Einstein: Physicist or Vegetarian? Summarizing Semantic Type Graphs for Knowledge Discovery

Tomasz Tylenda
Max-Planck Institute for Informatics
Saarbruecken, Germany
ttylenda@mpi-inf.mpg.de

Mauro Sozio
Max-Planck Institute for Informatics
Saarbruecken, Germany
msozio@mpi-inf.mpg.de

Gerhard Weikum
Max-Planck Institute for Informatics
Saarbruecken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

The Web and, in particular, knowledge-sharing communities such as Wikipedia contain a huge amount of information encompassing disparate and diverse fields. Knowledge bases such as DBpedia or Yago represent the data in a concise and more structured way bearing the potential of bringing database tools to Web Search. The wealth of data, however, poses the challenge of how to retrieve important and valuable information, which is often intertwined with trivial and less important details. This calls for an efficient and automatic summarization method.

In this demonstration proposal, we consider the novel problem of summarizing the information related to a given entity, like a person or an organization. To this end, we utilize the rich type graph that knowledge bases provide for each entity, and define the problem of selecting the best cost-restricted subset of types as summary with good coverage of salient properties.

We propose a demonstration of our system which allows the user to specify the entity to summarize, an upper bound on the cost of the resulting summary, as well as to browse the knowledge base in a more simple and intuitive manner.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.3.1 [**Content Analysis and Indexing**]: Abstracting methods

General Terms

Algorithms

Keywords

summarization, knowledge bases, semantic search

1. INTRODUCTION

1.1 Motivation

Knowledge-sharing communities such as Wikipedia represent a huge and surprisingly reliable source of information in a wide variety of fields. Knowledge bases such as DBpedia[1], YAGO[10], or Freebase[5] are a concise, formal representation of (specific pieces from) such encyclopedic sources. This opens up ways of using structured query languages for knowledge discovery. By extracting or tagging entities and their attributes in Web pages and linking them to corresponding facts in a background knowledge base, this can be further leveraged for semantic search on the Web.

Search engines for structured knowledge (in knowledge bases or gathered from the live Web), such as Entity Cube [3], Google Squared [6], Wolfram Alpha [7], sig.ma [4], or NAGA [8], tend to either give very brief answers, merely listing entity names, or overwhelm the user with the full set of facts that they find in their underlying repositories. For example, when you ask for "Swiss people", some of the above engines merely return a list of names. The user can click on each name to see more facts about the person, including Web pages that contain the entity, but this is a tedious way for knowledge discovery. Other engines show all – often hundreds of – facts about all Swiss people that DBpedia, Freebase, and other linked-data sources offer; this is a cognitive overload for most users. What we need instead is a kind of *semantic snippet* per result entity, highlighting the most salient facts about each but avoiding trivial or exotic information. For example, for the Swiss-people result Albert Einstein, we may want to see a compact summary saying that he was a scientist, won the Nobel Prize, was born in Germany (but grew up in Switzerland), graduated at the University of Zurich, and later was a professor at Humboldt University and even later at Princeton.

Google Squared does return attribute-structured records as answers to keyword queries – an adequate result granularity. However, the attributes are the same for each entity in the result set. For the Swiss-people query, Einstein is treated the same way as Roger Federer (a Tennis player): the presented attributes are fairly generic properties like birth date, birth place, death date, and death place.

1.2 Problem Statement

Explicit knowledge bases have very rich type systems, partly inferred from Wikipedia categories, the WordNet taxonomy, and other sources of this kind. For example, Yago knows 38 semantic classes to which Einstein belongs, and these have more than 50 superclasses in the Yago type system. Judiciously choosing a small subset of these could return more or less the ideal summary outlined above. Note that type/category names can often be viewed as encoding attribute values, such as class `NobelPrizeWinners` denoting the fact `hasWon: NobelPrize`. Conversely, we can see groups of entities with the same value for some interesting attribute as a semantic type/class, such as `bornIn: Germany` defining

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0637-9/11/03.

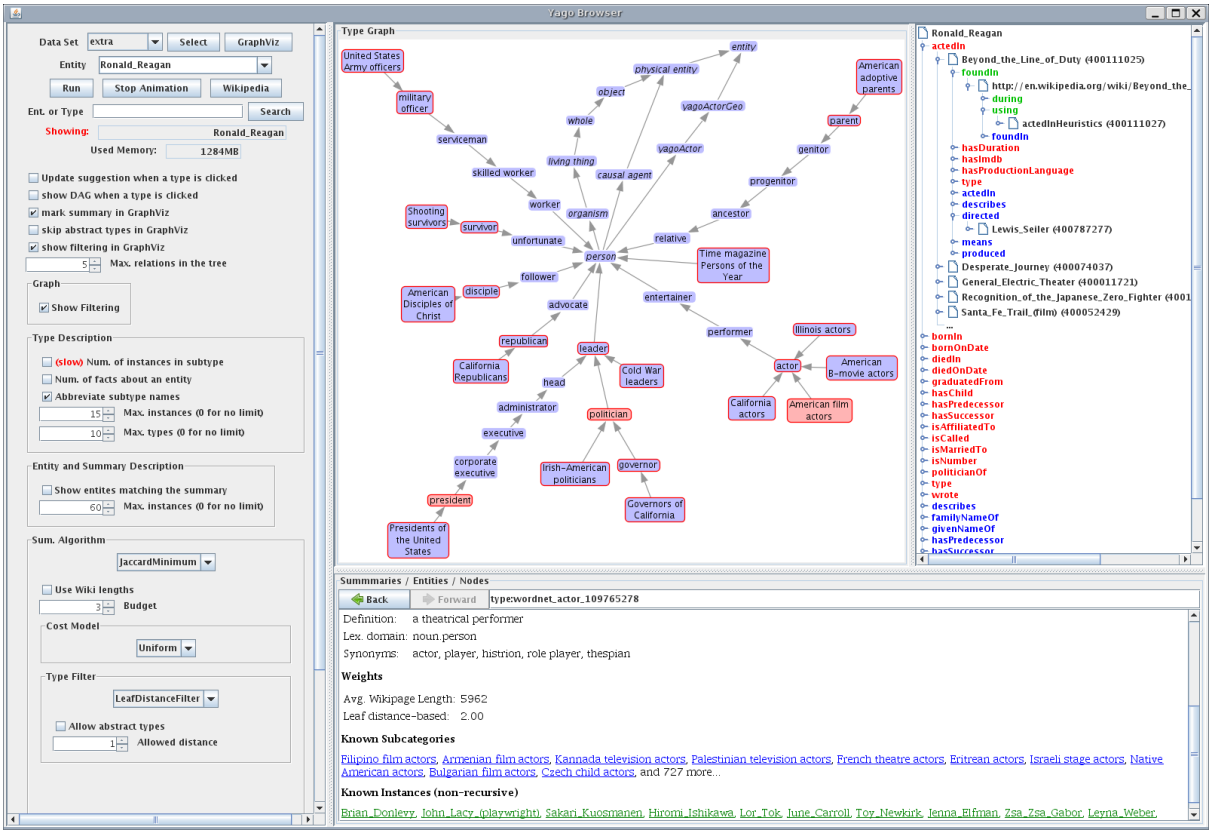

Figure 1: Yago type graph for Ronald Reagan shown in the browser. 1) controls of the browser and summarization algorithms, 2) type DAG, 3) relation view, 4) additional information about types.

the type `peopleBornInGermany`. Because of this duality, we restrict our approach for generating semantic snippets to selecting appropriate types from the rich repertoire available in the knowledge base.

In most search- and exploration-related situations, a good summary of an entity would contain between three and around ten properties. These could then be refined by explicit user interactions on specific subtype dimensions when needed. However, Yago and DBpedia have an order-of-magnitude larger number of types/classes to choose from. This is illustrated by the type graph for Ronald Reagan, as shown in a Yago browsing tool in Fig. 1. Note that the type graph of an entity is usually not a tree, but forms a directed acyclic graph (DAG), with subtype-supertype edges and a generic root type `entity`. For example, the type `mayor` has two supertypes: `politician` and `civil authority`, which converge in `person`.

Now we can define the problem addressed in this paper. Given the type DAG of an entity and a desired number of output types, select the most informative types. These will then be used to generate semantic snippets for semantic-search results.

The rest of the paper is organized as follows. In Section 2 we describe our summarization methods, while our system is demonstrated in Section 3.

2. OUR SYSTEM

We start by giving a short description of how knowledge bases are organized and in particular of Yago, which is the one we shall use for our purposes.

Arg1	Relation	Arg2
Albert Einstein	bornIn	Ulm
Albert Einstein	hasAcademicAdvisor	Alfred Kleiner
Albert Einstein	type	Swiss physicists
Swiss physicists	subClassOf	physicist
physicist	subClassOf	scientist

Table 1: An excerpt from YAGO. We are interested in `type` and `subClassOf` relations.

2.1 The Knowledge Base

YAGO contains facts extracted from semi-structured parts of Wikipedia (infoboxes and categories) and WordNet[2]. A small excerpt is presented in Table 1. The named objects in YAGO can be classified into *entities* which represent persons, locations, organizations, etc., e.g. `Albert Einstein` and `Univ. of Zurich` and *types* which form a categorization of entities, e.g. `physicist` and `Swiss physicists`.

We are particularly interested in the relations `type` and `subClassOf`. The former is defined between entities and types, and the latter between types. Using the `type` and `subClassOf` relations we can build a direct acyclic graph representing all types of an entity. Fig. 2 presents such a DAG for Aimé Argand. The `subClassOf` relation is transitive ($a \rightarrow b \wedge b \rightarrow c \Rightarrow a \rightarrow c$), but for clarity we do not show transitive edges in type DAGs.

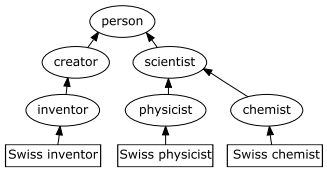

Figure 2: Type DAG of Aimé Argand. Rectangular nodes correspond to the wikicategories, oval nodes correspond to WordNet synsets. For simplicity some nodes were removed.

2.2 Toward a good summarization method

We compile the following list of desirable properties that any good summary should have.

Conciseness. A good summary should not be too long. Depending on how the summary is shown to the user, we consider two different measures of its size. If the types in the summary are presented in a list with each item occupying a single line, then we assign a unit cost to every type and enforce a simple cardinality constraint on the number of types that might be selected. If the types are presented as a list separated with commas, then the cost of each type is defined to be equal to its length in characters.

Importance. There are types which carry valuable information about an entity, e.g. the type `physicist` is crucial when we talk about Albert Einstein. On the contrary, other types describe non-salient facts (`left-handed person`, `vegetarian`). Since our summaries are meant to provide users with the most significant facts, our method should favor the former kind of types.

Granularity. General types, which describe large categories of entities, e.g. `person` or `location` should not be included in the summary. Similarly, too specific types which describe a very small number entities, e.g. `physicist born in 19th century in Zug` should not be included either.

Diversity. A summary should cover all aspects of an entity, e.g. {`member of the parliament`, `prime minister`} is worse than {`politician`, `physicist`}, because it focuses only on the political career of the described person.

A natural summarization method could serve the user with the top-ranked types according to some appropriate weighting function. However, we argue that this method would not deliver any good solution, as illustrated by the following example.

Consider the task of summarizing the type graph in Fig. 2, while having only a limited budget on the number of types that can be selected. Summaries that we find to be the best for cardinalities 1 to 3 are: $S_1 = \{scientist\}, S_2 = \{scientist, inventor\}, S_3 = \{inventor, physicist, chemist\}$.

If the budget is only one type, we prefer to use the more "informative" `scientist` rather than "too general" `person`. When the budget is increased to two types, we expand the summary with the type `inventor`. However, when our budget consists of three types it is better to substitute `scientist` with `physicist` and `chemist`. In general, when the budget is increased we do not just add more types, but choose different types or "break" the types (`scientist` is broken into `physicist` and `chemist`). Hence, simply returning the top-ranked types to the user would not give satisfactory results.

2.3 Algorithms

In this section we discuss two methods to determine the importance of a type, then we describe the problem of type with missing argument and finally we present two algorithms for summarization.

Some of our summarization methods require that we assign positive weights to types, which reflect how well they describe the summarized entity (good nodes have high value). Our system uses two summarization algorithms, which have slightly different requirements, therefore we devised two importance measures.

The first method exploits the fact that the entities in YAGO were extracted from Wikipedia and therefore we can easily obtain the lengths of their articles (in bytes). The importance of a type is the average article length of its members. Since it is calculated independently of the entity it describes, `physicist` has equal weights for Albert Einstein and Brian May, who is better known as a member of a British rock band Queen. Interestingly, some non-crucial types, such as `vegetarian` or `left-handed person` can have large weights. We hypothesize, that the reason is as follows: famous people are be added to all possible categories, whereas not-so-famous ones are only added to the most important categories.

The second summarization algorithm requires that weights satisfy the following properties: *i*) the types low in the hierarchy should have large weights as they are more precise, more abstract types should be assigned low weights, *ii*) if a type has multiple children (in a type DAG of a single entity) its weight should be amplified, e.g. in Fig. 2 the weight of *scientist* should be boosted, because it has two children, which shows that it is more important for the entity, *iii*) a single parameter should control the trade-off between choosing types low in the hierarchy and the ones which are high.

We developed a method which is based on a random walk on the type DAG. The walk starts at a leaf with probability proportional to the average length of its Wikipedia articles (the first weighting method). At each node we *i*) either restart the walk (jump to a leaf) with probability α (1 in sink nodes), *ii*) or, with probability $(1 - \alpha)$, we follow one of the outgoing links of the node.

In addition to finding too general, too specific or unimportant types we also need to recognize types which lack an argument, e.g. `citizen` versus `citizen of Germany`, or `member` versus `member of the European Union`. The problem was already discussed in [9], but their results are applicable mostly to Japanese. In our system we use a simple occurrence statistic based method to find types with missing arguments. We calculated the total number of occurrences of the noun, and the number of occurrences with an argument pattern like: *<noun> of, 's <noun>, (his|her|their|its) <noun>* The ratio of two values is used to decide whether an argument is needed.

Our system implements two main algorithms for summarization. The first one uses the fact that a type is a set of entities. Given an entity with the type set $\mathcal{T}$ we choose a small subset of $\mathcal{T}$, such that: *i*) its intersection is small, *ii*) the types are of medium cardinality. The first property gives us the diversity of the summary, e.g. if a person is both a musician and scientist, we will choose types which describe each aspect of their career. The second one ensures that the types we choose are not too general (e.g. `person`) or too specific (e.g. `German quantum physicist`). The method does not have any tunable parameters.

The second algorithm works on the type DAG of an entity.

Nodes are assigned weights which should reflect how well a type describes an entity. The summarization algorithm chooses a subset of types of an entity, such that: *i*) the set is smaller than the budget, *ii*) the sum of weights is maximized, *iii*) no two types are connected by a directed path. The last requirement ensures that we do not choose redundant types, e.g. `quantum physicist` and `scientist`. The algorithm itself does not have any tunable parameters, but the weighting method that we used has one parameter which governs the balance between general and specific types. Hence we are able to tune the method to choose more general or specific types.

In addition to the algorithms presented above, our system can exploit simple heuristics to determine which types are suitable for summaries. One of such heuristics uses a short list of abstract types, such as `entity, living thing, object`, and removes them before we run the summarization algorithm.

3. DEMONSTRATION

The demonstration shows three contributions of our work. Firstly, we developed a new browser for the YAGO database. Secondly, we present the entity relation graph in a tree, which offers intuitive navigation, easily expands and collapses parts of the graph and uses little screen space. Finally, we implemented the summarization algorithms within the browser, which facilitates interactive evaluation of the summarization algorithms.

3.1 Yago Browser

The main window of the browser is shown in Fig. 1. In the beginning the user has to choose an entity by selecting it from a list, entering its unique identifier or by using the built-in search box. The application shows the type DAG of the entity (in the center of the screen), its relations (tree on the left) and some additional information (bottom).

Let us look at the relation view in the example (Fig. 1). The top node is the entity selected by the user. Its direct children are the relations and the next level contains their arguments, e.g. `R. Reagan actedIn Beyond the Line of Duty`. The structure is recursive, that is the children of `Beyond...` are the relations, which have it as an argument. All entities and values are show in black. The relations can be red, blue or green. Red relations should be read top-down, e.g. `R. Reagan actedIn Beyond...`, blue relations have the opposite direction, e.g. `Lewis Seiler directed Beyond...`. Green relations describe metafacts (facts about facts). Each fact in YAGO has a unique identifier, e.g. `400111025: R. Reagan actedIn Beyond...`. The metafacts use these ids to refer to facts they describe, e.g. `400111025 foundIn en.wikipedia.org/wiki/Ronald_Reagan`.

The way in which we present the relations has several advantages over graph views, similar to the one used for type DAGs. Firstly, it is easy to implement in various programming environments, since it uses only the standard tree widget. Moreover, the users find it easier to use standard interface than a custom graph view. The second advantage is that tree views efficiently use screen space and allow the user to easily expand nodes and collapse non-interesting parts of the graph.

The bottom panel of the browser initially presents information about the selected entity. When the user clicks on a type in the central panel, more detailed information about this type is presented, it contains the number of sub- and supertypes, the number of instances, sample subclasses and instances, and in case of types derived from WordNet also the definition and synonyms.

3.2 Summarization

The browser shows the results of summarization of an entity on the type DAG. The panel on the left controls which summarization algorithm is used. The user can switch between no summarization, the set-based algorithm and the graph based algorithm. The interface allows to select a cost model based on the number of types or their lengths, as well as some additional parameters, e.g. the trade-off between specific and general types in the second algorithm. A sample summary is shown in Fig.1. The nodes with the red outline were selected in the preprocessing phase and passed to the summarization algorithm. Subsequently, the summarization algorithm chose the summary, which consists of the red nodes.

In the example in the Fig.1 we requested a summary of the length at most 3 types. The filtering stage allows only the nodes which are leaves or parents of leaves. Eventually, the summarization algorithm chose three types: `president`, `politician` and `American film actors`. Clearly, the summary includes the most important types – `president` and `politician`. It is diverse, because it mentions that R. Reagan was an actor, which is a valuable fact, as not many politicians are also actors. Additionally, we avoided selecting uninteresting types like `American adoptive parents` and too general ones like `leader`.

4. REFERENCES

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: a nucleus for a web of open data. In *ISWC'07/ASWC'07: Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] Christiane Fellbaum, editor. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May 1998.

[3] http://entitycube.research.microsoft.com/index.aspx.

[4] http://sig.ma/.

[5] http://www.freebase.com/.

[6] http://www.google.com/squared.

[7] http://www.wolframalpha.com/.

[8] Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, and Gerhard Weikum. NAGA: Searching and Ranking Knowledge. In *24th International Conference on Data Engineering (ICDE 2008)*. IEEE, 2008.

[9] Kow Kuroda, Masaki Murata, and Kentaro Torisawa. When nouns need co-arguments: A case study of semantically unsaturated nouns. In *Proceedings of the 5th International Conference on Generative Approaches to the Lexicon*, September 2009.

[10] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.