information. We are going to discuss next a project that uses the infobox templates to construct a taxonomy.



## 2.4. KOG

The Kylin Ontology Generator (KOG) [25] is a system that builds an ontology from the infoboxes of Wikipedia and combines it with WordNet. KOG is part of the Machine Reading project at the University of Washington. KOG was the first system to treat Wikipedia infoboxes as classes and to infer their attributes and subsumption relations.

### 2.4.1. Class Extraction

KOG starts out from the dump of Wikipedia. It treats each infobox template as a class. The attributes of the template are considered attributes of the class. KOG cleans the infobox data in 4 steps:

1. **Recognizing Duplicate Infobox Types:** Sometimes, the same class of things (e.g., US counties) are described by two types of infoboxes (e.g., *uscounty* and *us\_county*). KOG uses the Wikipedia redirect pages, the Wikipedia editing history and name similarity heuristics to detect and merge equivalent infobox types.
2. **Assigning Meaningful Names:** Some infobox types have obscure names such as *abl*. KOG uses the Wikipedia redirect links, the Wikipedia categories of the respective articles and Google spell checking to find more meaningful names for the types (such as *AmericanBaseballLeague* instead of *abl*).
3. **Inferring Attribute Ranges:** KOG collects all values of a given attribute (e.g., all people who occur as values of the attribute *directedBy*). For each value, it determines the class of which the value is an instance. KOG uses the YAGO hierarchy and the DBpedia hierarchy [2] for this purpose. It chooses the superclass of the most frequent classes as the range for the attribute. For example, if most people that appear in a *directedBy* attribute are instances of the class *AmericanMovieDirectors* or the class *ItalianMovieDirectors*, KOG chooses the superclass *movieDirectors*.
4. **Attribute Mapping:** Some attributes are similar, even if they appear in different infobox types. For example, both the infoboxes for actors and the infoboxes for scientists have an attribute *spouse*. KOG uses string matching techniques, information from the taxonomy (see below) and data from the edit history of pages to identify such attributes.

### 2.4.2. Taxonomy Construction

Next, KOG builds a taxonomy in 2 phases: First, KOG establishes which infobox template must be a subclass of which other infobox template. For example, KOG figures out that the template *EnglishPublicSchool* must be a subclass of *PublicSchool*. KOG uses name matching, pattern matching and information from the Wikipedia categories, the edit histories, and WordNet [8] for this purpose. Next, KOG connects the infobox classes to WordNet. For this purpose, KOG uses the mapping that exists already in YAGO [20] and the mapping that exists already in DBpedia [2], and trains a classifier to detect subsumption between a Wikipedia entity and a WordNet entity. KOG uses Markov Logic Networks [17] to take into account rules such as the transitivity of *subClassOf*. This produces a subsumption hierarchy of classes. Each class is canonic and has canonic attributes. The hierarchy is connected to the WordNet hierarchy.

### 2.4.3. Summary

An evaluation of KOG yields precision rates around and beyond 90%. As of the time of this writing, the KOG ontology is not available online. One of the main technical contributions of the KOG system is the exploitation of infoboxes of Wikipedia as classes. These classes are anchored in WordNet, so that the KOG ontology possesses a strong taxonomic structure. KOG implements many techniques to make sure its classes, entities and relations are canonic. Unlike DBpedia and YAGO, KOG achieves this canonicity without manual rules. KOG also implements reasoning techniques to ensure the consistency of the extracted data.

KOG is highly tailored to the internal structure of Wikipedia. We will next look at systems that go beyond Wikipedia.

## 3. IE from the Web

While Wikipedia contains already much information, it contains only part of the huge amount of data that is available on the Web. This is why several newer approaches have embarked to go beyond Wikipedia, and to extract ontological information from the entire Web. The Web is much more heterogeneous than Wikipedia, with different file formats, different languages, different page layouts, and only creeping standardization. Furthermore, the information on the Web exhibits various degrees of credibility. Data may be faulty, incomplete, contradictory, or wrong. In addition, the Web is one of the largest computer-processable resources at all. This makes IE from the Web particularly challenging.

On the other hand, an IE algorithm can also benefit from the size of the Web: The redundancy of the Web means that the same piece of information is likely to appear multiple times in multiple forms. For example, the fact that Elvis won the Grammy Award is likely to appear several pages in several forms. If the algorithm manages to extract this piece of information from one page, it may be able to discover it subsequently on another page. This may allow it to learn a new textual pattern. This pattern, in turn, can then generate new facts. This mutual reinforcement of pattern discovery and fact discovery is a well-known principle in IE [5,1].

Due to the redundancy, the goal is no longer to extract all information from a given single document. Rather, the goal is to extract all information that is expressed in the documents, no matter from which document. This idea heralds a paradigm shift, which has been called *machine reading* [15] or *macro-reading* [6]. Quite a number of approaches have embarked in this direction. Some of the more prominent ones are the Read-the-Web project NELL [6] of Carnegie-Mellon University, the TextRunner/KnowItAll project<sup>7</sup> and the OntoUSP project at the University of Washington, and the PROSPERA/SOFIE project at the Max Planck Institute for Informatics. In this article, we will focus on OntoUSP, NELL, and SOFIE, as they aim to produce ontological output with canonicity, consistency, and a taxonomy in mind.

---

<sup>7</sup><http://www.cs.washington.edu/research/knowitall/>

### 3.1. *OntoUSP*

OntoUSP<sup>8</sup> is a system that can extract information in an unsupervised way from natural language text [16]. OntoUSP is part of the Machine Reading project [15] at the University of Washington. OntoUSP can build up a knowledge base from the input documents and answer questions on it without using any training data.

#### 3.1.1. *Fact Extraction*

The goal of OntoUSP is to build a probabilistic knowledge base from a set of natural language documents. In this knowledge base, both entities and relationships will be represented as clusters of synonymous terms. For example, a protein called *IL-4* will be represented as a cluster of synonymous terms, such as *Protein IL-4*, *the cytokin interleukin-4* or *the IL-4 protein*. The relationship *enhance*, which holds between a protein and a gene, is represented as the cluster of the terms *enhances*, *induces* etc.

OntoUSP assumes that the documents have been parsed by a dependency parser (cf. Maynard and Bontcheva [13], this volume). Thus, OntoUSP takes as input a set of dependency trees. The dependency trees are first represented in a so-called *quasi-logical form* (QLF), i.e., in a first order logic conjunction: every node in the dependency tree becomes a constant; every edge becomes a binary predicate applied to the two node constants and every leaf becomes a unary predicate applied to the leaf node constant. For example, the sentence “IL4 induces CD11b” becomes

$$\textit{induces}(n_1) \wedge \textit{subj}(n_1, n_2) \wedge \textit{ILA}(n_2) \wedge \textit{obj}(n_1, n_3) \wedge \textit{CD11b}(n_3)$$

Parts of such QLFs can be generalized to *lambda forms*, i.e., to sub-formulas where some constants are replaced by variables. In the example, we can extract the lambda form  $\lambda x_2, x_3. \textit{induces}(n_1) \wedge \textit{subj}(n_1, x_2) \wedge \textit{obj}(n_1, x_3)$ . The OntoUSP algorithm groups lambda forms to *lambda form clusters*. A lambda form cluster is a set of semantically interchangeable lambda forms. For example, the cluster for “IL4 induces CD11b” will contain a cluster of synonymous lambda forms for *ILA* (such as *the IL-4 protein*), a cluster of synonymous forms for *induces* and a cluster of synonymous forms for *CD11b*. To build these clusters, OntoUSP uses three operators: MERGE (which merges two clusters to a larger cluster), ABSTRACT (which creates a super-cluster that contains two sub-clusters) and COMPOSE (which combines two forms to a sequence of forms, such as *the* with *protein* to *the protein*). OntoUSP uses a Markov Logic [17] to assign a cost to these operations and a hill-climbing algorithm to perform the clustering.

The result of this clustering is a probabilistic knowledge base, in which synonymous linguistic expressions are grouped together. The clusters are arranged in a taxonomic inclusion hierarchy. For example, OntoUSP can figure out that “induce” and “inhibit” are sub-clusters of the more general “regulate”. OntoUSP can also abstract away the difference between active voice and passive voice. This allows OntoUSP to answer questions on the knowledge base, even if the question uses other terms than the original text.

---

<sup>8</sup>Ontological Unsupervised Semantic Parsing

### 3.1.2. Summary

OntoUSP has been run on a corpus of 2000 biomedical documents and evaluated with 2000 natural language questions. The system achieves a precision of 91%. The main technical contribution of OntoUSP is the unsupervised learning of a hierarchy of synonymous linguistic forms. The knowledge that OntoUSP builds up is *implicit* in the sense that it lacks explicit labels, logic constraints and crisp assignments of entities to classes. Still, OntoUSP exhibits a canonicity in the sense that it groups together synonymous names for one entity. OntoUSP also builds a hierarchical structure of the forms, thus inducing a type of taxonomy. However, this structure is not a class hierarchy in the classical sense, because it lacks explicit labels and crisp membership.

Due its implicit nature, OntoUSP cannot use reasoning to ensure the logical consistency of its knowledge base. We will discuss next a system that makes use of logical constraints to extract knowledge.

## 3.2. NELL

NELL<sup>9</sup> is a research project that aims to extract ontological information from the whole Web [6]. The project is driven by the Carnegie Mellon University. The NELL system aims to gather ontological information continuously on a large scale. It was launched in January 2010 and has been running ever since. The idea is that NELL shall not just constantly accumulate new knowledge, but also become better at it.

### 3.2.1. Fact Extraction

The goal of NELL is to create a set of facts (triples) together with a taxonomy. NELL uses as input a text corpus of several million documents. It also uses an initial ontology with predefined categories and binary relations. Each category comes with a predefined set of seed instances and each relation comes with a predefined set of seed pairs. For example, the category of *insects* is filled with 10 initial known insect names. The relation *playsFor* is filled with 10 initial pairs of soccer players and soccer teams. Furthermore, the system is also given a set of mutual-exclusion constraints. For example, the system knows that vehicles and people are disjoint. In addition, domain and range constraints are also supplied. The key idea of NELL is to use not just one extractor, but multiple extractors in parallel. Information gathered by one extractor is used to support the other extractors. This principle is called *coupled training*.

More precisely, NELL uses three learners: The first learner is the Coupled Pattern Learner (CPL), which extracts patterns and facts from natural language text. Given a document and given seed instances for categories, CPL extracts the contexts in which the seeds appear. For example, if it is known that Microsoft is a company, and if Microsoft appears in a sentence like “Bill Gates was the CEO of Microsoft”, then the CPL will extract the pattern “CEO of X” as a pattern that identifies companies. If the CPL finds another sentence in which this pattern appears, it will propose the corresponding word as an instance of the category *company*. The CPL uses heuristics based on POS-tagging to determine the size of the patterns. The same idea applies to binary relationships: If a seed pair for a relation appears in a sentence, the CPL extracts the words between the two elements as a pattern. If this pattern appears again with two other elements, the

---

<sup>9</sup>Never-Ending Language Learner, <http://rtw.ml.cmu.edu/rtw/overview>

CPL proposes this pair of words as an instance of the relationship. The CPL uses some manually defined patterns for bootstrapping, which include the Hearst patterns [10].

The second learner is the Coupled SEAL (CSEAL), which is a modified version of SEAL [24]. CSEAL extracts patterns and facts in a similar way to CPL, but from structured documents. This means that the patterns that CSEAL learns are sequences of characters around the seed, which may include text, punctuation, HTML tags and HTML attributes. Just like CPL, CSEAL will learn new patterns from occurrences of seeds and use these patterns to propose new instances.

The third learner is the Meta-Bootstrap-Learner (MBL). The MBL runs in parallel with CPL and CSEAL and controls these extractors. For every instance that the extractors discover, MBL checks whether it satisfies the domain and range constraints and the exclusion constraints specified by the ontology. If that is not the case, MBL tells the extractor to filter out that instance. As a consequence, the extractor will downgrade the patterns that extracted this instance. Vice versa, MBL can also tell the extractor to promote an instance, if that instance was found by both extractors. As a consequence, the extractor will boost the patterns that extracted this instance. Through this coupling, MBL exploits the synergy of the two extractors and uses the results of one extractor to boost the performance of the other.

The NELL system has been continuously running since January 2010. The results undergo periodic screening and correction by a human. New categories and relations are also being added.

### 3.2.2. Summary

As of February 2011, NELL has accumulated a knowledge base of 500,000 asserted instances of 589 different categories and relations, at an estimated average precision of 87%. NELL is a live system and its progress can be followed online at the project homepage. The main technical contribution of the NELL project is the coupling of different extractors, which boost and control each other. Through the mutual exclusion rules and the type constraints, NELL can ensure a certain consistency of its results. Due to its reliance on Hearst Patterns, NELL is particularly strong on extracting *type* and *subclassOf* facts. This allows NELL to build a taxonomic structure.

Canonicity has not yet been a focus of the project. NELL does not work on ontological entities, but on strings, which may be synonymous or polysemous. We will next see a project that targets also this disambiguation of entities.

### 3.3. SOFIE

SOFIE<sup>10</sup> is an IE system that aims to extend the YAGO ontology [20] with facts extracted from natural language documents [22]. It is part of the YAGO-NAGA project at the Max Planck Institute for Informatics in Germany. SOFIE aims to solve the canonicity problem, the pattern discovery and the consistency problem in one unified framework.

#### 3.3.1. Model

In previous work, the problems of disambiguation, consistency reasoning and IE have mostly been attacked in isolation. However, these problems are highly intertwined. Con-

---

<sup>10</sup>Self-Organizing Framework for Information Extraction, <http://mpii.de/yago-naga/sofie>

sider, e.g., the sentence “Elvis died in 460 AD.”. If the ontology already knows that Elvis Presley was born in 1935, then the word “Elvis” in the sample sentence cannot refer to Elvis Presley, because people have to be born before they die. This indicates that consistency constraints can guide disambiguation. Likewise, disambiguation can produce hints for pattern discovery and so on. Therefore, SOFIE aims to solve all three problems in one unified framework. This framework is based on the Weighted Maximum Satisfiability problem (*Weighted MAX SAT*). In this setting, all data is expressed as logic formulas.

SOFIE first describes the input documents as logic formulas. For every pair of proper names or numbers that appears in the text within a certain window, SOFIE generates a proposition of the form  $patternOcc(Elvis@D1, \text{“died in”}, 460)$ . This proposition means that the word “Elvis” appeared with the string “died in” with the number 460.  $Elvis@D1$  identifies the word “Elvis” in document  $D1$ . Even though SOFIE does not yet know which entity is meant by “Elvis”, SOFIE assumes that all occurrences of that word in the same document have the same meaning. Therefore, all those words are identified by the same token, a *wic* (word in context). So far, the generated proposition makes a statement only about the occurrence of certain strings. It does not yet make a statement about entities.

SOFIE extracts facts only about entities that are already known to the ontology. Furthermore, SOFIE assumes that the ontology knows all names that are commonly used to refer to an entity. This allows SOFIE to make hypotheses about the meaning of the wics. By taking into account the context of the wic (the bag of words in the document) and the context of a possible meaning of the wic (the names of the entities related to the entity in question in the ontology), SOFIE can assign a weight to these hypotheses. In the example,  $Elvis@D1$  could give rise to the following propositions (with weights):

$means(Elvis@D1, ElvisPresley)[0.8]$   
 $means(Elvis@D1, SaintElvis)[0.3]$

SOFIE also maps the ontology to logic formulas. In the case of YAGO, which just contains simple triples, every triple of YAGO becomes one proposition. For example, SOFIE will know  $wasBornOnDate(ElvisPresley, 1935)$ . SOFIE also makes use of semantic rules. These have to be supplied manually. One example for such a rule is

$bornOnDate(X,Y) \text{ and } diedOnDate(X,Z) \Rightarrow Y < Z (*)$

Other, freely configurable rules can enforce functional dependencies or incorporate domain knowledge. In addition, SOFIE uses rules that glue the framework together:

$patternOcc(WX, P, WY) \text{ and } expresses(P, R)$   
 $\text{and } means(WX, X) \text{ and } means(WY, Y)$   
 $\Rightarrow R(X, Y)$

This rule means: If pattern  $P$  appears with wics  $WX$  and  $WY$  and  $P$  expresses the YAGO relation  $R$ , and if  $WX$  has been disambiguated to entity  $X$ , and  $WY$  to  $Y$ , then  $X$  and  $Y$  stand in relationship  $R$ . In the example, assuming that “died in” is known to express  $diedOnDate$ , we could deduce  $diedOnDate(ElvisPresley,460)$ . An analogous rule says that if a pattern  $P$  occurs with two entities, and if the ontology knows that the entities stand in relationship  $R$ , then  $P$  expresses  $R$ . For example, if YAGO knows that Einstein died in 1955, then a sentence of the form “Einstein died in 1955” (with appropriate disambiguation) will trigger the deduction of  $expresses(\text{“died in”}, diedOnDate)$ .

### 3.3.2. Reasoning

All of the above formulas are grounded, i.e., one creates one instance of the formula for every possible instantiation of the variables. All formulas that do not have an explicit weight are given a large weight  $W$ . This gives SOFIE a huge set of weighted propositional formulas. These formulas can never all be true at the same time. *means*, for example, is declared a function, but already has two images for *Elvis@DI*. Furthermore, the deduction of pattern meanings may be overly hasty and wrong. Lastly, the documents themselves may contain contradictory information. Therefore, SOFIE aims to find an assignment of truth values to the propositions that maximizes the weight of the satisfied formulas. In the example, assigning *true* to the proposition *means(Elvis@DI, ElvisPresley)* will gain a weight of 0.8. At the same time, this assignment will make it impossible to satisfy rule (\*). Therefore, SOFIE might decide to make *means(Elvis@DI, SaintElvis)* true instead and to deduce *diedOnDate(SaintElvis, 460)*. Finding the optimal assignment of truth values to the propositions is an NP-hard problem. SOFIE implements a heuristic algorithm that runs in polynomial time and has an approximation guarantee of 1/2.

The optimal solution of the Weighted MAX SAT problem corresponds to the most plausible interpretation of the documents, given the background knowledge. SOFIE can continuously analyze new documents and compute the most plausible interpretation of the evidence at any time. As more evidence accumulates, the most plausible interpretation of it may change over time.

### 3.3.3. Summary

SOFIE has been run on Web documents and has been shown to have a precision of 90%. This includes both the correctness of the facts and the correct disambiguation of entities. The main technical contribution of SOFIE is the modeling of disambiguation, pattern deduction and consistency in one single framework. This allows SOFIE to exploit synergies between these three tasks. Through its automated reasoning, SOFIE achieves both canonicity of the extracted facts and consistency with the ontology. SOFIE does not find new classes. It re-uses the existing taxonomy of YAGO. SOFIE has been extended to the PROSPERA system [14] to run in a parallelized way on large scale.

## 4. Conclusion

In this chapter, we have discussed IE for ontologies. Three main desiderata for ontological IE are the canonicity of entities and relationships, the consistency of the extracted facts and the taxonomic organization of the data. We have seen 3 approaches that extract information from Wikipedia with these goals in mind: DBpedia, YAGO and KOG. These systems can already use different degrees of reasoning to ensure the consistency of the extracted data. We have also discussed ontological IE from the Web. Such approaches can use the redundancy of the Web to mutually reinforce pattern discovery and fact discovery. We have seen 3 such systems, OntoUSP, NELL, and SOFIE. All of them use some degree of reasoning to ensure logical consistency of the extracted facts.

The rapid growth of the Web is drawing more and more attention to the “semantic gap” between syntactic text and semantic knowledge. At the same time, the availability of large scale knowledge bases (such as for example those on the Web of Linked Data

[3]) opens up new ways of using structured information to deal with unstructured text. These two trends have nurtured a strong interest in ontological IE. Today, numerous research groups work on making the Web semantically accessible. Newer IE approaches go beyond fact extraction and extract entire axioms. For example, IE can extract entire OWL axioms from the first sentences of Wikipedia articles [23].

## References

- [1] Eugene Agichtein, Luis Gravano, Jeff Pavel, Viktoriya Sokolova, and Aleksandr Voskoboinik. Snowball: a prototype system for extracting relations from large text collections. *SIGMOD Records*, 30(2):612, 2001.
- [2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735, Berlin, Germany, 2007. Springer.
- [3] Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the Web. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1265–1266, New York, NY, USA, 2008. ACM.
- [4] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semant.*, 7:154–165, September 2009.
- [5] Sergey Brin. Extracting patterns and relations from the world wide web. In *Selected papers from the International Workshop on the WWW and Databases*, pages 172–183, London, UK, 1999. Springer.
- [6] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*, 2010.
- [7] Gerard de Melo and Gerhard Weikum. Towards a universal wordnet by learning from combined evidence. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, pages 513–522, New York, NY, USA, 2009. ACM.
- [8] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [9] Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 577–583, Menlo Park, CA, USA, 2000. AAAI Press.
- [10] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the International Conference on Computational Linguistics (ICCL)*, pages 539–545. Association for Computational Linguistics, 1992.
- [11] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, 2013.
- [12] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of the International Conference on Very Large Databases (VLDB)*, 3:1338–1347, September 2010.
- [13] Diana Maynard and Kalina Bontcheva. Natural language processing. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.
- [14] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 227–236, New York, NY, USA, 2011. ACM.
- [15] Hoifung Poon, Janara Christensen, Pedro Domingos, Oren Etzioni, Raphael Hoffmann, Chloe Kiddon, Thomas Lin, Xiao Ling, Mausam, Alan Ritter, Stefan Schoenmackers, Stephen Soderland, Dan Weld, Fei Wu, and Congle Zhang. Machine reading at the university of washington. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 87–95, 2010.
- [16] Hoifung Poon and Pedro Domingos. Unsupervised ontology induction from text. In *Conference of the Association for Computational Linguistics (ACL)*, pages 296–305, 2010.



- [17] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.
- [18] Sunita Sarawagi. Information Extraction. *Foundations and Trends in Databases*, 2(1), 2008.
- [19] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *Proceedings of the International Conference on Very Large Databases (VLDB)*, 5(3):157–168, 2011.
- [20] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In Carey L. Williamson, Mary Ellen Zurko, and Prashant J. Patel-Schneider, Peter F. Shenoy, editors, *Proceedings of the International Conference on World Wide Web (WWW)*, pages 697–706, Banff, Canada, 2007. ACM.
- [21] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO - A Large Ontology from Wikipedia and WordNet. *Elsevier Journal of Web Semantics*, 6(3):203–217, September 2008.
- [22] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. SOFIE: A Self-Organizing Framework for Information Extraction. In *International World Wide Web conference (WWW 2009)*, New York, NY, USA, 2009. ACM Press.
- [23] Johanna Völker, Pascal Hitzler, and Philipp Cimiano. Acquisition of OWL DL axioms from lexical resources. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *Proceedings of the 4th European Semantic Web Conference (ESWC)*, volume 4519 of *Lecture Notes in Computer Science*, pages 670–685. Springer, JUN 2007.
- [24] Richard C. Wang and William W. Cohen. Character-level analysis of semi-structured documents for set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1503–1512, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
- [25] Fei Wu and Daniel S. Weld. Automatically refining the Wikipedia infobox ontology. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 635–644, New York, NY, USA, 2008. ACM.