

Repeatable Web Data Extraction and Interlinking

M. Kopecky¹, M. Vomlelova², P. Vojtas¹

Faculty of Mathematics and Physics
Charles University
Malostranske namesti 25,
Prague, Czech Republic

¹{kopecky|vojtas}@ksi.mff.cuni.cz, ²marta@ktiml.mff.cuni.cz

Abstract. We would like to make all the web content usable in the same way as it is in 5 star Linked (Open) Data. We face several challenges. Either there are no LODs in the domain of interest or the data project is no longer maintained or even something is broken (links, SPARQL endpoint etc.).

We propose a dynamic logic extension of the semantic model. Data could bear also information about their creation process. We calculate this on several movie datasets.

In this work in progress we provide some preference learning experiments over extracted and integrated data.

Keywords

Repeatable experiments; web data extraction, annotation, linking; dynamic logic; preference learning

1 Introduction, Motivation, Recent Work

For our decisions we often need automated processing of integrated web data. Linked (open) data are one possibility to achieve this vision. Still, there are some challenges.

Production URLs are sometimes subjects of change ([A]). Data migrate, data project run out of contracted sustainability period and so on.

SPARQL endpoints are expensive for the server, and not always available for all datasets. Downloadable dumps are expensive for clients, and do not allow live querying on the Web ([V+]).

In some areas there are no corresponding Linked data projects available at all. Imagine e.g. a customer looking for a car. He or she would like to aggregate all web data. Our idea is to remember previous successful extractions in given domain and use this in the current situation. For evaluation of previous extractions can help also social networks. We have presented this idea first in [PLEDFV]. We concentrated on one specific purpose – extract object attributes and use of these data in recommender systems. In this research we have tried to contribute to increase the degree of automation of web content processing. We presented several methods for mining web information and assisted annotations.

1.1 Semantic Annotator for (X)HTML

A tool for assisted annotation is available in [F2]. *Semantic Annotator* allows both manual and assisted

annotation of web pages directly in Google Chrome. It requires no complicated installation and is available on all platforms and devices where it is possible to install Google Chrome. Semantic annotation is available to all current users of the Internet not only to authors' site. Browser extension *Semantic Annotator* began as a prototype implementation in the Thesis [F1].

Google Chrome extension *Semantic Annotator* is used for manual semantic annotation of Web sites. The goal of semantic annotation is to assign meaning to each part of the page. The significance of real-world objects and their properties and relationships are described in dictionaries – either self-created or imported. Then the annotation process consists of selecting parts of the site and the assignment of meaning from the dictionary.

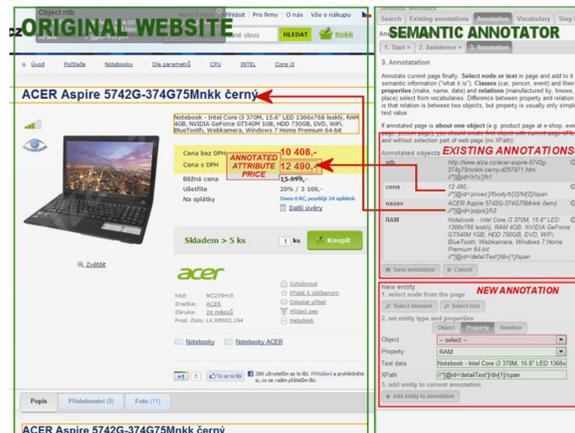


Figure 1: Illustrative figure for Semantic annotator, more in [F1], [F2]

In Figure 1 we show an example of annotation of product pages in e-shop. The user selects a web page and product name from the dictionary describing products that assigns a name meaning "product name". Then on the same page the user selects a product price and gives it the meaning of "price". Because of similarity of pages and usage of templates, annotating few pages enables to train an annotator for the whole site. Consequently, search of annotated website is then much more accurate.

1.2 Semantic Annotations for Texts

In previous section we have described a tool for annotation of (X)HTML pages. These are useful for extraction of attributes of products on pages created by the

same template even if texts are a little bit different. Even if there are no templates, in a fixed domain like reports on traffic accidents, there are still repetitions.

A tool for annotation of text was developed in our group in PhD thesis [D1]. The tool is available under [D2]. It is built on GATE [BTMC], using UFAL dependency parsing [UFAL] and Inductive Logic Programming extension. It represents a tool for information extraction and consequent annotation.

In the thesis [D1] are presented four relatively separate topics. Each topic represents one particular aspect of the information extraction discipline. The first two topics are focused on new information extraction methods based on deep language parsing in combination with manually designed extraction rules. The second topic deals with a method for automated induction of extraction rules using inductive logic programming. The third topic of the thesis combines information extraction with rule based reasoning. The core extraction method was experimentally reimplemented using semantic web technologies, which allows saving the extraction rules in so called shareable extraction ontologies that are not dependent on the original extraction tool. See Fig.2. on traffic accident data.

The last topic of the thesis deals with document classification and fuzzy logic. The possibility of using information obtained by information extraction techniques to document classification is examined. For more see also [PLEDFV].

In this research proposal we concentrate on synergy effect of annotation and integration of data for user preference learning, and consequently for recommendation.

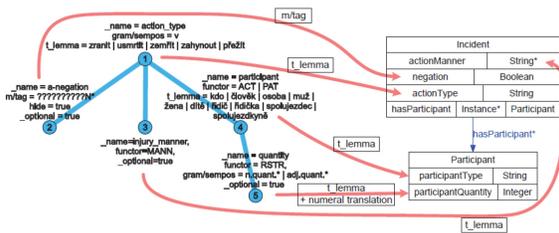


Figure 2: Illustrative figure of an extraction pattern for dependency tree, more in [D1], [D2].

It turns out that similarity and dynamic aspects of web data play a role here as well. We propose an appropriate *dynamic logic model* of web data dynamics and provide some experiments. We hope this can serve as preliminary experiments for a more extended research proposal.

2 Extraction Experiments

Our main goal is: Try to remember information about the context of data creation in order to enable repetition of extraction. In general we can remember algorithms, training data and metrics of success.

In this chapter we try to describe some extraction algorithms and process of data integration in movie domain. These data will be used in preference learning. By this we would like to illustrate our main goal.

2.1 Integration

We use *Flix* data (enriched *Netflix* competition data), *RecSys 2014 challenge* data [T] and *RuleML Challenge* data [K].

The datasets are quite different but they still have few things in common. Movies have their title and usually also the year of their production. Ratings are equipped by timestamp that allows us to order ratings from individual users chronologically.

One of problems was that different datasets use different **MOVIEID**'s, so the movies cannot be straightforwardly mapped across datasets. To achieve this goal we wanted to enhance every movie by the corresponding IMDb identifier.

We observed that the *Twitter datasets* use as their internal **MOVIEID** the numeric part of the IMDb identifier. So the movie “Midnight Cowboy” with Twitter **MOVIEID** = 64665 corresponds to the IMDb record with ID equal to 'tt0064665'. Therefore, we construct the algorithm

$$\tau_1: \text{Twitter-MOVIEID} \rightarrow \text{IMDbId}$$

which simply concatenated prefix 'tt' with the **MOVIEID** left-padded by zeroes to seven positions. The successfulness of this algorithm is shown in Table 1.

Table 1: Simple identifier transformation

Algorithm	MovieLens	Flix	Twitter
$\tau_1()$	0%	0%	100%

To be able to assign IMDb identifiers to movies from other datasets, we had to use the search capabilities of the IMDb database. We used an HTTP interface for searching movies according to their name. The request is send by HTTP request in form <http://www.imdb.com/find?q=movie+title&s=tt>.

The other versions of algorithm for assigning IMDb identifiers to movies can be in general formally described as

$$\tau_2^{ij}: \text{TITLE} \times \text{YEAR} \rightarrow \text{TT}$$

that is implemented by two independent steps

$$\tau_2^{ij}(\text{TITLE}, \text{YEAR}) = \sigma_j(\eta_i(\text{TITLE}), \text{TITLE}, \text{YEAR})$$

where the algorithm η_i transforms movie title to query string needed for IMDb search, while the σ_j algorithm then looks for the correct record in returned table. The simplest implementation of algorithms can be denoted as follows:

$$\eta_1: \text{TITLE} \rightarrow \text{TABLE}$$

$$\sigma_1: \text{TABLE} \times \text{TITLE} \times \text{YEAR} \rightarrow \text{TT}$$

where η_1 algorithm concatenates all words of the title by the plus sign and σ_1 algorithm returns TT in case the resulting table contains exactly one record. The results of this combination of algorithm are shown in Table 2 (ratio of correct answers).

Table 2: The simplest version of IMDb search by title name

Algorithm	MovieLens	Flix	Twitter
$\sigma_1(\eta_1())$	42,7%	51,2%	Not needed

To illustrate different algorithms for same extraction task we describe another version. Here the algorithm is not learned, but it is hand crafted. One of reasons for relatively low effectiveness of $\sigma_1(\eta_1())$ algorithm was the sub-optimal query string used for IMDb search due to quite different naming conventions of movie titles in different datasets. To improve the results we enhanced the movie title transformation incrementally and produced its new versions. Every new version added new step of transformation of the movie title:

η_2 : Convert all letters in movie title to lower case.

η_3 : If the movie title contains year of production at its end in brackets remove it.

η_4 : If the movie title still contains text in brackets at its end, remove it. This text usually contained original name of movie in original language.

η_5 : Move word “the”, respectively “a”/ “an” from the end of the title to the beginning.

η_6 : Translate characters “_”, “.”, “?” and “,” to spaces

η_7 : Translate “&” and “&,” in titles to word “and”

For example, the η_7 transformation changes title “**Official Story, The (La Historia Oficial) (1985)**” to its canonical form “**the official story**”.

This version of transformation then constructs the IMDb query in form <http://www.imdb.com/find?q=the+official+story&s=tt&...> and then looks up the resulting table to find identifier “tt0089276”.

The results of this combination of algorithm were:

Table 3: The more complex version of IMDb search by title name

Algorithm	MovieLens	Flix	Twitter
$\sigma_1(\eta_7())$	45,4%	70,9%	Not needed

In optimal case, the table returning from the IMDb search contains exactly one row with the requested record. For this situation the algorithm σ_1 behaves well and is able to retrieve the correct IMDb identifier. In many other cases the result contains more rows and the correct one or the best possible one has to be identified. For this purpose we constructed more versions of the σ_j algorithm as well:

σ_2 : The correct record should be from the requested year, so search only for records from this year and ignore other records

σ_3 : The IMDb search provides more levels of tolerance in title matching. Try to use thee of them from the most exact one to the most general. If the matching record from requested year cannot be found using stricter search, the other search level is used.

Currently, we have 13 081 out of all 17 770 *Flix* movies mapped onto the IMDb database. Even all 27 278 movies from the *MovieLens* set are mapped to the equivalent IMDb record. So the current results provided by the combination of most advanced versions of algorithms are:

Table 4: The most complex version of IMDb search by title name

Algorithm	MovieLens	Flix	Twitter
$\sigma_3(\eta_7())$	100.0%	73.6%	Not needed

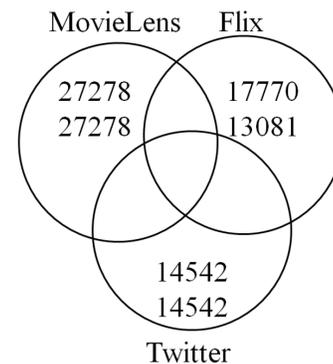


Figure 3: Numbers of movies mapped onto IMDb database

The diagram in the Figure 3 shows the number of movies in individual datasets and number of movies assigned to their corresponding IMDb record. Amount of movies associated to the IMDb record in different intersections after the integration is different. For example, the *MovieLens* dataset contains in total 27 278 movies. From these are 13 134 unique associated movies and also contain 3 759 associated movies common with the *Flix* dataset not existing in the *Twitter* dataset. The number of movies common for all three datasets is equal to 4 075. By summing of 3 759 and 4 075 we get the total number of 7 654 associated movies belonging to both *MovieLens* and *Flix* datasets, etc.

2.2 Extraction of attributes

For each movie registered in the IMDb database we then retrieved XML data from the URL address

<http://www.omdbapi.com/?i=ttNNNNNNN&plot=full&r=xml>

and then from the XML data retrieve following movie attributes:

IMDb title	(/root/movie/@title),
IMDb rating	(/root/movie/@imdbRating),
IMDb rated	(/root/movie/@rated),
IMDb awards	(/root/movie/@awards),
IMDb metascore	(/root/movie/@metascore),
IMDb year	(/root/movie/@year),
IMDb country	(/root/movie/@country),
IMDb language	(/root/movie/@language),
IMDb genres	(/root/movie/@genre),
IMDb director	(/root/movie/@director),
IMDb actors	(/root/movie/@actors)

The similar way the movies from datasets are mapped onto IMDb movies, we implemented the mapping technique described in [K] and assigned *DbPedia*¹ identifiers and semantic data to IMDb movies.

The *DbPedia* identifier of movie is a string, for example "The_Official_Story" or "The_Seventh_Seal". This identifier can then be used to access directly the *DbPedia* graph database or retrieve data in an XML format through the URL address in form <http://dbpedia.org/page/DbPediaIdentifier>. From the data available on the *DbPedia* page can be directly or indirectly extracted movie attributes GENRE, GENRE1, ACTION, ADVENTURE, ANIMATION, CHILDRENS, COMEDY, CRIME, DOCUMENTARY, DRAMA, FANTASY, FILM_NOIR, HORROR, MYSTERY, MUSICAL, ROMANCE, SCI_FI, THRILLER, WAR, WESTERN or attributes CALIF, LA, NY, CAMERON, VISUAL, SEDIT, NOVELS, SMIX, SPIELBERG, MAKEUP, WILLIAMS and many others.

3 A Dynamic Logic Model for Web Annotation

For effective using of changing and/or increasing information we have to evolve tools (e.g. inductive methods) used for creation of specific web service (here recommendation of movies). Our goal is to extend the semantic web foundations to enable describing creation, dynamics and similarities on data. To describe the reliability of extraction algorithms we propose a "half-away" extension of dynamic logic.

Our reference for dynamic logic is the book of D. Harel, D. Kozen, J. Tiuryn [HKT].

Dynamic logic has two types of symbols: propositions/formulas $\varphi, \psi \in \Pi$ and programs $\alpha, \beta \in \Phi$. One can construct a program also from a formula by test $\varphi?$ and formulas also by generalized modality operations $[\alpha], \langle \alpha \rangle$. The expression $\langle \alpha \rangle \varphi$ says that it is possible to execute α and halt in a state satisfying φ ; the expression $[\alpha] \varphi$ says that whenever α halts, it does so in a state satisfying φ .

Main goal of dynamic logic is reasoning about programs, e.g. in program verification. In our case programs will be extractor/annotators and can be kept propositional, as for now we are not interested in procedural details of extractors. Formulas will be more expressible in order to be able to describe the context of extraction.

Using the example above, let

φ be the statement that a Twitter data entry has title "Midnight Cowboy" and **MOVIEID** = 64665;

α be the algorithm concatenating prefix 'tt' with the **MOVIEID** left-padded by zeroes to seven positions; and ψ says movie "Midnight Cowboy" corresponds to the IMDb record with ID equal to 'tt0064665'.

The corresponding dynamic logic expression is

$$\forall x(\varphi(x) \rightarrow [\alpha]\psi(x))$$

saying that whenever α starts in a state satisfying $\varphi(m_1)$ then whenever α halts, it does so in a state satisfying $\psi(m_1)$ - see illustration in Fig.4.

Programs (extractors) remain propositional, states correspond to different representation of content on the web. On each of states the respective semantics is defined using appropriate query language.

Our logic has expressions of two sorts and each sort is, respectively can be typed:

Statements about web data: can be either atomic, e.g. $\Phi_0^{RDF}, \Phi_0^{FOL}, \Phi_0^{RDB}, \Phi_0^{XML}, \Phi_0^{DOM}, \Phi_0^{BoW}, \Phi_0^{PoS}, \Phi_0^{DepTree}$, etc. or more complex, e.g. $\varphi^{RDF}, \psi^{FOL}$, etc. With the corresponding data model and query language based semantics. All can be subject of uncertainty, probability extensions.

Programs (propositional): atomic Π_0^σ for subject extraction, Π_0^π for property extraction or Π_0^ω for object value extraction in case of HTML, XHTML, or XML data; Π_0^{ner} for named entity extraction in case of text data, etc. and more complex $\alpha^{\sigma\pi\omega}, \beta^{\sigma\pi\omega}, \gamma^{\sigma\pi\omega}$, etc. In this logic we do not prove any statements about program depending on their code, so program names point to code one would reuse.

Statements are typically accompanied by information about program creation like data mining tool, training data, metrics (e.g. precision, recall), etc. There is also a lot of reification describing the training and testing data and the metrics of learning. Our model is based on dynamic logic, calculates similarity of states and describes uncertain/stochastic character of our knowledge.

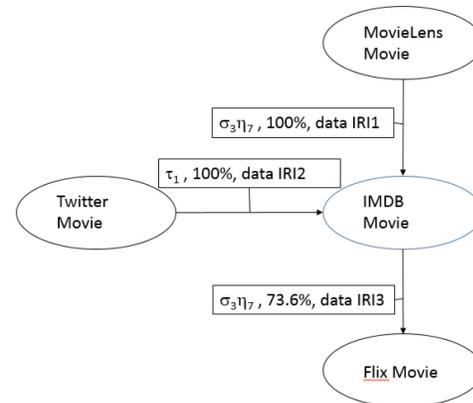


Figure 4: Extraction data enriched

¹ <http://wiki.dbpedia.org/>

Hence we are able to express our extraction experience in statements like

$$\varphi \rightarrow [\alpha]_x \psi$$

where φ is a statement about data D_1 before extraction (preconditions), ψ is a statement about data/knowledge D_2 , K_2 after extraction (postconditions), α is the program used for extraction. Modality $[\alpha]_x$ can be weighted, describing uncertainty aspects of learning.

Lot of additional reification about learning can be helpful.

The main idea of this paper is that if there are some data D_1' similar to D_1 and φ is true in some degree – e.g. because both resources were created using same template - then after using α we can conclude with high certainty/probability that the statement ψ will be true on data D_2' (knowledge K_2').

For instance the formula

“MyData are similar to IRI3” \rightarrow $[\sigma_{3\eta 7}]_{0.736}$ “IMDBId is correct”

Experiments with extraction and integration of movie data can serve as an example of this. In the next chapter we would like illustrate how this influences recommendation.

4 Preference Learning Experiments

To show usability of extracted and annotated data, we provide experiments in area of recommender systems.

4.1 Data Preprocessing

We selected all *Twitter* users with ratings less or equal to 10, random 3000 *MovieLens* users and random 3000 *Flix* users.

For these, we split the **RATING** data by assigning last (according to time stamp) 5 records from each user as a test data, the remaining data was used as train data.

Based on train data, we calculated aggregated variables:

Table 5: Computed variables for each movie

Variable	Description
α_1 CNT	Number of ratings for a movie
α_2 MAVG	Average rating for a movie
α_3 BAYESAVG	(GLOBAL.AVERAGE*50 +MAVG*CNT) / (CNT+50)

Table 6: Computed variables for each user

Variable	Description
α_4 USERSHIFT	The average over rated movies (user.rating-BAYESAVG)

Table 7: Computed variables for each pair of user and movie

Variable	Description
α_5 GENREMATCH	Equality of the most frequent user’s genre and the movie’s genre

4.2 Results

Based on these attributes, we learned a linear model of **RATING** as a function of (CNT+BAYESAVG+MAVG+USERSHIFT+GENREMATCH).

Table 8 summarizes the train mean square error, test mean square error and for comparison the mean square error of the 'zero' model predicting always the overall average of training data. These are the uncertainty part of our dynamic logic.

Table 8: Result summarization

Dataset	MRSS train	MRSS test	MTSS train
Flix	0.690	0.714	1.100
Twitter	1.420	1.660	2.890
MovieLens	0.607	0.633	1.070

We can see that in all datasets, the difference between train and test error is very small compared with the results of the zero model. This means the model is not overfitted.

Since the *Twitter* dataset uses scale 0 to 10 compared to the scale 0 to 5 for *Flix* and *MovieLens*, the error differences cannot be compared directly.

The models may be compared by the R^2 statistics, the amount of variance explained by the model

$$R^2 = 1 - \text{Sum}(R[i]^2) / \text{Sum}((y[i] - y^*)^2)$$

Here $R[i]$ is the i^{th} residual, $y[i]$ is the rating of i^{th} record, y^* is average rating over all train records in dataset. Its range is from 0 (useless model) to 1 (perfect prediction).

In table 9 we can see significant differences between datasets, ranging from 0.506 for *Twitter* (best improvement) compared to 0.356 for *MovieLens* and 0.262 for *Flix*.

Table 9: R^2 statistics

Dataset	R^2
Flix	0.262
Twitter	0.506
MovieLens	0.356

In Table 10 we show preliminary results on testing repeatability. We trained the model on the data set in the row and tested on the test data in column. No surprise that in each column the best result is on the diagonal.

Table 10: Repeatability tests

Dataset	Flix test	Twitter test	MovieLens test
Flix train	0.714	1.731	0.635
Twitter train	0.723	1.658	0.639
MovieLens train	0.715	1.679	0.633
Zero model	1.100	2.890	1.070

As the zero model we use movie rating average MAVG. In this research proposal, we do not evaluate role of similarity, we just illustrate similarity of our datasets. In Figure 5 we show MAVG function on a sample of movies (with IMDB ID#s). Table 11 show MAVG distances below diagonal. So far it is not clear to us which metrics to use to compute similarity – Euclidean or cosine? Further experiments are required as this can depend on domain.

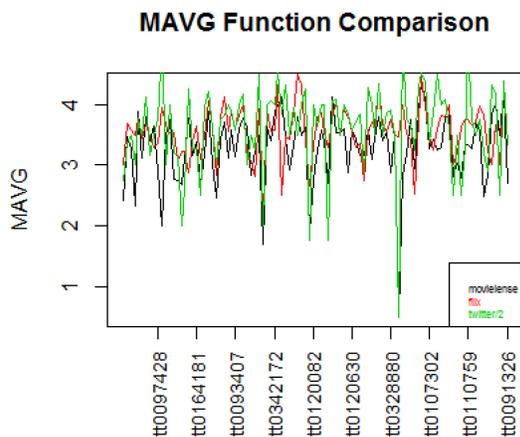


Figure 5: Towards calculating similarity – which vectors, which metrics? Here MAVG.

Maybe the right idea to calculate similarity is content based. We illustrate this by Fig. 6 with behavior of statistics on genres. Table 11 show genre based distance in cells above diagonal.

5 Proposal, Conclusions, Future Work

We have provided preliminary experiments with reusability of our algorithms. Results are promising, but still we need more extensive testing.

5.1 Proposal of Reusability Quality

Similarly, as in Linked data quality assessment, we can imagine similar assessment for reusability.

The main idea is, that this will be less sensitive to URL change, migration and “end-of-project-phenomenon”. One can imagine, that these information are published at <https://sourceforge.net/> or similar service. What follows is a vision, we would like to discuss:

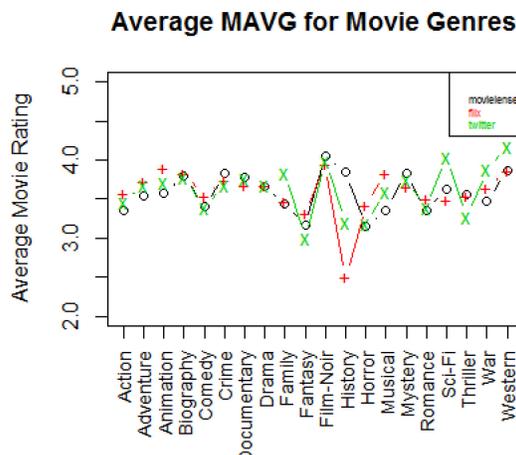


Figure 6: Maybe genres play a role?

Table 11: Distance of datasets. Below diagonal by MAVG, above diagonal with Genres

	Flix	Twitter	Movie Lens
Flix	0.00	0.10	0.14
Twitter	0.51	0.00	0.06
MovieLens	0.42	0.54	0.00

6★ Reusability extraction describes the algorithm, training data, metrics and results.

7★ Reusability extraction extends a 6★ one by additional similarity measures and thresholds for successful reuse. A corresponding formula can look like

$$“Data \approx_{0.2} IRI3” \rightarrow [\sigma_{3\eta}7]_{0.654} “IMDBid \text{ is correct}”$$

8★ Reusability extraction describes a 7★ one in a more extensive way with several different data examples and similarities. This can increase the chance that for a given domain you find solution which fits your data.

9★ Reusability extraction assumes a 8★ one in a server/cloud implementation. You do not have to solve problems that the extractor does not run in your environment properly.

10★ Reusability extraction assumes a 9★ one in a more user friendly way, you just upload your data (or their URL) and the system finds solution and you can download result. It is also possible to imagine this enhanced with some social network interaction.

5.2 Conclusions

We have presented a research proposal for improving degree of automation of web information extraction and annotation. We propose a formal dynamic logic model for automated web annotation with similarity and reliability.

We illustrated our approach by an initial prototype and experiments on recommendation on movie data (annotated and integrated).

5.3 Future work

The challenge is twofold:

- extend this to other domains
- provide deeper analysis of data mining and possible similarities

We can consider some more complex algorithms for preference learning, e.g., based on the spreading activation [GG].

Acknowledgement

Research was supported by Czech project Progres Q48.

References

- [A] [IBM] John Arwe. Coping with Un-Cool URIs in the Web of Linked Data, LEDP-Linked Enterprise Data Patterns workshop. Data-driven Applications on the Web. 6-7 December 2011, Cambridge, MA Hosted by W3C/MIT
- [BTMC] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham (2004), Evolving GATE to Meet New Challenges in Language Engineering, *Natural Language Engineering*, 10(3/4):349—373.
- [D1] J. Dedek, Semantic Annotations 2012, http://www.ksi.mff.cuni.cz/~dedek/publications/Dedek_PhD_Semantic_Annotations.pdf
- [D2] J. Dedek. Semantic Czech, <http://czsem.sourceforge.net/>
- [F1] D. Fiser. Semantic annotation of domain dependent data (in Czech). Master thesis, Charles University, Prague 2011
- [F2] D. Fiser. Semantic annotator. <https://chrome.google.com/webstore/detail/semantic-annotator/gbphidobolopkilhnpkbagodhalimojj>, <http://www.doser.cz/projects/semantic-annotator/> (User Guide and Installation Guide (server part) in Czech),
- [GG] L. Grad-Gyenge, Recommendations on a knowledge graph, 2015
- [HKT] D. Harel, D. Kozen, J. Tiuryn. *Dynamic Logic (Foundations of Computing)* The MIT Press 2000
- [K] Kuchar J.: Augmenting a Feature Set of Movies Using Linked Open Data, Proceedings of the RuleML 2015 Challenge, Berlin, Germany. Published by CEUR Workshop Proceedings, 2015
- [PLEDFV] L. Peska, I. Lasek, A. Eckhardt, J. Dedek, P. Vojtas, D. Fiser: Towards web semantization and user understanding. In EJC 2012, Y. Kiyoki et al Eds. *Frontiers in Artificial Intelligence and Applications* 251, IOS Press 2013, pp 63-81
- [T] Twitter data from RecSys 2014 challenge <http://2014.recsyschallenge.com/>
- [UFAL] S. Cinková, J. Hajič, M. Mikulová, L. Mladová, A. Nedolužko, P. Pajas, J. Panevová, J. Semecký, J. Šindlerová, J. Toman, Z. Urešová, Z. Žabokrtský (2006), Annotation of English on the tectogrammatical level, Technical Report 35, UFAL MFF UK, URL http://ufal.mff.cuni.cz/pedt/papers/TR_En.pdf.
- [V+] Verborgh R. et al. (2014) Querying Datasets on the Web with High Availability. In: Mika P. et al. (eds) *The Semantic Web – ISWC 2014*. ISWC 2014. Lecture Notes in Computer Science, vol 8796. Springer, Cham