User-sensitive Explanations under a Knowledge Pattern Lens

Alessandro Adamou[1,2], Paolo Ciancarini[1], Aldo Gangemi[2], and Valentina Presutti[1,2]

[1] Alma Mater Studiorum Università di Bologna, Italy
[2] ISTC, National Research Council, Italy

Abstract. This paper introduces our ongoing research on a general-purpose methodology for generating explanations for occurrences of interaction patterns. Explanations are tailored around a user's profile or interaction history in interactive systems. The approach relies on the recognition of interlinks between interaction patterns and knowledge patterns, both formally modeled as networked ontologies. In addition, the method constructs its statements using the same knowledge shared by the hosting system, including distributed knowledge such as Linked Data. We plan to implement and evaluate this approach in the context of Content Management Systems and related interaction patterns such as query disambiguation, content recommendation, faceted search and browsing.

1 Introduction

In interactive systems, the need for interpreting system behaviour has seen a steady growth as the supported recurring schemes, or *patterns* in human-computer interaction (recommendation, tagging, query disambiguation, faceted search and browsing being some) increase in amount and complexity. The more a system replaces natural language with an iconic or mutimodal one, the less it tends to be self-explanatory. We collectively dub the functionalities that address this issue (a simple example being *tooltips*) as "explanations", in that they justify either the content of system feedback, or its form of presentation to the user. Examples include providing information on an entity portrayed in a picture, the meaning of a chart portion being shown in a certain color, or a justification as to why a certain multimedia item is being recommended to a customer.

Software systems tend to short-circuit this issue by hardwiring ad-hoc functionalities, each dealing with a specific use case and whose output is tightly bound with the functionality they serve. For instance, explanation-featuring recommender systems support the entire cycle autonomously, by generating the exact sentence that will be delivered to the user along with the recommendation. Despite its basis on general principles common to other interaction-oriented functionalities, an approach like this is hardly reusable. In addition, when these functionalities work in a "boxed" fashion, not interoperable with the rest of the system, they may fail to deliver "user-sensitive" explanations. Here, user-sensitivity denotes the ability of a system to deliver explanations that are (i)

tailored around a user's interaction context and/or profile; (ii) comprehensible by an agent whose only required knowledge is that of the domain at hand.

Consider for example a member of a research project X browsing the annual report on its expenditures. A chart shows the Q1-2011 portion marked yellow as opposed to others marked green. If the *project manager* hovers on that chart portion, a tooltip or text console will explain the color as "Project X has spent $2,000 above its planned quota in Q1, 2011". For another member, the message could instead be "Project X has slightly overspent" or "has spent 5% more than its planned quota", depending on what information that user has access to.

As it emerges from the examples above, an explanation generator does not guess why the system provided a certain type of feedback. This information should come from the system itself: in another example, it is the recommender system that knows why it came to select a particular item for recommendation. The challenge for explanation functionalities is to deliver this information, or part thereof, in a user-sensitive way, by selecting relevant pieces of knowledge in a context, and providing directives as to how they should be *assembled* together.

Ontologies, knowledge patterns, inference rules and reasoning are a solid set of tools for sewing together these mutually agnostic systems and functionalities into a general-purpose semantic framework that can serve multiple interaction patterns. This paper describes the first insights into our ongoing research on a method that can accomplish these goals. After a brief overview on existing work in Section 2, we sketch in Section 3 the general workflow of our method under research and the types of resource it is based upon. In Section 4 we show with an example how this method applies to possible occurrences of the interaction patterns we can support. We conclude with Section 5, by describing the ongoing work and our plan for a proof-of-concept implementation of the method.

2 Related work

Historically, explanation generation has been one of the fields of study of computational linguistics for over two decades [6]. An algorithm based on abductive reasoning, which targets the explanation of queried events, has been around since 1987 [1]. Another explanation system grounded in the biology domain was Knight [5], which combined early knowledge pattern identification and usage in order to provide definitions of entities in the given domain.

We acknowledge the above as groundbreaking work that inspired our research. Nowadays however, with the rise of Linked Data and heterogeneous knowledge sources on one hand, and the increasing interaction patterns support on the other hand, demands of cross-domain and cross-application flexibility are being pushed beyond those systems. As to modern interaction patterns, *Tagsplanations* address the relevance and sentiment of users towards tags for generating recommendations [10], where tags on content items are sorted and selected to justify a recommendation. We argue that an approach such as this should not limit to processing user-generated tags and supporting the recommendation pattern, but should instead be a cornerstone for selecting relevant statements in general-

purpose explanation approaches for entities, facts and interactions. Tintarev et al. analyzed the goals and metrics involved with explaining recommendations, e.g. effectiveness, efficiency, persuasion and transparency; and investigated on a method to elicit effectiveness [9]. Experiments for evaluating the impact of explanations in automated collaborative filtering systems were also conducted [4]. They used a white-box model of explanations, which was designed ad-hoc for this single interaction pattern, as it arguably does not involve reasoning.

Finally, the exploitation of interaction patterns and their representation using formal semantics has also begun to occur in studies on multimodality [8]. Multimodal interaction patterns in this study encompass the spatial and temporal relations between modalities, such a sequentiality and simultaneity. While we are not currently targeting multimodality, we gain inspiration from this study with respect to the formal treatment of sequential input in interaction models.

3 Approach

Our proposed general-purpose approach relies on the following knowledge, either made available by the host system, e.g. a Content Management System (CMS), or authored as part of the explanation strategy itself.

Interaction Pattern catalog. A collection of solutions to common usability problems, which become recurring interaction schemes in a user-system dialog. Numerous libraries of interaction pattern specifications are available, one being by Martijn van Welie[3]. To reason upon them, we will require a model for representing them as ontologies. This is work-in-progress for this research[4], which concentrates on interaction patterns for manipulating content and knowledge. We will focus on interaction patterns expected to occur in a CMS, such as recommendation, faceted search and browsing, annotation and query disambiguation.

Knowledge Pattern catalog. A collection of formal minimal models used to describe a concept, state or event in the real world. Knowledge patterns (KPs) are *invariances across observed data or objects* that allow a *formal or cognitive interpretation* [3]. These will contribute to the selection of statements for explanations: in this respect, the ties with linguistic frames and FrameNet [7] as a repository of such models are evident. The Content Pattern section of the ODP portal[5], which we also plan to enrich, will be our experimental basis.

Real-world knowledge. With this term, we denote the content of the knowledge base managed by the host system. Other than its rendering as RDF, no assumption is made as to where this knowledge is stored and which vocabularies are used. It may, for example, be a simple hub that crawls and indexes the Linked Data cloud, or a centralized repository of in-house knowledge.

[3] Interaction Pattern Library, `http://www.welie.com/patterns/`

[4] Interaction pattern model WIP, version control at `https://iks-project.googlecode.com/svn/sandbox/explanation/trunk/src/main/resources/model/`

[5] Ontology Design Patterns, `http://www.ontologydesignpatterns.org`

User interaction trace. For explanations to be tailored around the flow of interaction of a given user with a given system, the system itself has to provide the interaction history, or *discourse context*. It is essentially a semantic log of user interaction and the interface elements, widgets or multimodal channels involved. It is an extension of the interaction pattern metamodel that we will provide.

Mappings of two different kinds:

- The *interpretation* of user interaction elements as the real-world entities they denote; for example, the avatar icon of a person can be interpreted as that very person; a drop area can be interpreted as a physical location, or a task; selecting a point on a map can be interpreted as requesting the points of interest nearby, or setting it as the next destination of a trip. Interpretations should be provided by the host system. The mapping vocabulary will be an extension of our interaction pattern metamodel.
- *Alignments* between controlled vocabularies and KPs, which are not built using these vocabularies. They can be e.g. `owl:equivalentClass` axioms between `kp:Person`[6] and the classes `Person` in FOAF and DBPedia. Alignments can be assumed to be provided by the knowledge pattern authors.

Annotations of knowledge patterns, with the dual purpose of marking their key elements in order to (i) determine if a pattern can be matched in the knowledge base; (ii) select entities and facts that should participate into statements that will form the explanation. Groups of such elements can also be marked as key.

Our explanation strategy proceeds along the lines of the following workflow. We assume it to be triggered every time a user action or system feedback is issued and logged by the host system, i.e. stored in the *interaction trace*.

1. **Interaction pattern detection.** If this information is registered by the host system, this step is unnecessary; otherwise, we will need to establish whether the most recent sequence of utterances in a user-system dialog matches an occurrence of one of the interaction patterns in our catalog. This is done by matching the registered actions and their involved UI elements against the classes of actions and UI elements defined in each interaction pattern (which are represented using the same ontology). If more than one such interaction pattern is satisfied, heuristics may be applied to select the most likely (e.g. the one that traces furthest back in the interaction history).
2. **Interpretation grounding.** For every individual in the detected interaction pattern occurrence, check which individuals in the knowledge base it maps to, according to the *interpretation mapping*. Extract the types - both asserted and inferred - of every such individual.
3. **Candidate knowledge pattern set construction.** If the interpretation explicitly states which knowledge pattern(s) the detected interaction pattern maps to, let $\mathcal{KP}$ be the set of these knowledge pattern(s) *and* their specializations, if any; otherwise, let $\mathcal{KP}$ be the entire set of knowledge patterns.

[6] The `kp` prefix is a placeholder for the namespace to be used for knowledge patterns.

4. **Knowledge pattern satisfiability check.** For each knowledge pattern in $\mathcal{KP}$, determine if it is satisfied. A knowledge pattern is satisfied iff all its elements marked as *key elements* are filled, i.e. for each key class or property there is a corresponding individual or predicate in the knowledge base. Let $\mathcal{KP}'$ be the set of *satisfied* knowledge patterns.
5. **Explanation assembly.** For each satisfied knowledge pattern in $\mathcal{KP}'$, select the statements (assertions or inferences) in the knowledge base that (i) map to the knowledge pattern and (ii) have the greatest weights according to their annotations in the knowledge pattern model. Weight corrections can be applied using a coefficient calculated by arbitrary heuristics, e.g. the number of mentions of that entity in the interaction context. The list of selected statements is the *explanation* for this occurrence of the detected KP.

4 Example

An explanation is, in general, an *interpretation* of an *interaction pattern occurrence* in a running system. One of our goals is to *classify* interaction patterns in terms of the types of explanations that typically accompany them. Note that explanations themselves can be part of some interaction patterns, such as those classifiable as *explanation requests*, e.g. tooltips that appear upon hovering on a widget, or clickable "*more...*" or "*why?*" links. When this is not the case, explanation can be seen as an *ancillary* interaction pattern, which conceptually accompanies another one such as recommendation, search or annotation.

In the conceptually simplest cases, an explanation consists of providing a summary description, or *synopsis*, of an entity in the 'real' (as opposed to 'virtual') world. Typically, this is conveyed through the provision of attributes and/or facts involving the object to be explained. We shall now exemplify this for an interaction pattern with a peculiar requirement in this matter.

4.1 Query disambiguation

An as-you-type entity search is issued for annotating the text item "George Bush" in the content being written. When the query returns both former Presidents of the United States, the system prompts the user to select the appropriate one. To aid the user in the selection, a description must be provided for both results, but these descriptions must differ enough for the user to be able to disambiguate. An explanation whose statements are "former President of the U.S.; war with Iraq under his administration", if perfectly sound for either George Bush individual, would be totally ineffective for the interaction pattern at hand. In other words, the synopses of the two entities must minimize their overlap.

Figure 1 exemplifies our rationale for this interaction pattern. A simple Query Disambiguation pattern can be formalized as including the following individuals:

- a *system action*, which in this case consists of responding to a previous query;
- a *selector widget* populated by the system action with a list of ambiguous entries, each of them mapping to (i.e. `interpretedAs`) a real-world entity.

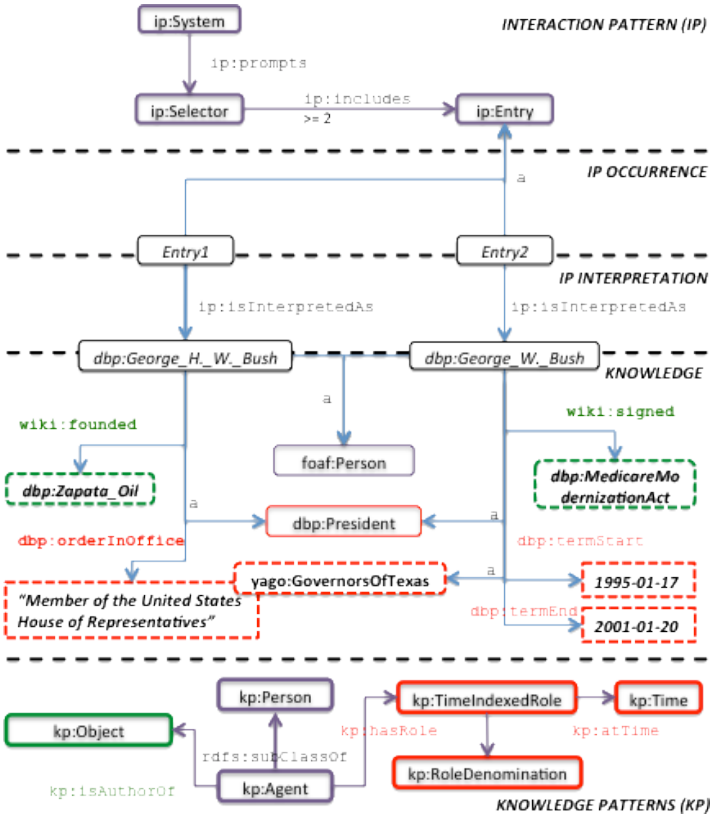

Fig. 1. Selection of statements for explaining disambiguation on the query "George Bush", from the interaction pattern (top) to the knowledge pattern (bottom) level. Entities marked in red indicate an association to a *time-indexed role* knowledge pattern; those in green indicate an association to an *authorship* knowledge pattern.

Suppose the explanation generator has access to all the named entities recognized for the same content, as well as the same knowledge base used by the semantic search engine that handled the query (see "knowledge" portion of the figure). Also, suppose the knowledge base types each "Bush" individual as a `foaf:Person`. The Query Disambiguation interaction pattern does not map to any specific knowledge pattern, since we have to discover which knowledge patterns apply to each entity. However, we do know we can restrict to those knowledge patterns that involve a `kp:Person` (for which we have mappings for commonly used types like `foaf:Person` in FOAF, `schemaorg:Person` in schema.org or `dbp:Person` in DBPedia). From all the facts - both asserted and inferred - about these two entities in the knowledge base, we detect several occurrences that satisfy a *time-indexed role* pattern, e.g. George W. Bush's office as Governor of Texas between 1995 and 2001, and George H.W. Bush's office as U.S. congressman 1967-1971. For a general *authorship* pattern, an instantiation is

detected for George H.W. Bush as a founder of the Zapata Oil company, and for George W. Bush as signer of the Medicare Act of 2003.

These facts satisfying some knowledge pattern are also weighed according to an arbitrary set of heuristics, which may include the number of occurrences in the knowledge base (suppose it to be, for example, a hub of large Linked Data sets where the same statements can occur multiple times using different identifiers) and the amount of mentions of related entities (such as Texas or Medicare) in the content item edited by the user. The greatest-weighed facts (denoted by the colored entities with a dashed outline in Figure 1) are selected and included as statements in the synopsis for each entity responding to a "George Bush" query.

If there are semantic relations involving the current user, then profile information can be included in the context and the KPs satisfied by these relations can be prioritized. For instance, if the knowledge base states that a `foaf:Person` matching this user (via alignments to a *self* pattern, e.g. by matching user names in the system or `foaf:name` property values) has worked as a White House administrative between 2002 and 2007, the combined *membership*[7] and *hierarchy* KPs can be satisfied. Then, a statement such as "your former boss" for George W. Bush will be viable for generation and have a greater weight for inclusion.

5 Ongoing and future work

Our current focus is on defining, reusing and reengineering formal schemas, annotation vocabularies and content for the knowledge required per the first part of Section 3. This includes knowledge and interaction pattern models, extensions for interaction traces, interpretations and mappings with popular and emerging controlled vocabularies such as DBpedia[8] and `schema.org`. For the interaction metamodel, we are taking inspiration from the interaction and interface modules of the C-ODO Light ontology network for managing ontology lifecycles [2].

As a basis for evaluating the designed methodology, the implementation of a proof of concept is in progress. We will focus on Content Management Systems (CMS) as versatile interactive systems in collaborative contexts where content and knowledge are managed. We intend to prioritize the association of explanation support with the interaction patterns that most frequently occur in such systems[9], e.g. (1) annotation of content with knowledge; (2) autocompletion; (3) faceted search and browsing; (4) query disambiguation; (5) content recommendation; (6) drag-&-drop (in latest-generation WebCMS).

The implementation under construction is a set of plugins for the **Apache Stanbol** service platform for semantic CMS[10], which provides functionalities for improving knowledge management and interaction. In particular, an inference rule language, compatible with the SWRL rule and SPARQL query languages, will be used for defining interpretations and alignments whenever assertions more

[7] `http://ontologydesignpatterns.org/cp/owl/timeindexedmembership.owl`
[8] DBpedia TBox, `http://downloads.dbpedia.org/3.6/dbpedia_3.6.owl.bz2`
[9] CMS interaction, `http://wiki.iks-project.eu/index.php/InteractionPatterns`
[10] Apache Stanbol incubation home, `http://incubator.apache.org/stanbol`

complex than equivalence or subsumption statements are required. Its ontology registry support will be used to manage multiple catalogs, such as knowledge patterns, interaction patterns and mappings. Its controlled environment for ontology networks allows us to scale reasoning and pattern detection tasks only on viable candidates or satisfiable knowledge and interaction patterns. A Web Service API will allow other modules in the host system to store the semantics of their user interaction traces, in accordance with the prescribed terminology.

Our explanation method will be user-evaluated against experimental and control groups extracted from Semantic CMS community members, which will include their users as well as their adopters and providers. The effectiveness of our approach will be evaluated by applying our proof of concept to specific use cases and domains which typically employ such interactive systems, e.g. project management and news publishing. Efficiency will be assessed through quantitative measurements such as the lag over on the main functionality output and the response time of users in the accomplishment of tasks with and without explanation support, as well as with traditional hardcoded explanations.

Acknowledgements This work has been part-funded by the European Commission under grant agreement FP7-ICT-2007-3/ No. 231527 (IKS - Interactive Knowledge Stack)

References

1. Coombs, M.J., Hartley, R.T.: The MGR algorithm and its application to the generation of explanations for novel events. International Journal of Man-Machine Studies 27(5-6), 679–708 (1987)
2. Gangemi, A., Lehmann, J., Presutti, V., Nissim, M., Catenacci, C.: C-ODO: an OWL meta-model for collaborative ontology design. In: Noy, N.F., Alani, H., Stumme, G., Mika, P., Sure, Y., Vrandecic, D. (eds.) CKC. CEUR Workshop Proceedings, vol. 273. CEUR-WS.org (2007)
3. Gangemi, A., Presutti, V.: Towards a pattern science for the Semantic Web. Semantic Web 1(1-2), 61–68 (2010)
4. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: CSCW. pp. 241–250 (2000)
5. Lester, J.C., Porter, B.W.: Developing and empirically evaluating robust explanation generators: The KNIGHT experiments. Computational Linguistics 23(1), 65–101 (1997)
6. Maybury, M.T.: Communicative acts for explanation generation. International Journal of Man-Machine Studies 37(2), 135–172 (1992)
7. Nuzzolese, A.G., Gangemi, A., Presutti, V.: Gathering Lexical Linked Data and Knowledge Patterns from FrameNet. In: Proc. of the 6th International Conference on Knowledge Capture (K-CAP). pp. 41–48. Banff, Alberta, Canada (2011)
8. Taib, R., Ruiz, N.: Integrating semantics into multimodal interaction patterns. In: Popescu-Belis, A., Renals, S., Bourlard, H. (eds.) MLMI. Lecture Notes in Computer Science, vol. 4892, pp. 96–107. Springer (2007)
9. Tintarev, N., Masthoff, J.: Effective explanations of recommendations: user-centered design. In: Konstan, J.A., Riedl, J., Smyth, B. (eds.) RecSys. pp. 153–156. ACM (2007)
10. Vig, J., Sen, S., Riedl, J.: Tagsplanations: explaining recommendations using tags. In: Conati, C., Bauer, M., Oliver, N., Weld, D.S. (eds.) IUI. pp. 47–56. ACM (2009)