An Autocomplete Input Box for Semantic Annotation on the Web

Tuan-Dat Trinh, Peter Wetz, Ba-Lam Do,
Peb Ruswono Aryan, Elmar Kiesling, and A Min Tjoa

TU Wien, Vienna, Austria
{tuan.trinh,peter.wetz,ba.do,
peb.aryan,elmar.kiesling,a.tjoa}@tuwien.ac.at

Abstract. A large share of websites today allow users to contribute and manage user-generated content. This content is often in textual form and involves names, terms, and keywords that can be ambiguous and difficult to interpret for other users. Semantic annotation can be used to tackle such issues, but this technique has been adopted by only a few websites. This may be attributed to a lack of a standard web input component that allows users to simply and efficiently annotate text. In this paper, we introduce an autocomplete-enabled annotation box that supports users in associating their text with DBpedia resources as they type. This web component can replace existing input fields and does not require particular user skills. Furthermore, it can be used by semantic web developers as a user interface for advanced semantic search and data processing back-ends. Finally, we validate the approach with a preliminary user study.

1 Introduction

The interaction paradigm on the world wide web has changed dramatically in recent years. Users are no longer limited to receiving information passively; rather, they often create and manage their own textual content. A lot of such user-generated content is in raw text format, which is typically entered via input fields (e.g., via the html *textarea* element). In various contexts, users find it difficult to understand each others' terms and expressions due to language ambiguities and inconsistent terminology. Adding annotations such as names, attributes, comments, descriptions, etc. to a selected part of the text allows users to more explicitly express semantics and increase the precision of the statements made.

TripAdvisor[1], for instance, provides more than 200 million traveler-created reviews, opinions, and photos of tourist attractions and accommodations. Many travelers use TripAdvisor to obtain information on points of interest such as parks, museums, historic buildings, streets, etc. To make decisions about what places to visit and to plan a trip, travelers spend a lot of time in search for additional information on interesting places. This search for context information

[1] `http://www.tripadvisor.com/` (accessed 20 July 2015)

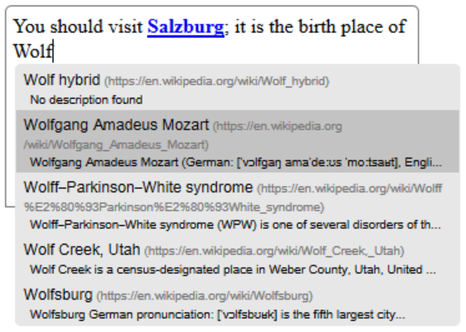
could be much more efficient if concepts in travel-related text were associated with the respective DBpedia content. We will illustrate some travel-related use cases in Section 3.

Although annotations can clarify meaning, provide a context for textual descriptions, and hence facilitate search and automatic data processing, most websites currently do not use them. This can at least partly be explained by the lack of a suitable input component that can easily be integrated into existing web sites. In this paper, we introduce a simple to use autocomplete-enabled annotation box as a replacement for traditional HTML textarea. Using this box, users can quickly and easily enrich selected text with semantic annotations (i.e., DBpedia resources). The component is written in Java Script and can easily be integrated by developers into their websites.

The remainder of this paper is organized as follows. We introduce the annotation box and its implementation and workflow in Section 2. Section 3 analyzes how users and developers can benefit from the features provided by the box, and Section 4 presents our user study to validate the approach. Finally, we discuss related work and draw conclusions in section 5.

2 Autocomplete Text Annotation

Our annotation input box looks and behaves similar to a standard HTML text input. Behind the user interface, however, there is a small processor that allows users to quickly annotate text. To create an annotation, users invoke the processor by pressing *Ctrl + Space*. Next, the processor uses the selected words to generate a query that returns a list of relevant resources (cf. Fig. 1). Users can then select an item from the results to associate their word with the respective resource. To annotate a compound word, users select and highlight related words and press *Ctrl + Space*.



You should visit **Salzburg**; it is the birth place of Wolf

Wolf hybrid (https://en.wikipedia.org/wiki/Wolf_hybrid)
No description found

Wolfgang Amadeus Mozart (https://en.wikipedia.org/wiki/Wolfgang_Amadeus_Mozart)
Wolfgang Amadeus Mozart (German: [ˈvɔlfgaŋ amaˈdeːʊs ˈmoːtsaʁt], Engli...

Wolff–Parkinson–White syndrome (https://en.wikipedia.org/wiki/Wolff%E2%80%93Parkinson%E2%80%93White_syndrome)
Wolff–Parkinson–White syndrome (WPW) is one of several disorders of th...

Wolf Creek, Utah (https://en.wikipedia.org/wiki/Wolf_Creek,_Utah)
Wolf Creek is a census-designated place in Weber County, Utah, United ...

Wolfsburg (https://en.wikipedia.org/wiki/Wolfsburg)
Wolfsburg German pronunciation: [ˈvɔlfsbʊʁk] is the fifth largest city...

Fig. 1: Autocomplete-enabled annotation box

To provide suitable suggestions, we need a common dataset to look up resources from the given text. Because the expected user content and accompanying contexts are arbitrary, the dataset used should cover various domains. To query for related resources, we therefore use DBpedia [1], which can be considered the central hub of the LOD cloud.

From a Wikipedia page we can obtain the corresponding DBpedia resource and vice versa. For instance, the matching DBpedia resource for *https://en.wikipedia.org/wiki/Swimming_pool* is *http://dbpedia.org/resource/Swimming_pool*. As a consequence, we can use either DBpedia or Wikipedia APIs to look up and suggest resources from the given text to users. Available options are (i) DBpedia

Lookup[2], (ii) DBpedia Faceted Search[3], (iii) Wikidata API[4], (iv) and Wikipedia API[5]. We performed an informal evaluation in order to decide which of these APIs supports us in finding relevant LOD resources. As a result of our experiments, we decided to use the Wikipedia API. We found that compared to the others, this API responds faster and returns more relevant resources.

The annotation box is an HTML5 *div* area whose *editableContent* attribute is set to *true*. To enforce compatibility with various browsers, we had to deal with a number of technical challenges. For example, when we edit the text and press enter to go from one line to the next, Firefox will add a *br* element to the *div*, while Internet Explorer and Chrome use *p* and a sub *div*, respectively. We also needed to standardize the final content of the box so that it can be saved consistently into a data base. It consists of *div* elements where each represents a line in the box; each *div* element contains only *span* and *a* elements for unannotated and annotated text, respectively. This structure allows developers to easily extract DBpedia resources from the input content to perform further processing in the back-end. Other implementation challenges were: (i) The box processor needs to calculate the correct location of the text cursor in pixels to present the resource list at a reasonable position (i.e., right below the first character of the selected word). (ii) Allowing to replace selected text by the selected link leading to the desired LOD resource.

3 Benefits of the Annotation Box

The annotation box supports two major use cases: (i) to replace HTML text controls in a data input form, and (ii) to be used as a search box – an element often required on web pages. In both cases, it reduces the ambiguity of natural language text and helps the processor (i.e., the back-end of websites) or other users to correctly understand a term.

Annotations associated with text lay the foundation for automatic data processing in the back-end. Suppose that TripAdvisor makes use of our annotation box; it could then easily extract annotated geographic places, historic buildings, parks, or museums from users' collected comments and reviews. This would, for instance, make it possible to perform evaluations on the popularity of these entities in different seasons of the year. The knowledge gained in the process could be used to automatically suggest users the best time period to travel to a given city, or list the most popular places that travelers should visit.

Moreover, the box facilitates search. It enhances precision of the result, because besides text matching, it allows for LOD resource mappings in the back-end by leveraging *owl:sameAs* properties.

A website equipped with annotation boxes is capable of performing queries on top of LOD resources. For example, if every geographic place listed in the

[2] `https://github.com/dbpedia/lookup` (accessed 20 July 2015)
[3] `http://dbpedia.org/fct/` (accessed 20 July 2015)
[4] `https://www.wikidata.org/w/api.php` (accessed 20 July 2015)
[5] `https://en.wikipedia.org/w/api.php` (accessed 20 July 2015)

travel guides of a city would be associated with DBpedia resources, we can access these resources and get the construction year of each place. From that, we can calculate the average age of every city spot mentioned in a travel guide. Assume that a traveler visits the TripAdvisor website to find a city in a country to visit. We can then ask users for their preference (i.e., modern or antique city) to infer and recommend the most appropriate place for them.

Finally, the annotation box is simple to use. It requires no additional technical expertise, and takes end users only a few seconds to enrich their words with DBpedia resources. Developers can replace the traditional HTML input with the annotation box and empower the back-end with intelligent search and automatic data processing. The box is written in JavaScript and published on GitHub[6].

4 User Study

We conducted a small-scale user study to evaluate the annotation box. We chose eight subjects aged between 25 and 35; all generally spend at least one hour per day on the internet and can be characterized as being experienced in working with computers. We explained to the subjects how words can be annotated by using the box. After that, we asked the subjects to perform three tasks of increasing complexity as follows. (i) Simple annotation. The subjects annotate predefined text, that is: "*Vienna is the capital of Austria*". As they type, they are asked to annotate Vienna and Austria with the respective Wikipedia resources. (ii) Compound word annotation. We present the subjects a predefined sentence, that is "*Do not confuse it with another Vienna, which is a town in Virginia, United States*". They are asked to annotate two single words (i.e., Vienna and Virginia) and a compound word (i.e., United States). (iii) Single and compound word annotation. The subjects are asked to annotate all places mentioned in the following sentence: "*When travelling to Vienna, Austria, we visited Rathaus, Graben, St. Stephen's Cathedral, Vienna Ring Road, Hundertwasserhaus, Hofburg Palace, Schönbrunn Palace, Belvedere, Naschmarkt*". If some place is not available in the list containing suggested annotations, the subjects should manually search and tie the Wikipedia link to the text.

Fig. 2 and 3 show the results of the experiment. While the first two tasks are designed to help the subjects getting used to the box, we mainly use the third to evaluate the usability of the box. It contains eleven terms needed to annotate. The subjects needed 138 seconds on average to complete this task, which means they spent 12.5 seconds per term. Most of the spent time is used for locating the relevant resources. The task completion time plot leads us to the conclusion that adding annotations to text can be done with little effort even for a high number of annotations. Moreover, the time needed for completing the tasks does not vary significantly between the subjects, indicating that the process of creating annotations is efficient and straightforward. The subjects typically made one mistake in the second task as they confused *Vienna, Austria* with

[6] `https://github.com/datsat/Annotating-Box` (accessed 20 July 2015)

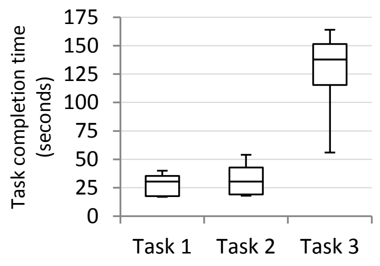


Fig. 2: Task completion time

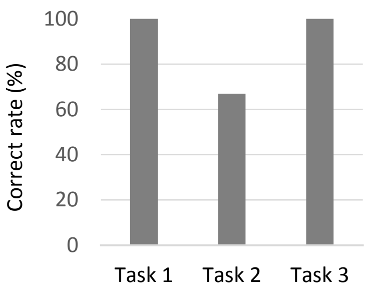


Fig. 3: Median correct rate

Vienna, USA. This can be seen in the bar chart for Task 2 where the median of correct annotations is at 67%. For the other tasks the median is at 100%.

At the end of the experiment, we asked the subjects to fill out a questionnaire in order to get feedback on the usability experience. They agreed that the box is relatively easy to use, but it would be easier if they could initiate the annotation process by right clicking or by pressing a single key. They also suggested to improve the process of locating resources which are not in the suggestion list. For instance, they would prefer not having to manually search and add the link to the text. The subjects enjoyed reading the annotations as it provided them additional information. However, they would only add annotations to their posts if it is really necessary, because it requires time and effort. All in all the results of this preliminary user study validate the efficiency and usability of the annotation box. Based on the feedback, we plan to further improve its usability and functionality.

5 Related Work and Conclusion

In the semantic web community *semantic annotation* has been an active field of research for several years. Semantic annotation describes a more granular approach than the tagging or the use of folksonomies [3] (i.e., a system of classification derived from collaborative or social tagging). The latter approaches assign keywords or terms to a whole document, speeding up search and helps us to find relevant and precise information. In contrast, semantic annotation enriches a word or a part of a document with context that is further linked to structured knowledge (e.g., a DBpedia page that provides information on a resource). Annotations are more informative than tags and allow to show results that are not explicitly related to the original search.

RDFace[7] is an online RDFa content editor that uses existing semantic web APIs to help users manage and embed RDFa contents to a web article. Users can manually add a triple, or simply select one or more NLP APIs to perform automatic named entity extraction [2]. Compared to enriching text with LOD resources using our annotation box, RDFa annotation requires specialized knowledge, meaning that users need to be familiar with semantic web concepts.

[7] `http://wiki.aksw.org/Projects/RDFaCE` (accessed 20 July 2015)

RDFace is appropriate for skilled users in a web content management system; However, it is not relevant for general users to quickly create simple web content (e.g., comments or posts). WYMeditor[8] is another WYSIWYM (What-You-See-Is-What-You-Mean) tool that allows for editing RDFa content; the functionality, however, seems not to be complete and development is already discontinued.

PoolParty thesaurus [9] is an example for a similar product that is already commercially available. It is a WordPress plugin that allows users to import a controlled vocabulary or retrieve a thesaurus from a SPARQL endpoint. It automatically analyzes a post to find words and phrases that match labels of a concept in the thesaurus. When hoovering annotated texts it displays respective tooltips. PoolParty also developed similar plugins for SharePoint and Drupal. The differences between PoolParty and our annotation box are: (i) PoolParty employs a high-level thesaurus whereas our box makes use of DBpedia resources. (ii) PoolParty only annotates text that matches limited concepts of the thesaurus whereas we annotate arbitrary text to DBpedia resources. (iii) We focus on real-time annotating while PoolParty annotates the whole text content after it is created. (iv) Because of the large number of DBpedia resources, we support manual annotating to enhance precision; the autocomplete feature will compensate the annotating time. Meanwhile, PoolParty implements an automatic approach where text is annotated without users interaction.

The idea of an autocomplete annotation box can also be found in services such as Facebook[10], ChatGrape[11] or Slack[12]. Users, when inputting text for a post or a chat message, can link text to resources such as friends, cloud documents, or calendar entries. However, these applications and their input boxes do not make use of semantic annotation; the added context is limited to their own resources.

To conclude, in this paper, we introduce an autocomplete-enabled annotation web component that can replace html text input areas. It enriches user-generated content with annotations of DBpedia resources to facilitate use cases such as semantic search and automatic data processing in the back-end.

References

1. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web 6(2), 167–195 (2015)
2. Mihalcea, R., Csomai, A.: Wikify!: Linking Documents to Encyclopedic Knowledge. In: Proceedings of the 16th ACM Conference on Information and Knowledge Management. pp. 233–242. CIKM '07, ACM, New York, NY, USA (2007)
3. Peters, I., Becker, P.: Folksonomies: Indexing and Retrieval in Web 2.0. Knowledge & information : studies in information science, De Gruyter/Saur (2009)

[8] `http://wymeditor.github.io/wymeditor/` (accessed 20 July 2015)
[9] `https://wordpress.org/plugins/poolparty-thesaurus/` (accessed 20 July 2015)
[10] `https://facebook.com/` (accessed 20 July 2015)
[11] `https://chatgrape.com/` (accessed 20 July 2015)
[12] `https://slack.com/` (accessed 20 July 2015)