Personalization in Skipforward, an Ontology-Based Distributed Annotation System

Malte Kiesel[1] and Florian Mittag[2]

[1] DFKI GmbH, Kaiserslautern, Germany
[2] University of Tübingen, Germany
malte.kiesel@dfki.de / florian.mittag@uni-tuebingen.de

Abstract. Skipforward is a distributed annotation system allowing users to enter and browse statements about items and their features. Items can be things such as movies or books; item features are the genre of a movie or the storytelling pace of a book. Whenever multiple users annotate the same item with a statement about the same feature, these individual statements get aggregated by the system. For aggregation, individual user statements are weighted according to a competence metric based on the constrained Pearson correlation, adapted for Skipforward data: A user gets assigned high competence with regard to the feature in question if, for other items and the same feature type, he had a similar opinion to the current user. Since the competence metric is dependent on the user currently viewing the data, the user's view of the data is completely personalized. In this paper, the personalization aspect as well as the item and expert recommender are presented.

1 Introduction

Rating and recommendation web platforms have become important and ubiquitous nowadays. Typically, these platforms support collaborative filtering; users can rate items and are recommended items that people who liked the same items gave a high rating as well. This works fine for many cases and many domains; drawbacks are that in-depth explanations of recommendations cannot be given, and that the user has little control over the actual recommendation process. On the other hand, there is content-based filtering, which recommends items based on the features the user presumably likes. E.g., "We recommend song X since that song features prominent drums that you seem to like". Pandora.com is an example for such a system. This approach does not have the shortcomings of collaborative filtering outlined above; however, getting the content-based annotations needed for the recommendation process is costly, as this typically requires trusted experts.

Skipforward [4] pursues a hybrid approach—in terms of [6], it is a semantic recommender system pursuing an active item-based approach. Every ontology-based statement or *feature (instance)* Skipforward uses consists of a link to the item it refers to, a feature type[3], applicability value (+1: The feature applies to

[3] Technically, every user statement is an RDF instance of a subclass of the Skipforward `Feature` class.

the item, -1: The feature does *not* apply to the item), confidence value (0..1), and a plain text comment. The applicability value, in traditional recommender terms, corresponds to a user rating with regard to an item and a feature type. In the following, we avoid the term "rating" since this term implies item liking which does not quite fit in our case.

Skipforward's simple basic data model allows quite thorough annotation of items and provides rich metadata for recommender and other functionality. Conflicting annotations do not break the system; a competence metric is used for weighting individual statements for aggregated views on the system's data. The competence metric is the foundation for much of Skipforward's recommendation functionality, which not only includes an item recommender that finds similar items or items fitting some user-chosen feature profile, but also an expert recommender, and annotation recommenders (functionality that helps annotating items).

2 Components of Interest

For an overview of most Skipforward components, also see [4] and the Skipforward website[4] which also includes a screencast and online demo. In the following, we will describe the building blocks of the system: its top-level ontology, domain ontologies, the user interface, and recommender functionality.

2.1 Ontologies used in Skipforward

We have a number of requirements the top level ontology (coined *Skipinions*) shall be able to handle.
The ontology should be able to represent user opinions of items such as books, movies, etc. – we solve this by providing an `Item` and a `Feature` class. For example, book features could be "Thriller (Genre)" or "Fast-paced writing style". Every `Item` can be associated to a `Feature` by using the `Item`'s `hasFeature` property.
The facts databases of multiple users should be easy to merge. Fact databases can be just copied together using this approach. Smushing of items and features is done using the `owl:sameAs` predicate.
Provenance of statements needs to be tracked. We solve this by assigning an individual namespace to every user. Then, the URI of every instance created by this user has to use this namespace. This also fits nicely with Linked Open Data principles.
Plain text comments should be supported. Internationalized plain text comments can be added to `Feature`s.
It should be possible to explicitly dissent with an opinion of another user. This is implemented by the `applicability` property on every `Feature`. Applicability -1 means "this feature does not apply to this item", applicability $+1$ means

[4] `http://skipforward.opendfki.de/`

"this feature applies to this item", implementing the Open World Assumption. Additionally, any `Feature` can point to another `Feature` instance, implementing discussion threading.
Marking an opinion as uncertain should be possible. This is implemented by the `confidence` property on every `Feature`. Together with `applicability`, this forms a Dempster-Shafer-like approach.
The amount of noise seen by users should be kept minimal. The feature hierarchy as presented by the system is created by the owner of the respective namespace so arbitrary changes of the feature hierarchy are not possible. This is both a limitation and a feature of the system. It is limitating insofar as users cannot create new feature classes on the fly. On the other hand, systems that implement this (i.e., most normal tagging systems) show that a *lot* of entropy enters the system otherwise. We try to keep this noise limited to the feature instance level where it can be handled in a coherent manner.

The basic top level ontology structure can be seen in Figure 1.

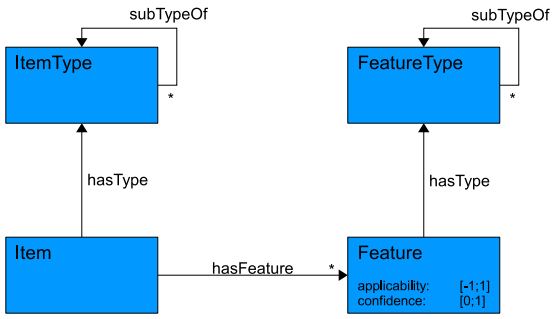

Fig. 1. Class diagram of Items and Features

Domain ontologies that subclass the `Item` and `Feature` classes as well as the `hasFeature` property are used for annotating actual items.

Currently, within Skipforward multiple domains are covered. Apart from ontologies that model features of board games and a corresponding set of instances, we mainly use *DBTropes* [3]. DBTropes.org is a wrapper of TVTropes.org, a wiki describing works of fiction by associating features—known as "Tropes"—to these works. The focus of TV Tropes is providing content-based annotations (as opposed to more technical information as, for example, supplied by IMDb.com), with a definite emphasis on fun and entertainment aspects. DBTropes extracts the information contained in the TV Tropes wiki, and publishes it as Linked Data. The implicit data model used in the TV Tropes wiki matches the data model used in Skipforward quite well. DBTropes uses Skipforward ontologies as its output format, and the main Skipforward application can consume this data directly. We use this data mainly as a source for feature types and the hierar-

chy within feature types. As of September 2011, the complete DBTropes data consists of about 10.000.000 RDF statements describing 22.000 items, 22.000 feature types, and 1.750.000 feature instances.

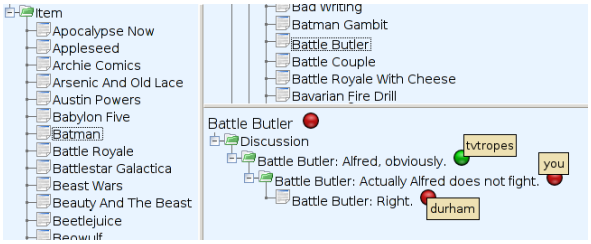

Fig. 2. Browsing DBTropes data in the Dojo-based Skipforward frontend.

2.2 Skipforward user interface

The Skipforward system is implemented as a web application. This allows running it easily in the background and in remote scenarios. Currently, we experiment with two frontends that serve slightly different purposes: A Dojo[5]-based Ajax UI, mainly used for annotating items, and a standard HTML template-driven UI that was built for better scalability and easier extensibility. The template HTML interface is mainly used for browsing and viewing additional information from the item and expert recommender components. In Figure 2, a small part of the data made available by DBTropes is shown, as visualized by the Skipforward UI implemented using the Dojo framework[6]. The left pane lists items (here, the movie *Batman* is selected); the upper right pane displays available feature types (here, the type *Battle Butler* is selected); the lower right pane shows instances of the selected feature type (i.e., users expressing opinions about one item with regard to one feature type). The uppermost (red) circle in the lower right pane shows the weighted average of applicability of the feature type with user opinions weighted according to their trust value. Here, three (threaded) user opinions for the feature *Battle Butler* in the movie *Batman* are available in the system. TV Tropes users stated that the feature *is* present for *Batman* (green circle: feature present) while the current user and the user *Durham* disagreed (red circle: feature not present). Note that the aggregated circle is deep red and not just a normal average of the individual user opinions (which would result in a light red or neutral tone). This leads us to the competence metric and weighting of user opinions.

[5] `http://dojotoolkit.org/`
[6] Note that the screenshot has been shortened for clarity.

2.3 Competence metric

The competence metric in Skipforward is based on user similarity with regard to feature types. This is different from traditional content-based filtering as it takes into account opinions of different users; it also differs from standard collaborative filtering which does not support multiple opinions concerning different feature types per item. It is calculated with a modified variant of the constrained Pearson correlation [7] shown in Formula 1. Here, u_x denotes user x, $r_{x,i}$ denotes the applicability value user x assigned to item i for feature type t, I_{xy} is the set of co-rated items of users x and y, and $w_{xy,i}$ denotes the combined confidence of the statements concerning feature type f of users x and y and item i. Note that these calculations need to be repeated for each Skipforward feature type.

$$sim_t(u_x, u_y) = \frac{\sum_{i \in I_{xy}} w_{xy,i} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_{xy}} w_{x,i} (r_{x,i})^2 \sum_{i \in I_{xy}} w_{y,i} (r_{y,i})^2}} \quad (1)$$

For efficient calculation, an incremental algorithm has been implemented, only recalculating similarity values on changes, and only locally. $sim_t(u_x, u_y)$ is used for weighting user statements for aggregated features. Aggregated features represent all user statements concerning one feature type and one item. They are used in the user interface and in recommenders. To compute the competence metric, the following algorithm is used.

Algorithm 2.1: CalculateAllCorrelations()

for each $i \in I$
  $copyFeaturesToCache(i)$
  for each $u \in U$
    $doInferencing(u, i)$
for each $t_f \in T_f$
  $calculateMean(localuser, t_f)$
  for each $u \in U$
    $calculateMean(u, t_f)$
    $calculateCorrelation(localuser, u, t_f)$

To compute aggregated features per item and feature type according to the competence metric, another algorithm is used: For all feature types an item is annotated with, the aggregated feature for a specific feature type is represented by a weighted sum of the individual features' applicability for this feature type. Applicability weight is the respective user's competence regarding the feature type.

In effect, this means that for the aggregated feature, statements made by people who have been assigned a low competence value influence the outcome less than statements made by people with a high competence value.

2.4 Item recommender

The item recommender shows items similar to the current item (Figure 3). It is based on a similarity metric comparing aggregated features assigned to the two items in question. For the standard item recommender user interface shown on each item page, this is a straightforward distance metric comparing feature applicability. Additionally, there is also an advanced recommender which lets users freely select and weight individual feature types, implementing recommendation channels.

Similar items

Item	Similarity	Matches
Natural State	0.67	Fast story pac
Far & Deep	0.52	Fast story pac
Here We Are, Falling Through Shadows	0.45	Fantasy, Revi
Orchestral Manoeuvers in the Dark Matter	0.42	Fast story pac
Freedom(TM)	0.40	Fast story pac
Intellectual Property	0.39	Fast story pac
The Stories of Ibis	0.34	Fast story pac
When the Thorns are the Tips of Trees	0.33	Fantasy, Revi
After Everything Woke Up	0.32	Fast story pac
Fool's Errand	0.32	Fast story pac

Fig. 3. List of recommended items including similarity and feature type matches.

2.5 Annotation recommender

The annotation recommender is a utility for annotating items quickly. It shows a list of feature types that the current item has not yet been annotated with. Internally, for generating this list of feature types, item recommender output is reused. The annotations present for recommended items are compared with the annotations present for the current item. Any feature types for that a statement exists for the recommended item but not for the current item is presented to the user for quick annotation (Figure 4, right side). Therefore, annotating using the annotation recommender quickly improves the quality of results given by the item recommender.

Algorithm 2.2: GetRecFeatureTypes($curItem, recItems$)

for each $i \in recItems$
$recFeatureTypes.add(getFeatureTypes(i) \backslash getFeatureTypes(curItem))$

2.6 Expert recommender

In Figure 4 (left side), the HTML interface representing the expert recommender is shown. The expert recommender works on a per-feature type basis, recommending users who expressed similar opinions concerning a selected Skipforward feature type compared with the current user ($sim_t(u_x, u_y)$ in Formula 1). Since the current user agreed with the user *Durham* and disagreed with TV Tropes (cf. Figure 2), TV Tropes was assigned a smaller weight for aggregation of the feature type *Battle Butler* than *Durham*.

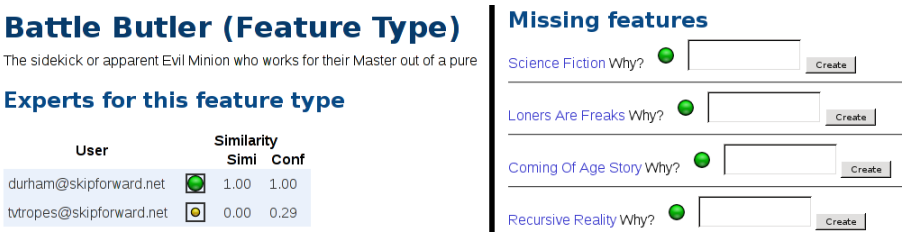

Fig. 4. Expert recommender for one feature type (left) — list of feature types recommended to annotate the current item with (right)

3 Related and Future Work

Skipforward is a unique amalgam of different technologies. Part of its functionality can be found in other systems; for example, *Revyu.com* [2] allows users to submit reviews which can be tagged with keywords. Absolute ratings can be given to items. Metadata is available as RDF/Linked Data; however, the tagging-based approach gives relatively shallow metadata only. In contrast to Skipforward, (formalized) discussions about annotations are not supported, and there is no personalization.

DBin [8] is similar to Skipforward but more generic and heavyweight. For example, it comes with its own messaging API, a plug-in architecture for its user interface, and needs dedicated metadata servers and a Java client whereas in Skipforward no server component is needed.

In terms of recommendation functionality, Skipforward implements a semantic hybrid filtering model. Similar approaches are discussed in [5] (the recommendation channels of Skipforward are similar to the *Collaboration via content* approach outlined in that paper) and [1] (clustering users based on domain concepts they are interested in—possible but not implemented in Skipforward yet).

Most tasks we want to pursue in the future are concerned with improving annotations and providing better recommendations. According to [5], recommender

approaches similar to those used in Skipforward in general cope well with annotation sparsity. However, the current implementation of the annotation recommender does not encourage overlapping annotations. I.e., several users should create feature instances for the same feature type and item to supply the competence metric with input, but the annotation recommender does not address this currently. We addressed this problem by using feature inference so far (i.e., inference using the feature type hierarchy is carried out), but this does not completely solve the problem. We plan to modify the annotation recommender accordingly. Another approach would be to introduce another recommender that explicitly targets annotation overlap to improve the competence metric.

A user study in the books domain will be carried out soon. A number of user interface improvements and additional statistics and recommender functionality will be added during that course.

4 Acknowledgments

This work has been supported by the German Federal Ministry of Education and Research (BMBF) in the context of the iGreen project (01IA08005A).

References

1. Cantador, I., and Castells, P. Multilayered semantic social network modeling by ontology-based user profiles clustering: Application to collaborative filtering. 2006, pp. 334–349.
2. Heath, T., and Motta, E. Revyu.com: A reviewing and rating site for the web of data. 2008, pp. 895–902.
3. Kiesel, M., and Grimnes, G. A. DBTropes—a linked data wrapper approach incorporating community feedback. In *EKAW 2010 Demo and Poster Abstracts. International Conference on Knowledge Engineering and Knowledge Management (EKAW-10), 17th International Conference on Knowledge Engineering and Knowledge Management, October 11-15, Lisbon, Portugal* (10 2010), J. V. O. Corcho, Ed., -. Best Poster.
4. Kiesel, M., and Schwarz, S. Skipforward—a lightweight ontology-based peer-to-peer recommendation system. In *International Semantic Web Conference (Demo)* (2008), C. Bizer and A. Joshi, Eds., vol. 401 of *CEUR Workshop Proceedings*, CEUR-WS.org.
5. Pazzani, M. J. A framework for collaborative, content-based and demographic filtering. *Artif. Intell. Rev. 13*, 5-6 (1999), 393–408.
6. Peis, E., del Castillo, J. M. M., and Delgado-López, J. A. Semantic recommender systems. analysis of the state of the topic. online, 2008.
7. Shardanand, U., and Maes, P. Social information filtering: algorithms for automating "word of mouth". In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1995), ACM Press/Addison-Wesley Publishing Co., pp. 210–217.
8. Tummarello, G., and Morbidoni, C. Collaboratively building structured knowledge with dbin: from del.icio.us tags to an rdfs folksonomy. *Workshop on Social and Collaborative Construction of Structured Knowledge at 16th International World Wide Web Conference (WWW2007)* (2007).