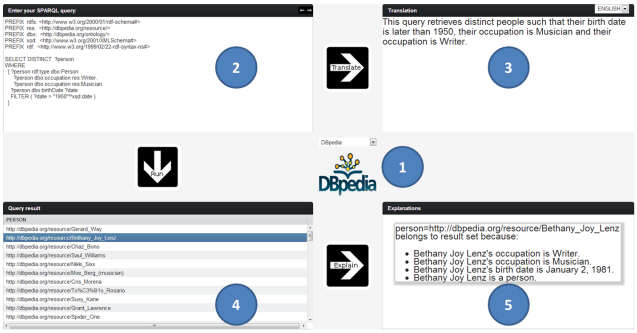

Figure 3: Screenshot of the SPARQL2NL online demo at `http://sparql2nl.aksw.org/demo`.

Consequently, we will demo SPARQL2NL both as in a standalone demo as well as integrated in a question answering tool.

The standalone part of the demo consists of the interface shown in Figure 3. The user begins by selecting the dataset against which he wants the SPARQL query to be ran (see ①). In our demo, we show results on the DBpedia and MusicBrainz Datasets. Now in ②, the user can give in a SPARQL query for which a verbalization is required. For example, he might choose to give in the query shown in Listing 1.

Clicking on the "Translate" button leads to the SPARQL2NL being ran and the query being verbalized. The verbalization of query is shown in ③. In our example, SPARQL2NL returns the verbalization shown in Listing 2.

In addition to viewing the verbalization of the query, the user can choose to run the query by clicking on the "Run" button. This leads to the results of the query (as far as some exist) being displayed in tabular format in ④. Each of the rows of the table is clickable. Upon a selection of a result and a click on the "Explain" button, the RDF statements that led to the row being included in the result set are retrieved from the endpoint selected by the user. These results are verbalized by SPARQL2NL. Verbalizing RDF triples makes use of the fact that each RDF statement can be regarded as a variable-free triple pattern. The verbalized RDF triples are finally displayed in the panel marked with ⑤. In our example, the verbalization of the results for Ian Cairns leads to the following explanatory statements:

- `Bethany Joy Lenz's occupation is Writer.`
- `Bethany Joy Lenz's occupation is Musician.`
- `Bethany Joy Lenz's birth date is January 2, 1981.`
- `Bethany Joy Lenz is a person.`

In addition to the standalone demo, we will present the benefits of SPARQL2NL by showing its integration into TBSL [5] as shown in Figure 4. Here the user can give in a natural-language question such as for example `Books written by Dan Brown` in the search field (①). TBSL then generate possible interpretations of user query in the form of several SPARQL queries. In the case of our example query, TBSL generate semantically very different SPARQL queries due to `written by` and `Dan Brown` each matching several resources from DBpedia. TBSL uses SPARQL2NL to verbalize each of the interpretations it generates and displays as well as tries out the highest scored interpretation first (see ②). If the interpretation is incorrect, the user can choose to click on the "Wrong!" button to see alternative interpretations of his natural-language query, which are displayed in the "Did you mean?" box (see ③). He can then select the natural-language representation that is most accurate and run this query without ever having to deal with SPARQL or RDF.

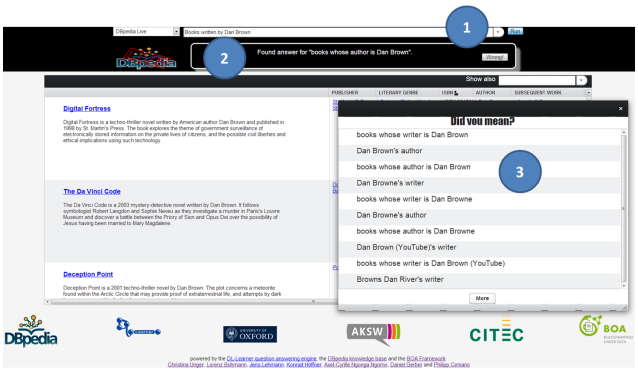

Figure 4: Screenshot of the TBSL online demo at `http://autosparql-tbsl.dl-learner.org/`.

5. REFERENCES

[1] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of HLT*, pages 138–145, 2002.

[2] Jens Lehmann and Lorenz Bühmann. Autosparql: Let users query your knowledge base. In *Proceedings of ESWC 2011*, 2011.

[3] Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. Sorry, i don't speak sparql – translating sparql queries into natural language. In *Proceedings of WWW*, 2013.

[4] Saeedeh Shekarpour, Sören Auer, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, Sebastian Hellmann, and Claus Stadler. Keyword-driven sparql query generation leveraging background knowledge. In *ACM/IEEE WI*, 2011.

[5] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over RDF data. In *Proceedings of WWW*, 2012.