UNIVERSITÀ DEGLI STUDI DI CAGLIARI

FACOLTÀ DI SCIENZE

Corso di Laurea in Informatica

A Pattern-Induced RDF Statement Extraction System

Docente di riferimento:
Maurizio Atzori

Candidato:
Riccardo Angius
(matr. 45775)

ANNO ACCADEMICO: 2012/2013

Abstract

The remarkable quantity of structured data extracted by DBPedia from Infoboxes on Wikipedia articles lends itself as a great starting block for further extraction of data. The project aims to collect and catalogue the natural language patterns therewith the data is presented on the actual discourse of Wikipedia articles, and exploit these patterns in order to obtain and store an analogue data set (i.e. RDF statements pertaining to the predicates associated with these patterns) from both within Wikipedia and external text sources. Mimicking the natural human approaches as defined by the current chunking theories of language acquisition, the experimental algorithms developed for the purpose employ the model of Stanford Typed Dependencies to reach a precision rate of 0.26 and a recall rate of 0.26. These result from tests on 200 sentences sampled from the same corpora of 51,536 Wikipedia articles concerning human settlements (cities, towns, etc.) used for collecting patterns in conjunction with a training set retrieved article by article from DBPedia, and do not consider as retrieved the statements obtained by matching a pattern to the sentence it originated from.

Contents

Chapter 1

Introduction

The average person has developed an ability to understand the semantics of a sentence with a complex structure, such that they will not have to stop to think about every single piece of information contained in each and every word. In fact, they will unconsciously start folding them and their meanings together as soon as they come, and they will have an almost-complete grasp of the sentence's meaning as soon as it has been completely received (save for complex concepts unfamiliar to the sentence's recipient). This is something that has been widely accepted by both parties in the psycholinguistic field of the Nature versus Nurture debate, as it lies outside the theories on whether such mechanism relies on innate or taught abilities.

As much as one is supported in this process by formal knowledge of grammatical structures, introduced in school some years after one has already developed an ability to speak their L1, a big part of language acquisition lies in being systemically presented with fixed chunks of words, carrying a general meaning, along with one or more inset references that vary. As argued in [14] this systematic repetition teaches one to induce the acquired chunks so to reapply them to analogue concepts: this is, in fact, what toddlers do when first learning language.

The aim of the proposed system is to induce patterns modelled on the Stanford Typed Dependencies, identified on the corpus of Wikipedia articles through the help of the structured data made available by the DBPedia project, on any kind of descriptive text so to extract additional structured data, and verify whether the chunking theories are a viable approach to automatic collection of such data from natural language.

1.1 The Resource Description Framework

The possibilities of a Semantic Web, as detailed in [4], urged the World Wide Web Consortium to publish a set of standards for the collection of structured, machine-readable data regarding relations between resources on the Web, the Resource Description Framework.

The definition of "resource" initially referred solely to documents retrievable in the Web (e.g. a webpage), but was later extended to include any thing that can be therein identified, i.e. to whom a Uniform Resource Identifier (URI, a generalisation of URL) has been assigned. Through these URIs any independently intelligible entity (e.g. a person, a geographic region, a concept, etc.) can be linked to any else through what the frameworks calls a statement.

Statements allow for the definition of an ontology, classifying resources as being an in-

stance of a class, and for classes to be subclasses of other classes. The framework, in addition to providing a way to define new ontologies (and integrate them with the pre-existing), comprises a basic ontology necessary for its own definition.

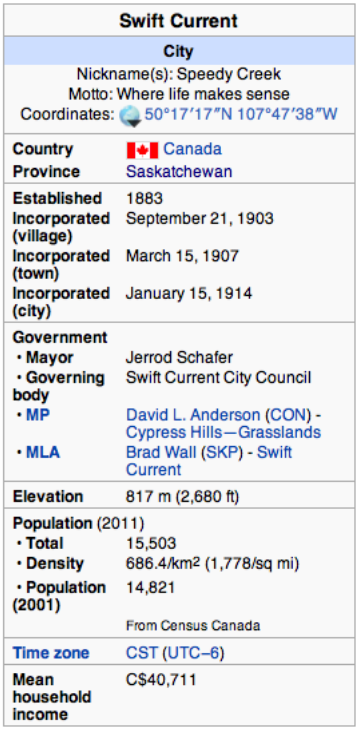
Each RDF statement is a triple σ:

$$\sigma := (S, P, O)$$

where S is the statement's subject, P is the statement's predicate and O is the statement's object. The framework requires every component of a triple to be denoted by an URI, with the O component denotable by a typed literal as well.

Additionally, the predicate needs to be an instance of a RDF property, i.e. a resource whose hierarchy of superclasses must include `rdf:property`[1], defined as the URI for a relating concept between two other resources. Moreover, the framework allows the restriction of valid resources denoted by the S and O components to instances of classes specified through `rdfs:domain` and `rdfs:range` statements, respectively.

1.2 DBPedia



Swift Current	
City	
Nickname(s): Speedy Creek Motto: Where life makes sense Coordinates: 50°17′17″N 107°47′38″W	
Country **Province**	Canada Saskatchewan
Established **Incorporated (village)** **Incorporated (town)** **Incorporated (city)**	1883 September 21, 1903 March 15, 1907 January 15, 1914
Government • **Mayor** • **Governing body** • **MP** • **MLA**	 Jerrod Schafer Swift Current City Council David L. Anderson (CON) - Cypress Hills—Grasslands Brad Wall (SKP) - Swift Current
Elevation	817 m (2,680 ft)
Population (2011) • **Total** • **Density** • **Population (2001)**	 15,503 686.4/km² (1,778/sq mi) 14,821 From Census Canada
Time zone	CST (UTC–6)
Mean household income	C$40,711

Fig. 1.1 Detail of an Infobox on Wikipedia, presenting a common template for cities [16].

DBPedia is a conceptual graph bootstrapped by assigning to each and every page in Wikipedia a URI. For instance, the resource described by the Wikipedia article with title "Swift Current" is assigned the URI `dbpedia:Swift_Current`.

DBPedia then proceeds to leverage the power of RDF, through the aforementioned statements, to relate the obtained collection of resources. The statements are extracted from the Infoboxes present on Wikipedia articles, which are a semi-structured data model designed to be filled out by human writers with data relevant to the article. There exists a variety of templates for Infoboxes, ranging from cities to cars, musicians and so forth, all custom-designed for each specific kind of encyclopedic entity. An example of an extracted RDF statement from the Infobox in Figure 1.1 would be (`dbpedia:Swift_Current`, `dbpedia-owl:country`, `dbpedia:Canada`).

Unfortunately, Wikipedia was never built with the goal of collecting structured data, hence the

[1] This is short-hand for `http://www.w3.org/1999/02/22-rdf-syntax-ns#property`. URIs are often grouped in logical collections, having them share the same string for their first part. The shared string is called a namespace and the framework allows for a properly declared prefix, followed by a colon, to be used in its place in URIs. For a complete list of namespaces and prefixes used in this document, see Appendix B.

standards followed vary greatly from one template to another, as they are solely the result of customs initiated by the contributor that created the template when the need arose. Consider three classes of people, e.g. politicians, actors and singers: they are all people, thus share the kind of properties a person has, but an Infobox must catalogue and present different information for each of them as well (e.g. party, first appearance on the screen and musical genre, respectively), hence different templates are used.

The problem arises when the shared properties are referred to with different identifiers: for instance, *birth_place*, *birthPlace* and *origin* are all used in different Infobox templates to catalogue the same piece of information. A traditional hierarchical structure was never designed and imposed for template creation, thus there is no easy or automated way to re-instate order.

The effort of DBPedia contributors maps to consistent URIs in its ontology all these idiosyncratic (*raw*, in DBPedia's terminology) properties. In our example, all the aforementioned names for one's birth date are mapped to a single one, i.e. `dbpedia-owl:birthDate`. Although not every raw name has been mapped yet, as every day new templates are created and older ones are extended with new properties, the availability of mapped ones is rather considerable. At the moment, the DBPedia Data Set, extracted from the English Wikipedia, comprises about 33 million triples with standardised URIs for their predicates, providing a vast collection of unambiguous relations between entities.

1.3 The Stanford Typed Dependencies

Introduced in [7], the Stanford Typed Dependencies were designed to describe relations existing between words in a sentence. The model goes one step further from Phrase Structure Grammars (PSG, an example of which is shown in Figure 1.2), composed of production rules that rewrite a sentence so to obtain a tree whose non-terminal nodes consist in syntactic categories, whereas terminal ones are words from the sentence. This structure can hinder semantic analysis, making it overly complex, as syntactic categories include a higher degree of grouping of words, namely phrasal categories, above the unitary classification through the traditional parts of speech.

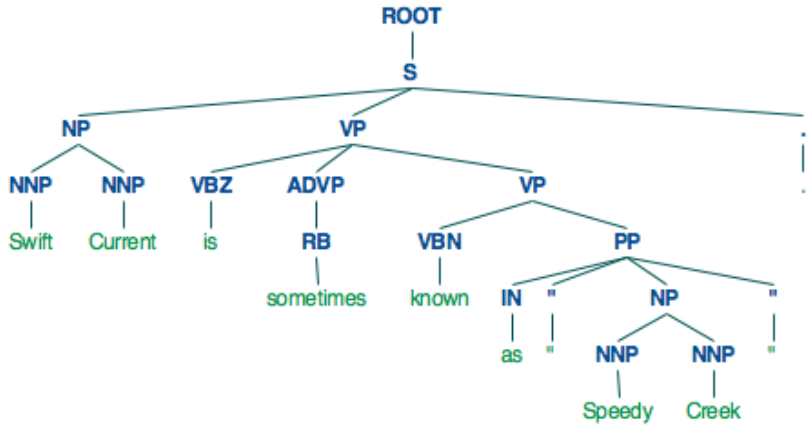


Fig. 1.2 An example of a tree obtained through a Phrase Structure Grammar

Our elected model chooses instead to focus on establishing one-to-one relations be-

tween words, accepting a PSG tree as input, and then proceeding to describe how words connect to one another thereby increasing and adjusting the meaning of the others.

Starting from a root node, which is always either the sentence's main verb or, in case of a copula, its complement, all the remaining words are inserted in a directed graph structure, connected through a labeled edge (denoting the dependency) to the most direct word their semantics refer to.

Figure 4.1 shows an example of the dependencies associated with the sentence:

Swift Current is sometimes known as "Speedy Creek".

We can see how *known* is found to be the sentence's root (root), whereas *is* is the auxiliary concurring to the passive clause (auxpass), *sometimes* is an adverbial modifier that changes the temporal meaning of *known* (advmod), *Current* introduces the passive nominal subject (nsubjpass), *Creek* does the same for the object of preposition *as* (prep_as) and both *Swift* and *Speedy* are nouns that act as meaning modifiers in a compound (nn). For a complete overview of all Stanford Typed Dependencies, see [8].

1.4 Features and architecture of the proposed system

The main features requested of the system are two:

A. Scan Wikipedia articles in order to collect patterns;

B. Scan text originating from any source looking for matches to the collected patterns and collect their corresponding RDF statements.

A subsystem, called Pattern Harvester, has been synthesised to achieve feature A: it accomplishes its job analysing one Wikipedia article at a time. It does this through the structured data available (in the form of RDF statements) for the entity subject of the article and the aid of an external parser, the Stanford Core Natural Language Processing tools[2] (SCNLP), that yields a Dependency Graph for each sentence in an article. The structured data are then converted ("worded") to their natural language representations so to seek them in sentences and, through the help of the SCNLP, every wording and every sentence are converted to dependency graphs that are compared to check if a relation described by a statement exists in the sentence. Once a relation pertaining to a particular predicate has been identified, it is generalised and stored as a pattern associated to that predicate.

A second subsystem, the Pattern Matcher, focuses on feature B. When it is fed a text, it annotates sentences just like the Pattern Harvester and then proceeds to seek in them occurrences of the previously stored patterns. Once a match has been found, it identifies the wordings corresponding to the subject and the object and then stores a new RDF statement composed by these wordings and the predicate associated to the matched pattern.

[2]Coupled with a Python wrapper [15]

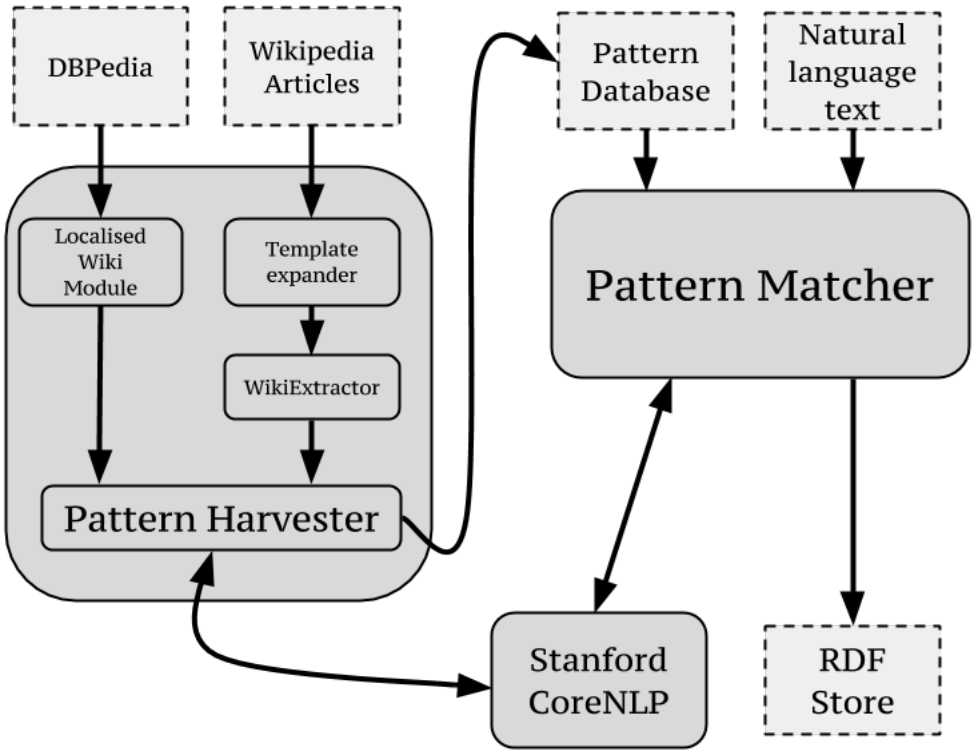


Fig. 1.3 Architecture of the system, comprising subsystems and modules

Chapter 2

Related work

For more than a decade now, efforts in text mining have seen a surge due to the increase in both the overwhelming availability of corpora in digital form and the advances in distributed computing and in processing power. The following is a review of the state-of-the-art methodologies reported in literature.

Very closely related to our problem and approach is IBMiner, a system described in [11] whose ultimate goal is to automatically fill in incomplete Infoboxes with data extracted from their containing articles' text. Its methodologies differ from ours though, as it makes use of sets of, as much as general, preset patterns, defined through both grammatical features and regular expressions with hard-coded English prepositions, effectively making it difficult to straightforwardly apply the system to a different language.

Another example is Wanderlust, detailed in [1], that uses a link grammar to find patterns and extract triple statements from free text, but lacks any connection whatsoever with a standard ontology such as DBPedia's, defining predicates instead as string literals, determined by a segment of the list of words interlinking subject and object in a sentence.

Whilst considering DBPedia not only as a training set but as a destination of the extracted triples as well, in accordance with our proposal, the Entity Extraction system introduced in [12] efficiently exploits PropBank [13], looking first for predicates and only then for entities, considered as arguments to the predicate. Unfortunately, the usage of PropBank restricts the system to English as it is the product of extensive human semantic annotation on a corpus, even though several efforts carry on to obtain similar annotations of predicates in corpora written in other languages.

More complex systems exist, such as KnowItAll. Presented in [9], KnowItAll works in an unsupervised fashion, i.e. it does not use a training set, but instead exploits a handful of domain-independent predefined patterns in combination with statistical algorithms that asses the validity of the extracted triples and, unlike our system, requires that all predicates of interest be specified before runtime.

Also, leveraging the power of search engines in the same fashion as KnowItAll's, the more recent TextRunner [3] achieves better precision and efficiency, without the need to specify beforehand predicates of interest and requiring less processing power for comparable results.

The system described throughout this document is to be considered a prototype implementation of a novel approach, not focused on short-term functionality but rather on the synthesis of new algorithms, still in their infancy and worthy of longer term research, not yet comparable to the full-fledged systems mentioned in this Chapter but in terms of

elected purposes and instruments.

Chapter 3

Sourcing and wording data

The following is an overview of how data is treated before pattern harvesting. As the system has been designed to be extremely easy to port for use with languages other than English, it includes an abstract class, named LocalisedWikiModule, that needs to be implemented with regards to the conventions used by the Wikipedia in the language of choice, so to address a number of issues arising when dealing both with wiki markup, typed data and natural language in general.

3.1 Wikipedia

The Wikimedia Foundation periodically distributes a dump of all the contents of Wikipedias of every language. These very large files in XML format need to be indexed before a particular article can be retrieved. For the task we use the Wikidump software [10], which accepts as input the article's title and then returns its raw source, which comprises HTML, wiki markup (a syntax used throughout Wikipedia for formatting text, generating hyperlinks and presenting data in a standardised fashion through templates) and the actual text. On our development machine, indexing the English Wikipedia's dump from March 4, 2013 took about 20 hours.

To strip out all unnecessary markup on these raw Wikipedia article sources, we use the WikiExtractor software [2], obtaining a plain text version of the article. Unfortunately the resulting text does not include an expansion for Wikipedia templates used throughout articles to display inline data[1], and they are simply discarded. Thus, before and after passing a raw article's text to WikiExtractor, the system performs some actions in order to support semantic analysis and preserve data presented through wiki markup.

Templates have been historically defined and updated for new functionality so to be retro-compatible, so no one of them presents a well-defined grammar that encompasses all its possible parameters and their disposition in a signature. For the lack of a better solution, which we are confident will come soon enough due to the increasing interest Wikipedia as a corpus is gaining in the research community, we chose to expand the most relevant templates before WikiExtractor's processing, designating the implementation of LocalisedWikiModule to carry out the task momentarily.

[1] An example of such a template is {{*convert*}}), that accepts as input numbers in various units of measure and then displays it in the corresponding imperial and metric units, allowing for a degree of rounding to be specified as a parameter as well.

After WikiExtractor's processing, we perform some more adjusting on the text, converting "Day Month" formats into "Month Day" and stripping out any left-over HTML tags. Finally, we have a plain text version of the article.

3.2 From DBPedia objects to natural language

As earlier mentioned, each article on Wikipedia has a injective relation with a URI in the `dbpedia` namespace. When the system initiates a scan for pattern harvesting on one of the former, it starts by retrieving from DBPedia all the statements presenting the latter as a subject. The set of retrieved statements X can be subsequently partitioned by predicate, yielding:

(α) Basic RDF properties in the `rdf` and `rdfs` namespaces, together with others defined in de-facto standards such as `foaf`;

(β) Properties in the `dbpprop` namespace;

(γ) Properties in the `dbpedia-owl` namespace, created by normalising the names of properties in β.

We choose to retain only partitions α and γ as their statements are provided in a standard fashion. First of all, their property names (predicates) are universally defined and do not allow for any misunderstanding or mutual complete overlapping in their semantics. Secondly, when the object consists of a numeric literal it is accompanied by an argument that specifies its data type. This allows us, for instance, to tell whether a float refers to a measure of area or of population density, and additionally what unit and unit system are used.

On the other hand, statements in β present property names inconsistent across different Infobox templates of origin and thus do not allow for their semantics to be precisely identified outside the accompanying text, making them unsuitable for the definition of a pattern. Moreover, numeric literals in β do not carry any information on their unit of measure and seeking them in text as they are would see the number of their false positive occurrences skyrocket.

Wording procedures Once the filter has been applied to X, the system proceeds to word each subject and object in a statement to a natural language string. The type of the value determines the transformation which will be applied:

- a string literal is retained as they are, discarding the statement if it is in a different language than the article's;
- for a URI we consider as its wording the string literals present in statements having the URI (or any of those whose corresponding Wikipedia page redirects to the page associated with that URI) as subject and `rdfs:label` as predicate. Considering `dbpedia:Los_Angeles` the whole set of wordings would consists in {'Los Angeles', 'City of Los Angeles, California', 'El Pueblo de Nuestra Señora la Reina de los Ángeles', 'City of L. A.', 'La-la Land', 'Los Angelos', ...} comprising historic names, nicknames and misspellings;

- when it is a numeric literal, its wording is obtained through a simple agent to be implemented for the language of interest. The system defines an abstract class for the purpose, named LocalisedWikiModule.

The following are some of the transformations from numeric data to natural language applied by the draft implementation of LocalisedWikiModule for the English Wikipedia, as empirically found to be congruous within several articles:

- dates are converted to a "Month Day, Year" format;
- floats are rounded so not have any decimal digits, in numbers of any kind a comma is inserted every three digits starting from the least significant;
- typed integers and floats, besides the previous transformations, are suffixed with the appropriate unit of measure.

Further refinements will necessarily include that the methods of the interface return multiple values corresponding to different magnitudes of rounding and multiple units of measure, in order to increase the probability of a match.

Chapter 4

Harvesting patterns

Before the system can proceed to extract new RDF statements from arbitrary text, it needs to go through a phase of supervised learning as described in this Chapter.

4.1 Methodology

The learning is carried out by the Pattern Harvester module after it is fed a list of Wikipedia titles, and the following procedure is performed for all of them.

The system proceeds to query DBPedia to fetch all RDF statements having the resource `dbpedia:`[*Article_Title*] as subject. It then starts to construct a set of natural language wordings referring to this resource ("subject wordings"): besides those contained as string literals in `rdfs:label` statements, this set will contain all the names of classes the resource belongs to, according to `rdf:type` statements. This is a preliminary approach to coreference resolution, letting the system look in text for references to the subject even when not explicitly called by name. Once this has been done, the Pattern Harvester executes the procedures detailed in Paragraph 3.2 on all the values for objects, yielding a set of wordings with an associated predicate ("object wordings").

Each wording in this set is treated as a sentence, thus parsed so to obtain its associated dependency graphs (DG).

At this point, the analysis starts on the article's plain text (as obtained through the procedures described in 3.1). First of all, it is split into a list of sentences through a pre-trained NLTK tokenizer [5].

Then, for every element in this list, the following is performed. Once a DG for the sentence is obtained, the set of its nodes is intersected with set of root nodes for subject wordings to verify that it is not empty. If that is the case, the system goes on to check whether the strings associated with these nodes in the graph are indeed the same as a subject wording. We define the string associated with a node in a dependency graph as follows: given a node, find in the graph all its successors such that the label L connecting them with the node is found in {noun compound modifier (nn), numeric modifier (num), element of compound number (number), adjectival modifier (amod)}; the string associated with the node is the segment of the original sentence that starts and ends with the first and last of these successors (in order of appearance from the left). If one or more wordings for the subject find such a match in the sentence, the same procedure is repeated for wordings of objects contained in the statements from DBPedia.

When the system has finally identified in the DG for the sentence a node for the subject and one for an object, it has found a pattern.

From the dependency graph we extract the graph spanning the shortest paths from the root node to the node associated with the subject and to the node associated with the object. This graph is generalised, considering the nodes associated with the wordings for subject and object as variables. Furthermore, an entity variable replaces every other node in the pattern whose associated string is a proper noun: considering the resource being the subject of the statement with `rdfs:label` as predicate and the proper noun as object, the pattern retains an association between the replacing entity variable and the resource's classes. Finally, every other node whose associated string starts with a digit is considered to be a numeric variable.

The system then proceeds to save this pattern, together with the subject and object wordings from the originating sentence, for subsequent retrieval from free text of RDF statements having `foaf:nick` as predicate.

4.2 A practical example

Now let us suppose we are to analyse the English Wikipedia article named for the Canadian city of Swift Current. DBPedia maps the article's name to `dbpedia:Swift_Current` and we are given, between others, the following two statements:

(`dbpedia:Swift_Current`, `rdfs:label`, "Swift Current")
(`dbpedia:Swift_Current`, `foaf:nick`, "Speedy Creek")

We proceed to obtain the DGs for "Swift Current" and "Speedy Creek": they will have "Current" and "Creek" as their root nodes, respectively. Going through the article, we encounter the sentence:

Swift Current is sometimes known as "Speedy Creek".

Its DG, shown in Figure 4.1, contains both "Current" and "Creek" as nodes, with their associated strings indeed matching the wordings for the subject and the object of the second statement above.

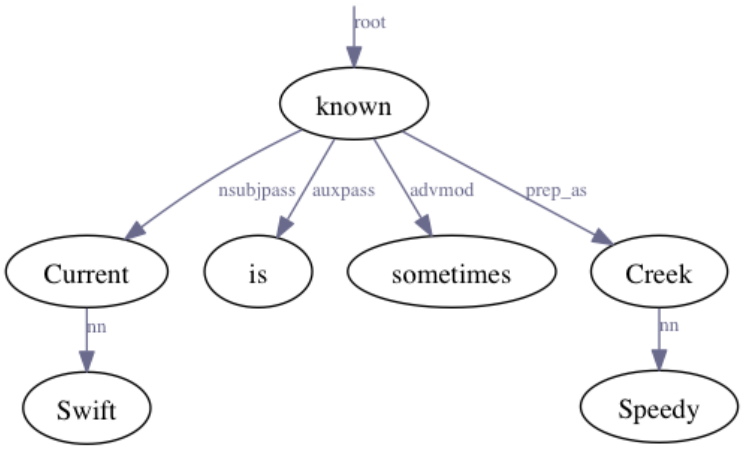


Fig. 4.1 A Stanford Dependency graph

We then proceed to extract the graph spanning the shortest paths to these nodes, shown in Figure 4.2a, and subsequently replace them with variables, yielding the pattern in Figure 4.2b.

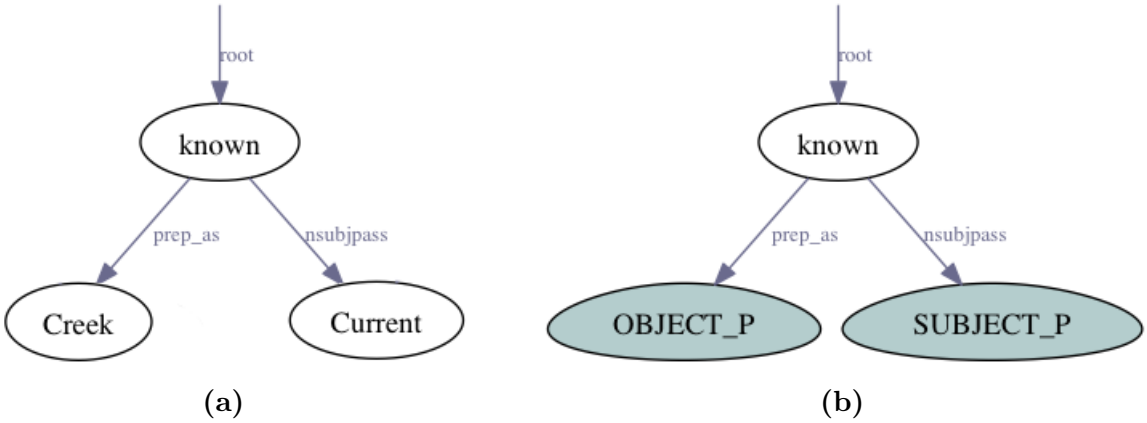


Fig. 4.2 Sequential transformations of a DG into a pattern for predicate `foaf:nick`

Chapter 5

Matching patterns

When the phase of supervised learning over a sufficient number of articles has been carried out, the system is ready to collect new RDF statements.

5.1 Methodology

When fed a text, the Pattern Matcher first splits it into sentences, in the same fashion as the Pattern Harvester does, and then iterates the following for each of them.

One by one, all pattern graphs are compared with the DG for the sentence. First, it checks whether there exists a path compatible with the shortest from the root node to the subject variable. If that is the case, it proceeds to do the same using the shortest path from the root node to the object variable. If once again one is found to be compatible with one in the DG for the sentence, the strings associated with the nodes at the end of these paths are collected and related through the pattern's predicate, yielding a new RDF statement.

Let us define a path X as a finite sequence of (node, edge label) pairs and x_i as the element of X in position i.

Two paths X and Y are defined compatible when they have the same length k and $\forall x_i \in X, y_j \in Y$ such that $i = j$, x and y are defined compatible.

Two elements of a path are defined compatible when their edge labels are the same and one of the following is true:

- one of them has a subject variable as node;
- one of them has a object variable as node;
- one of them has a entity variable as node and, considering the intersection I of the entity variable's classes with the classes of the resource that has the string associated with the other node as object in a `rdfs:label` statement, $I \neq \emptyset$;
- one of them has a numeric variable as node, and the other is found to be numeric as well;
- their nodes have the exact same value.

.

At the moment the system does not have any employ coreference resolution to identify the resources subject or object of a statement. This results in statements presenting string literals for both S and P components, obviously violating the RDF's requirements. The complexity of this subproblem has led us to postpone its addressing to future research so to focus our preliminary efforts on pattern recognition, as it does not hinder our ability to assess the relevancy of extracted statements.

5.2 A practical example

Suppose, during the analysis of a text, we encounter the following sentence:

> *In the 19th century, the district was known as Hacienda Tala to settlers who cultivated the land and considered it a star ("tala") from heaven.*

If a pattern such as the one shown in Figure 4.2b has been previously harvested, the Pattern Matcher can proceed to compare it with the DG obtained for the sentence, shown in Figure 5.1.

The pattern's shortest path to the subject variable is [("*known*", "*nsubjpass*")], which is found to be compatible with a path in the DG that ends in node "district".

The pattern's shortest path to the object variable is [("*known*", "*prep_as*")], which is found to be compatible with a path in the DG that ends in node "Tala".

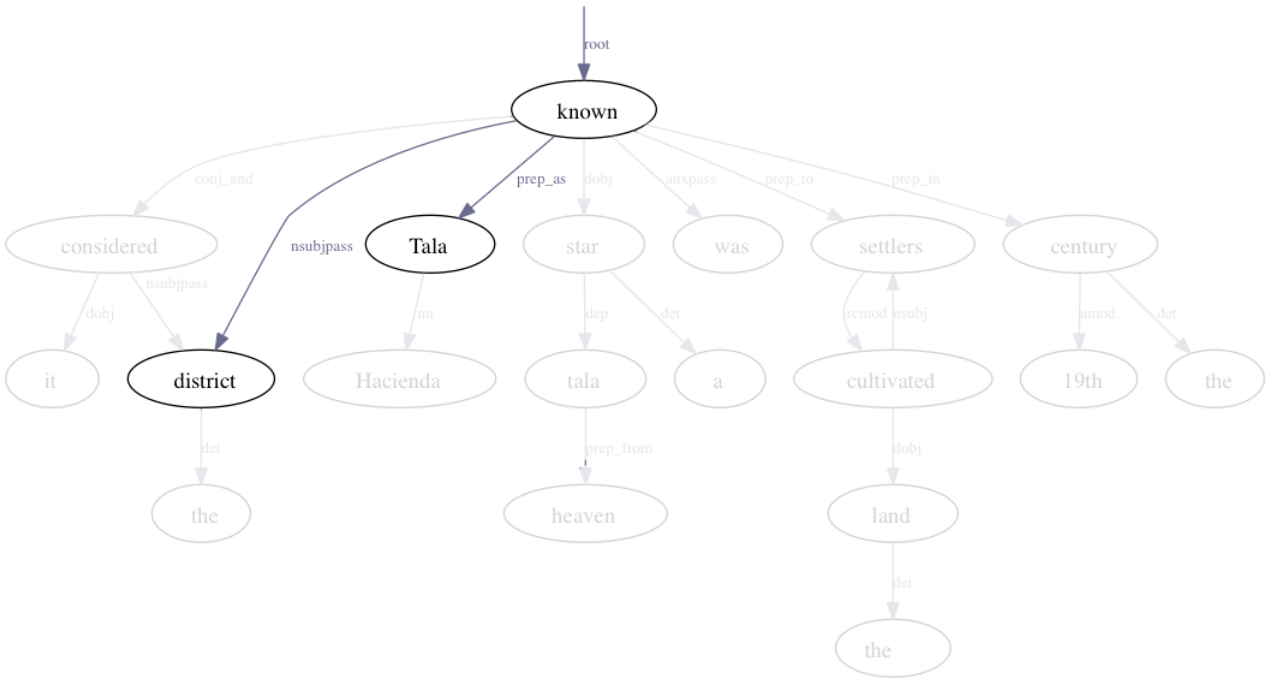


Fig. 5.1 A dependency graph. The opaque parts indicate where the pattern found a match.

A new statement is thus formed, carrying the strings associated with these nodes as subject and object, respectively, together with the pattern's predicate, yielding:

("district", `foaf:nick`, "Hacienda Tala")

Chapter 6

Experimental results

The system has been tested so to assess its effectiveness in the RDF extraction process. In the following, we use the standard metrics of precision and recall, computed as the relevant percentage of retrieved statements and the retrieved percentage of relevant statements, respectively. The definition of relevant here refers to all the human-retrievable statements having as predicate one of those included by statements in partitions α and γ (as detailed in Paragraph 3.2) of all DBPedia statements obtained during the supervised learning phase.

6.1 Classification of errors

Incompatible Localised Range (ILR) The class of the extracted object and the one required by the predicate each have a relation with the subject. The semantics of these relations substantially but not completely overlap, thus making the statement partially relevant but definitely improper. Consider the case of `dbpedia:chanchellor` and `dbpedia-owl:rector`: the first applies to chief executives of universities in several English-speaking countries, whereas the other is used most often throughout Europe;

Metonymic Error (ME) The class of the extracted object and the one required by the predicate are somehow related, but the predicate requires that the former be either a part of, a composition of one or more of or a sibling of the latter. This results from the common usage of the same constructs to state the same kind of relation between a subject and different classes of objects. Consider the case of `dbpedia-owl:employer` that requires the object belong to class `dbpedia-owl:Organisation`: the natural language instances of the statements having the former as predicate present themselves as phrases containing "hired by", which language allows to be followed by a person's name as well, tricking the system into extracting a non completely relevant statement.

Truth Error (TE) All other statements whose semantics are well-defined and valid but do not apply to reality.

Ill-Defined Semantics (IDS) All other statements.

6.2 Assessment of results

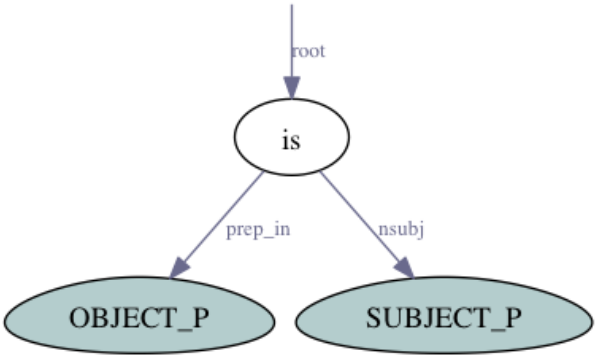


Fig. 6.1 A very generic pattern.

The learning phase has been carried out through the analysis of the 53,893 articles on human settlements curated by the WikiProject Cities from the English Wikipedia as of February 2013. The project includes lists of articles and various meta-pages (in Wikipedia namespaces such as "Category:") to aid human navigation and categorise content, and as such they are not included in this count. On our humble development machine (detailed in Appendix A) it took about 20 days for the Pattern Harvester to process all these articles and their corresponding statements in DBPedia. The system managed to identify on average 7.83 object wordings and extract 0.27 unique patterns per article, totalling 14,328 patterns for the whole corpus. This low number is to be primarily imputed to the lack of a proper coreference resolution module which hinders the ability to identify a subject wording for every identified object wording.

The assessment of RDF statement extraction has been carried out through 200 sentences randomly sampled from the aforementioned articles, and the following results exclude the trivial statements, obtained by matching a pattern with the DG of its originating sentence, from the set of retrieved ones.

A preliminary test, in which we excluded any patterns deemed to be insufficiently specific as their nodes were only an object and a subject variable, resulted in the extraction of 249 statements, with a precision of 0.26 and a recall of 0.26, matching 129 unique patterns.

The set of not relevant statements has been broken down to the following: 60% IDS, 16% ILR, 18% ME and 6% TE.

Deeper analysis of erroneously matched patterns highlights how the high rate of Ill-Defined Semantics and Metonymic error statements finds its cause in common usage of the same generic constructs to denote a variety of specific relations, across several combinations of different domains and ranges, yielding in too generic patterns.

Consider the sentence:

Amarillo is in the U.S.

Encountered in the learning phase, it allowed the system to harvest the pattern shown in Figure 6.1 and associate it with predicate `dbpedia-owl:country`.

Subsequently, when the Matcher encountered in the sample:

The Town of Hammond is in the northwest corner of St. Lawrence County.

It extracted the triple:

("Town", `dbpedia-owl:country`, "northwest corner")

that must be discarded as Ill-Defined Semantics, as it is too distant from the semantics defined by `dbpedia-owl:country`, which require its range to be of class `dbpedia-owl:Country`.

Further developments of the methodology will necessarily have to focus on this problem and its possible solutions.

Chapter 7

Conclusions

We have shown our draft methodology, inspired by current theories on the importance of lexical chunks in language acquisition in humans, and its process of supervised learning of natural language patterns, defined in terms of Stanford Typed Dependencies, on a training set sourced from Wikipedia and the DBPedia project, aimed at subsequent automatic extraction of RDF statements from free text. Experimental pattern matching on a random sample of the same corpus show a high recall rate of 0.26 and a low precision rate of 0.26, encouraging further research on the problem with a focus on the combinations allowed by predicates for their domains and ranges. As attested by much of the recently published work in Information Retrieval, a tradeoff between these two rates of assessment has been established as a necessary step towards better performance in similar systems, and our preliminary results provide a good margin of action for future elaboration of such a compromise.

7.1 Future work

Shorter-term goals in our future research will necessarily focus on acceptable levels of precision: one proposal that appears to be worthwhile is the rejection of patterns identical to others associated with less specific predicates. Also, additional tests will focus on the suitability of the system for cross-domain text analysis.

Once the system is deemed to be sufficiently precise, we would be able to implement an adaptation of the Dual Iterative Pattern Relation Extraction technique, described in [6], so to exploit the duality of patterns and predicates and look for new patterns through natural language occurrences of statements obtained from free text.

To make the system suitable for usage by larger ones, we will also need to investigate possible ways for coreference resolution, so to be able to harvest more patterns and extract more triples.

Furthermore, the impact of effective natural language computer processing can only be underestimated. A full-fledged version of the proposed system would be beneficial to various subfields in Information Retrieval.

Machine translation would certainly benefit from the availability of patterns classified through universal predicates, denoting a particular relation between two entities, that would allow for sentences to be translated in the most semantically accurate way from a language to another using a predicate as a reference for semantics.

Question answering systems would be able to compare existing patterns with DGs

obtained from interrogative sentences, provided by users in their native language, and understand what information the user is asking for.

Connecting the system to a news feed (e.g. CNN's Twitter account) could be a way to collect new information so to have it available almost immediately for consumption by financial algorithmic trading systems.

Looking at all these possibilities, we consider additional work on the matter to be worthy of consideration.

Appendix A

Development details

The development and testing of the system have been carried out through a machine running with a 1.8 Ghz i5 CPU, 4GB of RAM, a Solid State Drive. The prototype implementation was written in Python 2.7, consisting in about 2800 lines of code. An approximate temporal evaluation of the work suggests about 400 man-hours were employed for research and development.

Appendix B

URI prefixes and namespaces

The following is the list of the URI prefixes, and their corresponding namespaces, mentioned in this document.

rdf `http://www.w3.org/1999/02/22-rdf-syntax-ns#`

dbpedia `http://dbpedia.org/resource/`

dbpprop `http://dbpedia.org/property/`

dbpedia-owl `http://dbpedia.org/ontology/`

foaf `http://xmlns.com/foaf/0.1/`

Bibliography

[1] Akbik, A. and Broß, J., "Wanderlust: Extracting Semantic Relations from Natural Language Text Using Dependency Grammar Patterns" in: Proceedings of the Workshop on Semantic Search (SemSearch 2009) at the 18th Int. World Wide Web Conference (WWW 2009), Madrid, 2009.

[2] Attardi, G. and Fuschetto, A., "Wikipedia Extractor", `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`, last accessed: August 2nd, 2013.

[3] Banko, M., et al., "Open Information Extraction from the Web", Communications of the ACM, 51(12), 2008, pp. 68–74.

[4] Berners-Lee, T., et al., "The Semantic Web", Scientific American, 2001, pp. 29-37.

[5] Bird, S., et al., *Natural Language Processing with Python*, O'Reilly Media, 2009.

[6] Brin, S., "Extracting Patterns and Relations from the World Wide Web" in The World Wide Web and Databases, Springer, 1999, pp. 172–183.

[7] De Marneffe, M.-C., et al., "Generating Typed Dependency Parses from Phrase Structure Parses" in: Proceedings of the Fifth International Conference on Language Resources and Evaluation Conference, Genoa, 2006.

[8] De Marneffe, M.-C. and Manning, C.D., "Stanford Typed Dependencies manual", `http://nlp.stanford.edu/software/dependencies_manual.pdf`, last accessed: September 3rd, 2013.

[9] Etzioni, O., et al., "Unsupervised Named-Entity Extraction from the Web: An Experimental Study", Artificial Intelligence Journal, 165(1), 2005, pp. 91–134.

[10] Lui, M., "WikiDump", `https://github.com/saffsd/wikidump`, last accessed: August 2nd, 2013.

[11] Mousavi, H., et al., "Deducing InfoBoxes from Unstructured Text in Wikipedia Pages", Computer Science Dep. Technical Report #130013, University of California, Los Angeles, 2013.

[12] Exner, P. and Nugues, P., "Entity Extraction: From Unstructured Text to DBpedia RDF triples" in: Proceedings of 1st International Workshop on The Web of Linked Entities, 2012.

[13] Palmer, M., et al., "The Proposition Bank: An Annotated Corpus of Semantic Roles", Computational Linguistics Journal, 31(1), 2005, pp. 71-106.

[14] Schimmt, N., "Lexical Chunks", English Language Teaching Journal, 54(4), 2000, pp. 400–401.

[15] Smith, D., et al., "A Python Wrapper for the Java Stanford Core NLP tools", `https://bitbucket.org/torotoki/corenlp-python`, last accessed: August 2nd, 2013.

[16] Wikipedia, "Swift Creek", `http://en.wikipedia.org/wiki/Swift_Creek`, last accessed: August 2nd, 2013.

Ringraziamenti Vorrei ringraziare: il mio relatore Maurizio Atzori per avermi dato la possibilità di confrontarmi con una sfida così interessante e innovativa; i miei genitori, che ci sono sempre stati, anche nei momenti più difficili; Silvia, Renato e i miei amici, perché mi ricordano costantemente che il mondo è fatto di gente assolutamente brillante; i miei compagni di studio, per le infinite ore spese assieme in biblioteca (e al quinto piano).