A Case Study on Information Extraction from the Internet: Populating a Movie Ontology

Gijs Geleijnse
Media Interaction Group
Philips Research Laboratories
Eindhoven, the Netherlands
gijs.geleijnse@philips.com

Abstract

A method to extract information from the internet is discussed. We choose to formally represent information by ontologies, and the information extraction problem is reformulated as an ontology population problem.

A case study on populating a Movie ontology is presented. We extract instances of some class of the ontology by analyzing query results from the search engine Google. Within the abstracts which Google generates, we interpret natural language expressions to find instances and their relations.

We present the algorithm and discuss the results of our work. Ideas for future work are given as well.

1 Introduction

Suppose that we want to use the internet to find information, but we do not know whether or where it can be found. A good heuristic is to use a search engine to find web pages with relevant content. However, current search engines retrieve web pages, not the information itself. We have to search within the search results in order to acquire the information. Moreover, we make implicit use of our knowledge (e.g. of the language and the knowledge domain), to interpret the web pages.

Our aim is to design a system which automates this process. It thus has to retrieve, combine and structure information from the internet to make it easily (machine) accessible.

To gain insight in the problems and possibilities in this field of research, we have done a case study on a specific domain of interest: information about movies. Information like "all movies in which the director is also part of the cast" or "movies with both Madonna and Sean Penn", we wish to extract with the algorithm to-be-developed.

A straight forward method to extract such information, is to implement a wrapper (Maedche, Staab, 2001), which extracts information from a structured web site on the domain of interest. The Internet Movie Database[1] is a good candidate for this purpose.

However, we are interested in a technique which is domain independent. Therefore, we concentrate on information extraction from natural language texts.

We have chosen to study the movie domain for two reasons. Firstly, numerous web pages handle this topic: the query 'movie' in Google results in 180,000,000 hits. The performance of the algorithm will thus not or barely be influenced by the lack of data available. Secondly, we can easily verify the results and formulate benchmarks for evaluation purposes.

The work presented in this paper, relates to a wide range of disciplines. We mention ontology learning (Maedche, Staab, 2001), automatic term discovery (Frantzi et. al., 2000), pattern extraction (Brin, 1998) and information extraction with the use of a search

[1] http://www.IMDb.com

engine (Cilibrasi, Vitanyi, 2004).

The organization of this article is as follows. In the next section, we sketch an approach for our problem and highlight the choices we have made in our case study. Section 3 discusses the outline of the algorithm we have implemented in the case study, while Section 4 handles the experimental results. In the last two sections, conclusions and ideas for future work can be found.

2 Approach

Our problem of information extraction from NL texts on the internet falls apart into four research questions:
1. How to formalize the concept *information*?
2. How to find possibly relevant web pages?
3. How to extract information from these pages?
4. How to present this information?

The focus in this article will be on the third question (Section 2.2). In the first section, we give a structure to represent information.

We find possibly relevant web pages, using the Google search engine[2]. If necessary, in future work we can explore possibilities to design a dedicated web crawler.

A method to present the information, is to publish it in a Semantic Web language (w3c, 2004). With the use of query languages for the Semantic Web, a user can easily automatically extract relevant information from the content.

2.1 A formal representation of the concept *information*

In extension to the Semantic Web nomenclature, we use ontologies to represent information. For our purposes, we define a reference ontology O by a 4-tuple (C, I, P, T), where

- $C = (c_0, c_1, ..., c_{N-1})$, an ordered set of N classes,
- $I = (I_0,\ I_1,\ ...,\ I_{N-1})$, with I_j, $0 \leq j < N$, the set of instances of class $c_j \in C$,
- $P = (p_0, p_1, ..., p_{M-1})$, a set of M binary relations on the classes, with $p_i : c_{i,0} \times c_{i,1}$, $0 \leq i < M$, and $c_{i,0}, c_{i,1} \in C$, and
- $T = (T_0,\ T_1,\ ...,\ T_{M-1})$, is a set of instances of the relations in P, with $T_j = \{(s,o) \mid p_j(s,o)\}$ for each $j,\ 0 \leq j < M$ and $s \in I_{j,0}$ (an instance of $c_{j,0}$) and $o \in I_{j,1}$ (instance of $c_{j,1}$).

A *partial ontology* of O is defined as O' = (C, I', P, T'), where

- $I'_j \subseteq I_j$, (for all j, $0 \leq j < N$)
- $T'_j \subseteq T_j$, (for all j, $0 \leq j < M$)
- $(s,o) \in T'_k \Rightarrow s \in I'_i \wedge o \in I'_j$ for some i, j, k.

We reformulate the information extraction problem as an ontology population problem: given a partial ontology O', extend O' to some O'' which approximates the reference ontology O as well as possible.

Two main criteria are commonly used to quantify this approximation: *precision* and *recall*. We give these measures of a class $c_i \in C$ as

$$precision(c_i) = \frac{|I_i \cap I''_i|}{|I''_i|}$$

and

$$recall(c_i) = \frac{|I_i \cap I''_i|}{|I_i|}\ .$$

Similar measures can be formulated for relations p_j.

We have selected a small partial ontology on the movie domain. It is defined as

$$\begin{aligned} O'_{movie} = \ (\ & (\ Director, Actor, Movie)\ , \\ & (\ \{\ Spielberg, Coppola\}, \emptyset, \emptyset)\ , \\ & (\ acts_in(Movie, Actor), \\ & \quad director_of(Movie, Director))\ , \\ & (\ \emptyset, \emptyset) \\ ). & \end{aligned}$$

2.2 Information extraction from web pages

The focus of this study is, to populate an ontology by analyzing natural language texts. In earlier work, Hearst (1992) identifies methods to extract information from enumerations in natural language texts (e.g. "*i, j* and other *C*"). Verschoor et. al. (2004) show that this technique is useful to acquire instances of some class in an ontology.

In this research, we firstly find NL formulations of relations p_k. Then, given a NL formulation of p_k

[2] `http://www.google.com`

and an instance s, we scan the texts for NL formulations of $p_k(s, o)$. If such an o is found, we add it as an instance to the ontology O''. We thus simultaneously find instances and instances of relations.

Verschoor et. al. (2004) have done work in this area, but they focus on identification of relations, with a given set of instances.

The following two important criteria for the NL formulations of relations p_k can be identified.
(Precision.) This phrase must be unambiguous, i.e. the probability that the terms found do not belong to the intended class must be small. For example, consider the relation place_of_birth(Person, City). The expression 'was born in' is not an unambiguous representation of this relation, since it can proceed the name of a country or a date as well. This formulation is therefore rejected.
(Recall.) The expression must frequently occur.

At the moment, we have selected the phrases manually. In future work, we want to find and implement (semi-) automatic techniques to find such expressions. A way to do so is to analyze how the relation between a pair of instances (e.g. (Germany, Berlin)) is expressed in natural language. When we do so for a number of pairs (e.g. (Germany, Berlin), (Canada, Ottawa),...), we can get an impression of commonly used expressions and select one (or more) based on the precision criterion.

```
(1) identify NL phrases for all p_k.
(2) ; identify lists of stopwords
(3) ; O'' := O'
(4) ; add all the instances to U
; while U ≠ ∅ do
 ( 5) select i ∈ U
 ( 6) ; query i with NL phrase
 ( 7) ; extract all j st p_k(i,j)
 ( 8) ; add all j and p_k(i,j) to O''
 ( 9) ; add all j to U
 (10) ; remove i from U
od
```

Table 1: pseudo code of the algorithm

3 Outline of the Approach

We are aiming for an algorithm, that uses its output again as input. That is, when an instance of some class is identified, this instance can be used to find instances of other classes. We use the set U in the pseudo code in Table 1 to identify the set of instances which have not been used for this purpose. The algorithm terminates, when no new instances can be found.

The numbers between brackets in this section refer to the corresponding lines in Table 1.

3.1 NL phrase selection

In order to find instances, we have to select natural languages phrases which express relations between the classes (1). These phrases have been selected manually. That is, we have queried several pairs ((Movie, Director) and (Movie, Actor)) in Google and analyzed the way their relationship was expressed. We applied the criteria in Section 2.2 and selected the following three expressions, given in italics:
- (Director) *'s* (Movie)
- *director:* (Director). (Movie) *starring* (Actor) (, (Actor) and (Actor))
- (Movie) *starring* (Actor) (, (Actor) and (Actor)) *directed by* (Director)

When we have selected the NL phrases, we use them in queries we hand to the Google search engine (Section 3.3, (6)). The abstracts that Google retrieves are used to find new instances for our ontology (7). Section 3.4 discusses this issue in detail.

3.2 List of stopwords

A list of stopwords is a list of words and symbols that do not occur in an instance of some class. We use such lists to improve the precision.

Since the classes Actor and Director both contain person's names, we use the same stopword list. This list contains 90 words that typically do not occur in person's names, but do frequently occur in a web page on movies, e.g. biography, DVD and review.

It is harder to construct a stopword list for titles of movies, since titles can contain all sorts of words. The stopword list of this class only contains 30 words.

The lists have been constructed manually, after

running the algorithm without the use of a stopword list and analyzing the retrieved false instances.

3.3 Googling

In each phase of the algorithm, an instance i from U is used in a Google-query `(6)`. This query is a combination of i and the corresponding NL phrase.

In the case study, one of the following queries to Google is performed:

"(Director)'s"
"starring (Actor)"
"(Movie) starring"

The Google search engine returns abstracts of the web pages in which the queried phrases occur. We have chosen to only retrieve the first 100 of abstracts, since they are to be the most relevant results.

3.4 Instance extraction

In `(7)`, we scan the generated abstracts for one of the following patterns, depending on the queried expression in `(6)`:
(Director)'s (Movie)
(Movie) *starring (Actor)*
director: (Director). *(Movie) starring* (Actor) (, (Actor) and (Actor))
(Movie) starring (Actor) (, (Actor) and (Actor)) directed by (Director)

Above, the queried expressions are given in italics. The problem is to identify the other instances in the texts. Note that when we send a query "(Movie) starring" to Google, we scan the abstracts for both instances of Director and instances of Actor.

The problem is to identify the (multiple word) instances in the texts. In this case study, we look for two categories: movie titles and names of persons. Instances of these categories differ syntactically. We therefore use different approaches to extract them.

In general, a person's name consists of two or three words, each of them starts with a capital. If such a term occurs in the text, fitting the pattern given above, we check if it does not contain a stopword. If so, we identify it as a candidate instance.

The recognition of a name of a movie is more difficult, because of a number of reasons. We mention:
1. The lengths of the titles vary.
2. The words in the titles do not always start with a capital.
3. A title can include punctuation marks.

For the moment, we have chosen to only identify movie titles which are denoted between quotation marks or apostrophes. When an expression between quotation marks or apostrophes fits the pattern, and this expression does not contain a stopword, it is a candidate instance.

In an earlier version of the algorithm, we identified titles which were denoted in capitals as well, but this variant had poor precision. In future work, we want to make use of Automatic Term Recognition techniques (Frantzi et. al., 2000), in order to come to a more generic approach.

Now, when we have identified a candidate instance, we perform a check whether this candidate is really an instance of the concerning class. We give – depending on the class – one of the following queries to Google:
"The movie *Movie*"
"*Actor* plays"
"*Director* directed"

A candidate instance is accepted as an instance, when the concerning query results in enough hits (we used 10 or more).

These natural language phrases are also selected manually. In future work however, we will do research on a semi-automatic acquisition of such phrases.

The last steps in a phase in the algorithm consist of the updating of O'' and the set U, `(8)` - `(10)`.

4 Experimental Results

Our first observation is, that all the tests we have done, terminate. They all did so in 12 to 16 hours time, depending on the accessibility of Google and load on the local network.

We first ran the algorithm with the names of two (well-known) directors as input: Francis Ford Coppola and Steven Spielberg. Afterwards, we experimented with larger sets of directors and small sets with less famous directors as input.

Another interesting observation is, that the outputs are relatively independent from the input

sets. That is, when we take a subset of the output of an experiment as the input of another experiment, the outputs will be roughly the same. The small differences between the outputs can be explained by the changes in the contents of the web and the ranking of the Google search engine.

We have found 7,000 instances of the class Actor, 3,300 of Director and 10,000 of Movie. The number of retrieved instances increases, about 7%, when 500 query results are used instead of 100.

We have also kept track of the number of times an instance was found in the Google abstracts. We give the top-5 and the number of times they were identified within the abstracts (N), of each class. Moreover, for the actors and directors, we give the number of movies we have found in which they participate (M).

	Top 5 Actors:	N	M
1.	Denzel Washington	1443	23
2.	Johnny Depp	1431	12
3.	Tom Hanks	1395	16
4.	Jim Carrey	1372	19
5.	John Travolta	1320	21
	Top 5 Directors:		
1.	Alfred Hitchcock	452	4
2.	Ingmar Bergman	228	6
3.	John Carpenter	223	5
4.	Robert Altman	210	6
5.	Steven Spielberg	197	10
	Top 5 Movies:		
1.	Hamlet	52	
2.	Beat	39	
3.	Ray	35	
4.	Lost in Translation	32	
5.	Elf	32	

The occurrences of the movies are relatively low compared to the other classes, because of the method we used to identify movie titles.

Precision

When we analyze precision of the algorithm, we use the data from IMDb, as a reference. An entry in our ontology is accepted as a correct one, if it can be found in the IMDb-database.

We have manually checked three sequences of 100 instances (at the beginning, middle and end of the generated file) of each class. We conclude that 78 % of the instances are correct and unique.

We identify four categories of incorrect instances in the ontology:

1. Different formulations of the same instance, e.g. “the Karate Kid” vs. “Karate Kid”, and “Leo DiCaprio” vs. “Leonardo DiCaprio”. In the future, we hope to be able to recognize these cases by analyzing the context (e.g. when 2 actors act in the same set of movies) and by using of approximate string matching techniques (Gusfield, 1997).
2. Common misspellings. We hope to identify them, using the same techniques.
3. Different titles for the same movie (e.g. “Hable con Ella” vs. “Talk to Her”). Often, both titles are mentioned on web sites. When two movie titles share the same director and set of actors, a check can be done to find the relationship between the two titles.
4. False Instances. On each candidate instance we perform a check. However, terms like James Bond and Mickey Mouse slip through this check and are identified as instances. We believe that this category is harder to solve, if we do not want to deteriorate the recall.

We have also analyzed the precision on the relationships, by taking a random set of about 200 movies with there actors and directors. If we do not take false instances of the classes into account, the precision of the relations between movie and director is around 85 %, between movie and actor around 90%.

Recall

We have formulated various benchmark sets to analyze recall. Firstly, we made lists of all Academy Award winners (1927-2004) in a number of categories, and checked the recall:

category	recall
Best Actor	96%
Best Actress	93%
Best Director	98%
Best Picture	85%

An actress we did not find was Cher, since her name, consisting of only one word, is not recognized as such by the algorithm.

IMDb has a top 250 of best movies ever. The algorithm found 85% of them.

We observe that results are strongly oriented on Hollywood productions. We have made a list of all winners of the Cannes Film Festival, the 'Palm d'or'. Alas, our algorithm only found 45% of the winning movies in this category.

5 Conclusions

We studied the problem of automatic information extraction from the internet. We translated this problem into an ontology learning problem.

With the use of a web crawler (e.g. Google), text on web pages can be found with possibly relevant data.

We presented a detailed description of a method to extract instances of a given class from NL text fragments.

The performance of the algorithm in a restricted field – the Movie domain – is encouraging. The output of the algorithm is not influenced by the input set, under the restriction that items in this set are correct and exemplary.

Both recall and precision measures are high. Moreover, there exist possibilities to improve the precision of the algorithm.

We have given ideas to create a more generic algorithm.

6 Future Work

The technique to identify candidate instances in the texts is open to improvement. This will be necessary, if we want to develop an algorithm which is not (or less) domain specific. We will experiment with Automatic Term Recognition techniques (Frantzi et. al., 2000).

Apart from these improvements, we want to combine the technique described in this paper, with the techniques from (Hearst, 1992) and (Verschoor et. al., 2004).

Currently, we do not recognize different formulations for the same instance within our ontology. With the use of approximate string matching techniques, combined with reasoning with the generated ontology, we hope to discover such occurrences. We expect this will improve the precision of the algorithm.

The selection of natural language phrases which we use in our queries can be done semi-automatically. We want to automatically analyze how typical instances of two classes are linked to each other in natural language. The algorithm can then give the user suggestions on which phrase to choose.

Moreover, we will do research on techniques to automatically find the phrases which we use to check candidate instances.

Acknowledgements

I gratefully thank Jan Korst and Nick de Jong for their help and suggestions.

References

Sergey Brin. 1998. *Extracting patterns and relations from the World Wide Web*. Proceedings WebDB Workshop at EDBT'98.

Rudi Cilibrasi, Paul Vitanyi. 2004. *Automatic Meaning Discovery Using Google*. `www.cwi.nl/~paulv/papers/amdug.pdf`.

Katerina Frantzi, Sophia Ananiado, Hideki Mima. 2000. *Automatic recognition of multi-word terms: the C-value/NC-value method*. International Journal on Digital Libraries 3: 115-130.

Hearst, M. 1992. *Automatic acquisition of hyponyms from large text corpora*. Proceedings of the 14th International Conference on Computational Linguistics, Nantes, France, 539-545.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Alexander Maedche, Steffen Staab. 2001. *Ontology Learning for the Semantic Web* in *IEEE Intelligent Systems* 16(2): 72-79.

Gautam Pant, Padmini Srinivasan, Filippo Menczer. 2003. *Crawling the web*. in M. Levene and A. Poulovassilis, editors: Web Dynamics, Springer-Verlag.

Michael Verschoor, Jan Korst and Nick de Jong. 2004. *Ontology-based Extraction of Information from the Internet*. Presented at IPA Fall days 2004, `http://www.win.tue.nl/ipa/`.

The World Wide Web Consortium (W3C) 2004. `http://www.w3.org/`