Extracting Literal Assertions for DBpedia from Wikipedia Abstracts

Florian Schrage(✉), Nicolas Heist, and Heiko Paulheim

Data and Web Science Group, University of Mannheim, Mannheim, Germany
florian.schrage@sap.com,{nico,heiko}@informatik.uni-mannheim.de.de

Abstract. Knowledge Graph completion deals with the addition of missing facts to knowledge graphs. While quite a few approaches exist for type and link prediction in knowledge graphs, the addition of literal values (also called instance or entity attributes) is not very well covered in the literature. In this paper, we present an approach for extracting numerical and date literal values from Wikipedia abstracts. We show that our approach can add 643k additional literal values to DBpedia at a precision of about 95%.

Keywords: Knowledge Graph · Completion · Literals · DBpedia

1 Introduction

In the past, adding missing facts to Knowledge Graphs to increase the data quality in knowledge graphs has gained a lot of attention [10]. Most prominently, link prediction using embedding models is a very active field of research [14].

While a lot of research is devoted on missing links, i.e., relations between two entities, the prediction of missing facts involving literals (e.g., numbers or dates), is considerably underrepresented in the current research landscape [10].

In this paper, we aim at closing this gap by identifying and extracting literal values from abstracts in Wikipedia[1]., defining an abstract as the In contrast to standard relation extraction, there are a few additional challenges to face:

- Natural text uses a lot of different number formats (e.g., w.r.t. decimal and thousands separators) [15]. Even within a single Wikipedia article, the number formats may be inconsistent [11].
- Numbers often come with units of measurement, which complicate the extraction, since those units need to be harmonized [13].
- Exact numbers are often rounded in natural text (e.g., *about 3,000* instead of *3,085*, which can make it difficult to assess whether a rounded and an exact number refer to the same or a different fact.

The contribution of this paper is an approach for extracting literal values (numbers and dates) from Wikipedia articles, which can deal with roundings and different units of measurement.

[1] We follow the *long abstract* notion in [9], extracting the "'text before a table of contents"' from a Wikipedia page.

M. Acosta et al. (Eds.): SEMANTiCS 2019, LNCS 11702, pp. 288–294, 2019.
https://doi.org/10.1007/978-3-030-33220-4_21

2 Related Work

There are not many approaches for completing literal values in knowledge graphs. In [12], the use of Web tables as a source for literal values is explored. This setting is different, since the authors work on structured, not unstructured data, so they can use different features.

One of the few closer approaches is presented in [2], where the authors run open relation extraction on Wikipedia text and perform an a posteriori mapping to the DBpedia ontology. Although they do not deal with numbers, their approach can extract date literals and reaches an overall precision of 74.3%[2].

Similarly, the authors of [1] train specific models for DBpedia relations, but only evaluate their approach on two date-valued and one integer-valued relation. They extract 440k date valued literals (birthdate and deathdate) at a precision of 91%, and 237k integer-valued literals (population) at a precision of 70%[3].

In comparison to that state of the art, the approach discussed in this paper yields superior results both in absolute numbers of values extracted, as well as in precision.

For error detection in knowledge graphs, there are approaches based on outlier detection [3], probabilistic data modeling [7] and data fusion [8]. There, it can also be observed that the amount of research directed towards literal values is much underrepresented in comparison to relation assertions between individuals.

3 Approach

Our approach builds on previous works for extracting relation assertions from Wikipedia abstracts [4,5]. That approach exploits *links* in Wikipedia abstracts, learns characteristic patterns for relations (e.g., *The first place linked in the*

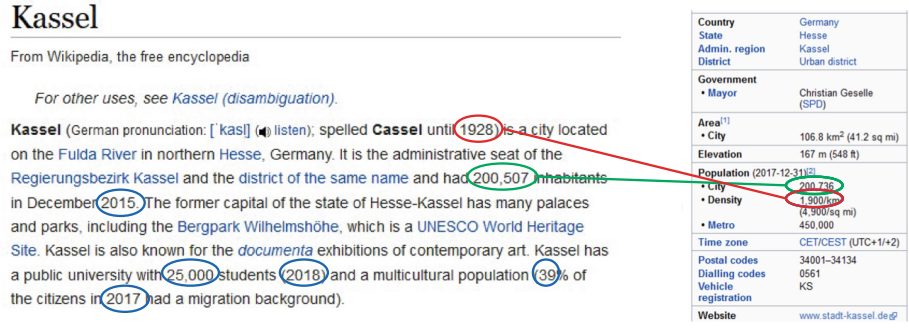

Fig. 1. Example from Wikipedia with a correct and an incorrect example extracted, as well as non-matching literals marked in the abstract.

[2] Their final dataset contains about 5k date literals mapped to properties in the DBpedia ontology.

[3] The totals include both literals contained and not contained in DBpedia.

Wikipedia abstract about a person is that person's birthplace), and then applies the models to extract new statements (e.g., new facts for the relation *birthplace*). For each relation, a separate model is trained and validated on the existing instances, which allows for only applying models that achieve a desired precision.

3.1 Training Data Creation

To create the training data, we use regular expressions to detect and parse numbers in the abstract in various formats (i.e., thousands and decimal separators), and SpaCy[4] and dateparser[5] to detect and parse dates.

With these approaches, we extract the sets of numerical literals N and date literals D from the Wikipedia abstract describing a DBpedia entity e. Since numbers may be rounded, we accept a training example $n \in N$ as positive example for a relation if there is a statement $r(e, v)$ in DBpedia with $n \in [v \cdot (1-p), v \cdot (1+p)]$ for a deviation factor of p. We manually examined candidates drawn at deviation factors of 1%, 1.5%, and 2%, and observed that the precision at 1% and 1.5% was 65%, and dropped to 60% when further increasing the deviation factor. Hence, we decided to use a factor of 1.5% in our further experiments.

Figure 1 illustrates this generation of examples. Since DBpedia is constructed from infoboxes in Wikipedia, the values in the infobox on the right hand side correspond to the values in DBpedia. Given the Wikipedia abstract, 200,507 would be extracted as a training example for the relation *population* (correct), while 1928 would be extracted as a training example for the relation *density* (incorrect). The deviation is 0.11% and 1.47%, respectively.

Since dates are not rounded, training data for date valued literals are based on exact matches with DBpedia only[6].

As negative training examples, we use all numbers or dates, respectively, which have been tagged in the abstract which are not identified as positive examples for the relation at hand. In the example depicted in Fig. 1, we would use all numbers except for 200,507 as negative training examples for the relation *population*.

3.2 Unit Conversion

An initial look at the training data revealed that this approach misses quite a few numerical training examples, since the units of measurement in which the facts are stored are often different from the ones in the abstracts. For example, areas (of countries, cities, ...) are stored in DBpedia in square meters, while they are typically written in square kilometers or non-metric units. Therefore, for those relations, the training data sets create are often very small (e.g., for *area*, which is one of the most frequent relations in DBpedia, we initially collected less than 100 training examples).

[4] https://spacy.io/.
[5] https://pypi.org/project/dateparser/.
[6] Note that it is not trivial to detect that *1928* in the text is a date, not an integer.

Table 1. Examples for unit conversions learned from the data.

Token	Target unit	Correct factor	Inferred factor	R squared
km^2	m^2	1,000,000	997,097	0.9949
$km2$	m^2	1,000,000	999,927	0.9999
ha	m^2	10,000	9,467	0.8987
pupils	$	–	13,613	0.9062
kilometers	*m*	1,000	973	0.9347
century	*m*	–	73,453	0.9421

Therefore, we decided to enhance the training example generation with unit conversion. We follow the assumption that (1) units of measurement are typically the token after a number[7], and (2) the function for converting units to their standard unit in DBpedia is usually a simple multiplication by a factor. Thus, we group numeric literals for each relation by the token following the number (e.g., *ha*) and try to learn a regression model for that token. From those regression models, we derive unit conversion rules which are applied to the literals extracted as above before mapping them to relations in DBpedia. Following an initial inspection of the data, we accept unit conversions learned on at least 100 examples and having a coefficient of determination of at least 0.85. Table 1 shows a few example unit conversion factors, including useful rules learned, but also some misleading rules (e.g., converting the "unit" *pupils* to *$*).

3.3 Feature Extraction

For each positive and negative training example extracted, we create a set of features to feed into a classifier. We use a similar set of features as in [4], e.g., position in the sentence, position of the sentence in the abstract, etc., plus a bag of words representation of the sentence in which the literal is located, and, for numerical literals, the deviation from the mean divided by the standard deviation of all values of the respective relation, in order to discard outliers.

3.4 Model Building

To learn models given the examples and feature vectors, we experimented with different classifiers from the scikit-learn library[8], i.e., SGD, Naive Bayes, SVM, Decision Trees, Random Forest, Extra Trees, Bagging Decision Trees, and XGBoost. Out of those, the latter five delivered the best results in an initial experiment (using split validation on a sample of relations with the most already existing instances), without much variance in quality. Random Forests were chosen because of a good trade-off between runtime and accuracy.

[7] There are rare exceptions, like currencies, which we ignore.
[8] https://scikit-learn.org/.

4 Evaluation

For our evaluation, we used the most recent downloadable version of DBpedia, i.e., DBpedia 2016-10[9] and the abstracts contained therein[10]. We tried to train models for all 405 number and date valued properties in the DBpedia ontology. To learn meaningful models, we discarded all properties that were too small (i.e., less than 100 positive examples), leaving us with 120 properties.

Following the approach in [4], we aimed at achieving a high precision in the extraction in order not to add too much noise to the knowledge graph at hand. Therefore, we validated all models internally using a training (75%) and a test (25%) split, and kept only those models achieving a precision of at least 95%. Out of those 120 properties, we could learn a model at 95% precision in 28 cases.

As shown in Table 2, for those 28 relations, the approach creates almost 9M statements, however, only a smaller fraction (about 7%) is not yet contained in DBpedia. That share of new statements is considerably higher for dates (11%) than for numbers (less than 1%). The majority of the former are birthdates, the majority of the latter are population numbers.

Table 2. Number of statements extracted at 95% precision according to internal validation.

Range	Properties	Statements	New statements
Date	17	5,525,089	621,747
Int	6	224,606	15,326
Float	5	3,185,497	5,955
Total	28	8,955,192	643,030

In order to validate the precision values of the internal validation based on the test set, we randomly sampled 500 of the new statements for manual inspection. This inspection yields a precision of 94.2%, which confirms the estimation based on the internal test set.

In terms of runtime, a complete run on the entire DBpedia and the corresponding Wikipedia abstracts takes about 135 h on a Linux server with 512 GB of RAM. The by far longest time is consumed by the preprocessing of the abstracts, e.g., the date tagging and parsing takes 65 h alone, whereas the model training and statement creation take 1.9 and 3.6 h each.

[9] https://wiki.dbpedia.org/downloads-2016-10.
[10] http://downloads.dbpedia.org/2016-10/core-i18n/en/long_abstracts_en.tql.bz2.

5 Conclusion and Outlook

With this paper, we have aimed at closing a gap in the current research landscape on knowledge graph completion. While research in this field is strongly focused on type and relation prediction, we have shown how numeric and date valued facts can be extracted from Wikipedia abstracts. While there are quite a few challenges, including number and date formats and unit conversions, we have shown that it is possible to achieve an extraction at a precision of about 95%. The code used to create the results reported in this paper is available online[11].

In the future, we plan to apply the approach to other Wiki-based knowledge graphs, such as DBkWik [6].

References

1. Aprosio, A.P., Giuliano, C., Lavelli, A.: Extending the coverage of DBpedia properties using distant supervision over Wikipedia. In: Workshop on NLP & DBPEDIA (2013)
2. Exner, P., Nugues, P.: Entity extraction: from unstructured text to DBpedia RDF triples. In: The Web of Linked Entities Workshop (WoLE 2012), pp. 58–69. CEUR (2012)
3. Fleischhacker, D., Paulheim, H., Bryl, V., Völker, J., Bizer, C.: Detecting errors in numerical linked data using cross-checked outlier detection. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 357–372. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_23
4. Heist, N., Hertling, S., Paulheim, H.: Language-agnostic relation extraction from abstracts in Wikis. Information **9**(4), 75 (2018)
5. Heist, N., Paulheim, H.: Language-agnostic relation extraction from Wikipedia abstracts. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 383–399. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_23
6. Hertling, S., Paulheim, H.: DBkWik: a consolidated knowledge graph from thousands of Wikis. In: 2018 IEEE International Conference on Big Knowledge (ICBK), pp. 17–24. IEEE (2018)
7. Li, H., Li, Y., Xu, F., Zhong, X.: Probabilistic error detecting in numerical linked data. In: Chen, Q., Hameurlain, A., Toumani, F., Wagner, R., Decker, H. (eds.) DEXA 2015. LNCS, vol. 9261, pp. 61–75. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22849-5_5
8. Liu, S., d'Aquin, M., Motta, E.: Towards linked data fact validation through measuring consensus. In: Workshop on Linked Data Quality (2015)
9. Morsey, M., Lehmann, J., Auer, S., Stadler, C., Hellmann, S.: DBpedia and the live extraction of structured data from wikipedia. Program **46**(2), 157–181 (2012)
10. Paulheim, H.: Knowledge graph refinement: a survey of approaches and evaluation methods. Semantic Web **8**(3), 489–508 (2017)
11. Paulheim, H.: A robust number parser based on conditional random fields. In: Kern-Isberner, G., Fürnkranz, J., Thimm, M. (eds.) KI 2017. LNCS (LNAI), vol. 10505, pp. 337–343. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67190-1_29

[11] https://github.com/FlorianSchrage/DBpediaLiteralRelations.

12. Ritze, D., Lehmberg, O., Oulabi, Y., Bizer, C.: Profiling the potential of web tables for augmenting cross-domain knowledge bases. In: Proceedings of the 25th International Conference on World Wide Web, pp. 251–261 (2016)
13. Subercaze, J.: Chaudron: extending DBpedia with measurement. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10249, pp. 434–448. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58068-5_27
14. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
15. Wienand, D., Paulheim, H.: Detecting incorrect numerical data in DBpedia. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 504–518. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07443-6_34