Inferring Resource Types in Knowledge Graphs using NLP analysis and human in-the-loop validation: The DBpedia Case

Idafen Santana-Pérez [a,*] and Mariano Rico [b]
[a] *Departamento de Señales y Comunicaciones, Universidad de Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, Spain*
E-mail: idafen.santana@ulpgc.es
[b] *Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain*
E-mail: mariano.rico@upm.es

Abstract. Defining proper semantic types for resources in Knowledge Graphs is one of the key steps on building high quality data. Often, this information is either missing or incorrect. Thus it is crucial to define means to infer this information. Several approaches have been proposed, including reasoning, statistical analysis, and the usage of the textual information related to the resources. In this work we explore how textual information can be applied to existing semantic datasets for predicting the types for resources, relying exclusively on the textual features of their descriptions. We apply our approach to DBpedia entries, combining different standard NLP techniques and exploiting complementary information available to extract relevant features for different classifiers. Our results show that this approach is able to generate types with high precision and recall, above the state of the art, evaluated both on the DBpedia dataset (94%) as well as on the LDH gold standard dataset (80%). We also discuss the utility of the web tool we have created for this analysis, NLP4Types, which has been released as an online application to collect feedback from final users aimed at enhancing the Knowledge graph.

Keywords: DBpedia, Data quality, Knowledge graph, NLP

1. Introduction

Type statements are the most basic and fundamental piece of information for semantic resources [1–4]. They allow us to classify resources with different levels of granularity and to apply semantic restrictions over them. This information can be generated by different means, during the development of semantic datasets, including manual, automated and semi-automated approaches. In this paper we cover DBpedia, one of the main dataset in the context of Linked Data. DBpedia is generated automatically from the information contained in Wikipedia, using a set of mappings, mainly from the entries in tabular format contained in the infoboxes of each Wikipedia page. The information includes types and properties, and it is converted to RDF. As not all pages contain infoboxes it is not always possible to know the type of a resource. According to our calculation, around a 16% of resources from Wikipedia do not have any type mapped to DBpedia (English). Notice that mappings are manually create and many DBpedia datasets barely have mappings or do not have them at all. Therefore this 16% is a lower boundary and most DBpedia chapters have higher values.

We have also to take into account that, even in those cases in which this information can be generated, it is not always correct. As mappings are created collaboratively by users, it is often the case that the types assigned are either incorrect (e.g. classifying *San Francisco* (California) as a *Person*, instead of as a *Place*),

*Corresponding author. E-mail: idafen.santana@ulpgc.es.

or too abstract (e.g. *Cristiano Ronaldo* as an *Athlete*, when it should be stated as a *Soccer Player*).

This information can be inferred and corrected using several approaches, as we discuss later in this paper. Depending on which features are available, different processes can be defined. In this paper we aim to explore how to induce types when there is not structured information available, relying just on the unstructured description of the resource in the form of text. We explore how the textual abstracts from the Wikipedia entries can be exploited using NLP techniques to classify entries into the DBpedia ontology. As our results show, applying a combination of document-to-term matrix and Named Entity Recognition produces a system able to classify resources with high precision and recall (around 95%).

The reminder of this paper is structured as follows. In Section 2 we introduce the main approaches in the field and how they align and differ from the one introduced in this work. Section 3 covers the methodology we have applied, which is evaluated in Section 6, using both DBpedia and a gold standard dataset. Finally, Section 7 discusses about the main advantages and drawbacks of our results and how they can be further improved.

2. Related Work

The identification of the type of resources on large datasets is a widely studied problem that has been addressed during last decade. On this regard, several approaches have been proposed, being the SDType [4] system the most prominent one, specially in the context of DBpedia, in which it has been used as part of the regular data generation process. SDType exploits the statistical information of the distribution of properties over resources and types to infer new statements about the type of a resource. Based on their empirical evaluation, authors conclude that properties targeting a resource (so called, *ingoing* properties) are more useful for inferring types than those with the resource as subject (i.e. outgoing properties). It also shows that the more properties a resource have (i.e. ingoing degree) the more precise results are. As described by authors, their system has been proven to outperform most of the existing systems in the area.

In this same way, more recent studies have proposed the use of machine learning techniques, as we do, for improving the types of resources in a dataset. Different contributions have been explored in this context, using multilabel approaches [5, 6], varying on the algorithms applied and how the training data is selected. In general, all of them define the process of assigning types as a multilabel classification problem, as several types are expected for each resource. Approaches that do not rely on ontologies can be defined as domain-agnostic, having the advantage of being able to work without any context information, but not guaranteeing that the results are consistent with the expected data hierarchy. They also provide worse results than those including the taxonomy information. Many of these approaches are modelled as binary classifiers [7], trained either at the class level or at a general level. Taxonomy-based classification approaches use the ontology to divide the training data into different subsets according to different criteria [8]. According to [9], which provides a comprehensive discussion on the state of the art, four main types of hierarchical multilabel approaches can be defined.

- **Global approach**: in which the system is trained with one single set, including the taxonomy, guaranteeing that for each type assigned, all its parent types in the hierarchy are also included, ensuring taxonomy consistency. Our approach falls in this category.
- **Local Classifier Per Node** (*LCN*): in which binary classifiers for each type. These classifiers are similar to general multilabel classifiers, which is used in the SLCN system [9], providing an scalable manner of generating types.
- **Local Classifier Per Parent Node** (*LCPN*): in which for each type a classifier is generated to classify its direct sub-types, thus only applied to non-terminal nodes. That is, each trained model is able to disambiguate and add more specific knowledge from the next level in the hierarchy.
- **Local Classifier Per Level** (*LCL*): in which a multilabel model is generated for each level of the hierarchy. LCL approaches have not been fully explored, as they do not guarantee taxonomy-consistent results. LCL models generate one type per level, but not necessarily following the classes hierarchy.

Both of these two mentioned systems [4, 9] rely on structured data, extracted from the property statements from the datasets, to infer resource types. Even when this approach is valid and has been proven to be successful, the main goal of our study is to cover also those cases in which such information is not available.

We pursue an approach able to generate types relying only on text (unstructured data) rather than on property assertions (structured data).

In this context, several works have been introduced, exploiting different NLP-based techniques for type assignment based on text. In [10] a hierarchy of Support Vector Machines (hSVM) is introduced for applying lexico-syntactic patterns using a bag-of-words model, extracted from short abstracts and Wikipedia categories. This work extends the Linked Hypernym Dataset Framework [11], by the same authors, for extracting these pattern-based structures. These works introduce also a gold standard dataset, which we use in this paper, to measure the performance of our system and compare it to other existing tools. This gold standard has been produced, as reported by authors, using experts to assign types to a subset of the DBpedia resources.

Finally, it is worth to consider other contributions that provide means for measuring the effectiveness of type prediction systems, where other gold standards have been introduced. Through the OKE challenge series several systems have been proposed [12, 13], as part of their types inference tasks, being evaluated against the gold standards provided by organizers.[1,2] These tasks are intended to generate new classes and align them to existing ones, based the textual description of resources. The related gold standards provides links to such classes and alignments.

In this work, as mentioned before, we focus on the LHD dataset[3], which provides a list of DBpedia resources and a curated list of types from the DBpedia ontology for each one of them. Data is generated using crowdsourcing, reaching sufficient consensus for each type assertion. The gold standard provided are used to evaluate the performance of the LHD system [10] itself, which uses text mining techniques for class inductions, as well as to evaluate how it compares to the aforementioned SDType. Thus, we will use the LHD gold standard to evaluate our system, as it provides means for comparing our contribution to both, hSVM and SDType.

3. Methodology

In this work we explore how text classification techniques can be applied to infer types on DBpedia entries. For this, we have implemented a pipeline in which different NLP techniques are applied. As shown in Figure 1, this process is composed by eight main steps, some of which are optional, allowing us to measure the impact of applying or not some techniques. These are the main features of these steps:

- **Get Abstract text:** get the text from available abstracts. All those resources that do not have an abstract are discarded as they can not be used to train (or test) our system.
- **Named Entity Recognition:** using DBpedia Spotlight [14] the system detects the Named Entities on the text, obtaining their types. The surface form of those entities is simply ignored and only the types (e.g. Person, Organization) are used. These types are codified as classes from different ontologies, including the DBpedia Ontology, Schema, or FOAF. In this paper we only consider types belonging to the DBpedia Ontology. As for the Spotlight configuration, all the requests exposed in this paper are performed with a 0.3 confidence threshold.
- **Text pre-process:** apply several text normalization techniques on the abstracts. These techniques include stop words removal, lemmatization, and stemming. More precisely we have used the English stop words from the NLTK corpus, the WordNet Lemmatizer [4], and the well-known NLTK Porter Stemmer [5] for the stemming process.
- **Data vectorization:** translate textual data into a vector space model, using a Bag of Words approach. To increase the classification capability of our classifier, we apply a TF-IDF metric to get a more discriminatory score of each token in a document. The input data for this process depends on the two previous steps, and whether the Named Enity Recongnition step has been carried out. In general, the vectorization process combines both the text and the types of the Named Entities, adding these types as new words to the input text.
- **Training:** train the classifier, using the vectorized data generated before. Depending on the execution parameters, the amount of data used for training and the amount reserved for testing varies. In this paper we have used a Support Vector Ma-

[1] https://github.com/anuzzolese/oke-challenge
[2] https://github.com/anuzzolese/oke-challenge-2016
[3] http://ner.vse.cz/datasets/linkedhypernyms/evaluation/
[4] https://www.nltk.org/_modules/nltk/stem/wordnet.html
[5] https://www.nltk.org/howto/stem.html

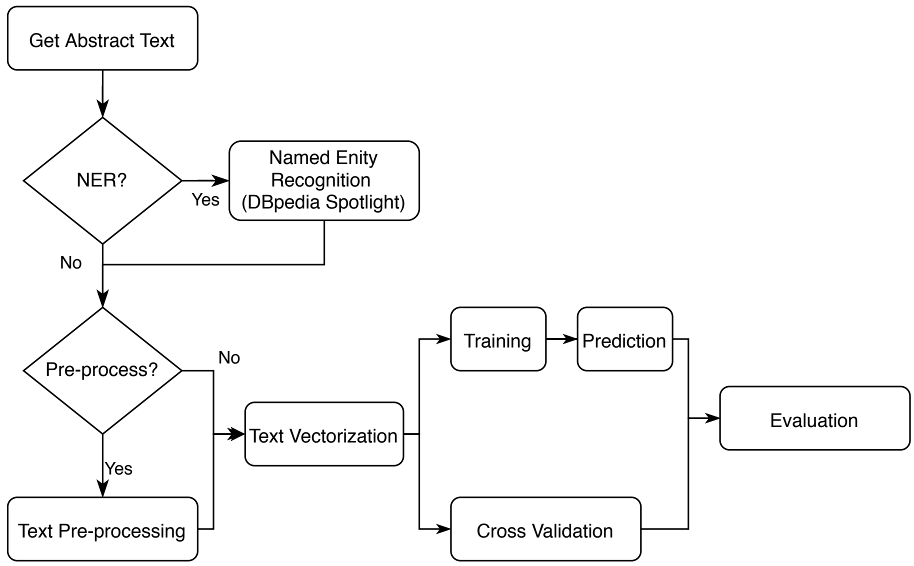


Fig. 1. Overall view of the NLP pipeline.

chine classifier from the scikit-learn python library. Using of SVMs for text classification has been proven to be efficient, performing at the state-of-the-art level[6].

- **Prediction:** once our classifier has been trained, the produced model can be used to predict data. Either test data, reserved during the training phase, or new unseen data, can be vectorized and fed to infer new types. We use these training and prediction steps when validating our approach against the gold standard.
- **Cross validation:** to have a more generalized view of the performance of the system, reducing the overfitting effecting the model, we apply a K-fold cross validation process over the dataset. As we describe later in Section 6, we have applied a 5-fold evaluation, in which the system is trained with 80% of the data and tested with the remaining 20% each iteration.
- **Evaluation:** the final step of the workflow consists on evaluating the results obtained, comparing how the predictions, produced from any of the steps introduced above, fit the labelled data. The metrics for calculating how accurate the system is are introduced in Section 6.1

4. System description

The aforementioned workflow has been implemented in Python, including NLTK [15] and scikit-learn [16] libraries for NLP and machine learning processing. The system has been designed as a data-oriented scientific workflow, in which intermediate results are produced and stored so the execution can be modular and resumed at any point. The code containing all these features is available online[7].

4.1. Architecture

The system can be decomposed is different modules, including the training, prediction and user interface modules. Figure 2 displays the structure of these modules. The main module in the online application is the user interface, which is implementend and hosted using a Flask Server[8]. The usage of this interface is described in more detail in Section 5.1.

The prediction module is the one in charge of ingesting the text written by the user. It applies the same process described in Section 3 to the text, which is always the same used during the training, including the annotation of Named Entities using DBpedia Spoltlight (this is optional both for training and prediction). The trained models, obtained during the training phase, are

[6] https://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-and-machine-learning-on-documents-1.html

[7] https://github.com/idafensp/NLP4Types

[8] https://github.com/pallets/flask

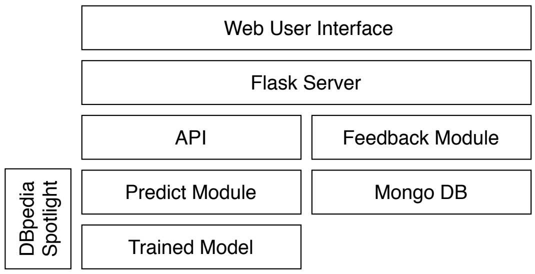


Fig. 2. System architecture.

used by the prediction module to generate the types shown to the user.

On the right hand side of the figure, the feedback module is the one in charge of providing all the options for the user, so he or she can provide feedback on how good the suggested type is. The results of this process would be stored in a Mongo database[9].

The usage of the interface, including the feedback process, is further detailed in the following section.

5. Online application

An online web application has been created aimed at providing a user interface in which anyone can type or paste textual descriptions. The application returns the predicted type. This application is available at http://nlp4types.linkeddata.es. This application uses our best model for predictions, which applies Named Entity Recognition but does not include pre-preoccessing techniques, as discussed in Section 6.

5.1. User interface

Based on the NLP pipeline introduced above, the online interface of NLP4Types allows user to predict types from any free-text sample. The current version of the tool includes a model trained with all the resources from DBpedia 2016-10, which is used to predict types. Currently only types belonging to the DBpedia ontology are predicted.

5.2. User feedback

On of the goals of our tool, beyond providing a service for users to test the results generated by our prediction module, is to obtain information from them regarding how accurate those predictions are. In order to obtain this information, we implemented a set of features to collect feedback from the user. This information could be later used to evaluate the performance of our system, as well as to improve the quality of our models.

Once a prediction is obtained, the user can evaluate the result and provide feedback. As shown in Figure 3, five different criteria are provided, to specify whether the prediction is wrong or right, or how it should be improved. Once the user selects one, it is asked for some extra feedback, including the expected type, user expertise and text source, as shown in Figure 4. The goal of this tool is twofold: allowing the user to interact and test the predictions, as well as to to collect feedback that would allow us to analyze and improve the system and its evaluation.

We are currently on the process of collecting feedback from users, expecting to collect enough information to be used. This information will allow us to understand when and how our system is being to abstract (i.e. inferring types that are too generic) or to concrete (i.e. inferring types that are too precise and wrong). This would lead to new experimentation processes, in which we use modified training datasets, using types that are higher or lower in the DBpedia taxonomy. Studying the implications of these kind of variations could lead to interesting discussion on how to overcome the limitations of working with crowdsourced Knowledge Graphs. As discussed before, improving the quality of these kind of datasets if challenging, as we are limited by the training data contained on them, which usually contains errors introduced by the community annotators.

6. Evaluation

To measure the performance of our system we have implemented two different evaluation setups. The first one is a K-fold evaluation process, using the DBpedia dataset for training and testing. For the second evaluation we have used the aforementioned LHD Gold Standard, to measure how well our model works using external curated data. In both cases we have applied different ways of measuring the performance, using metrics that are more suitable for analyzing the results obtained in a hierarchical scenario, such as the ones in our data.

[9]https://www.mongodb.com/

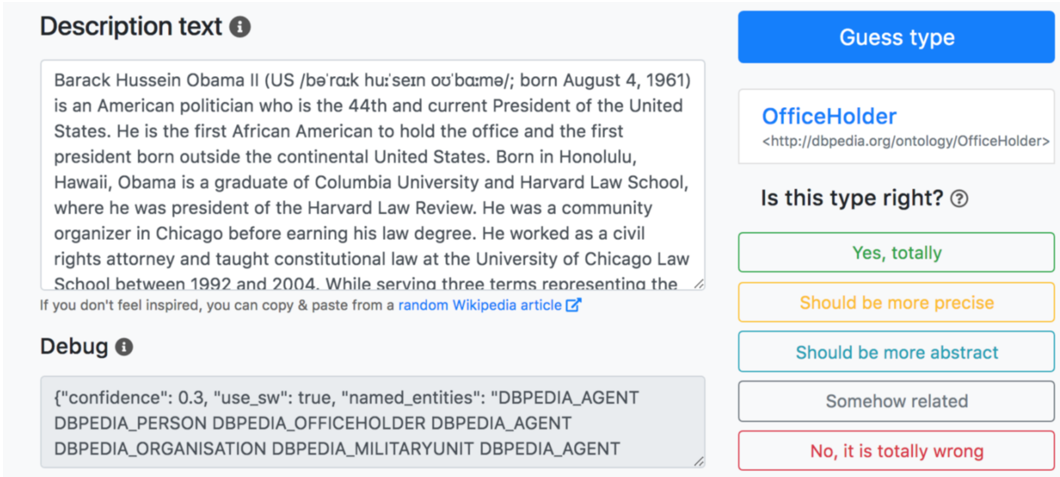


Fig. 3. A snapshot of the Web interface of NLP4Types.

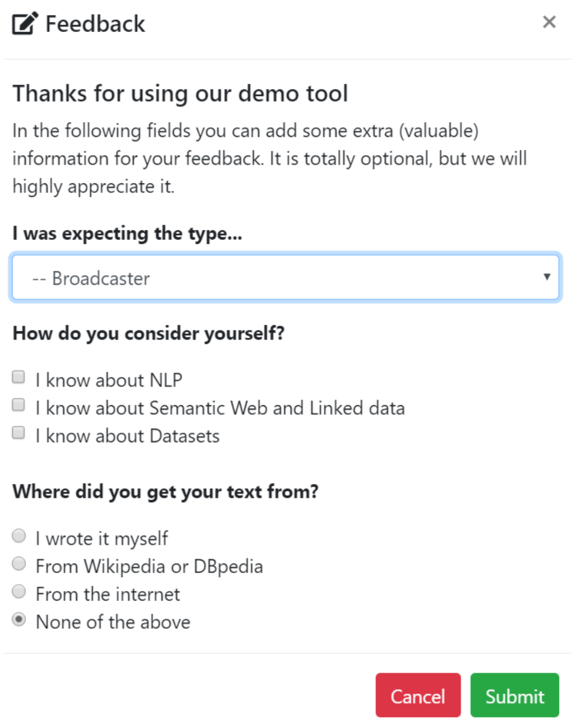


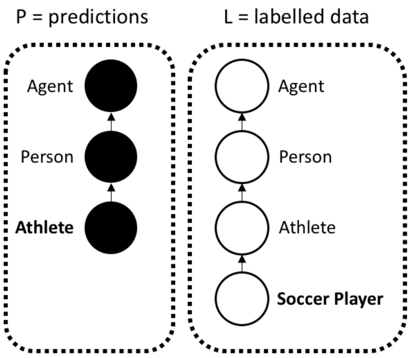
Fig. 4. Feedback panel.

6.1. Metrics

We have selected four main metrics for evaluating our results, aimed at providing better insights on how our classifier works on taxonomical data. In a general classification problem, if the predicted label is not the same as the expected one, the prediction is computed as an error. However, when working with labels structured in a hierarchy, more flexible evaluation metrics can be defined to take this into account. The DBpedia ontology, for example, classifies *Soccer Player* as a subtype of *Athlete*, and this as a subtype of *Person*, which is a subtype of *Agent*. If we consider the resource *Cristiano Ronaldo* as a *Soccer Player* on the labeled data, but the prediction says it is an *Athlete*, it will be counted as a completely erroneous prediction according to the general classification metrics, whereas it is indeed partially right.

This is discussed in [10], where authors define the hierarchical precision, recall, and F-measure metrics to evaluate the performance of different systems over the same gold standard used in our evaluation. Thus, we use these metrics, plus the regular accuracy, to evaluate our system. Figure 5 illustrates graphically how hierarchical precision works.

As shown in the figure, the resource has type as *Soccer Player* on the labelled data, while the system predicts the type *Athlete* for it. From both types we can extract the full type path, containing all parent classes (we have omitted *owl:Thing* for clarity) following the *rdfs:subClassOf* property in the DBpedia ontology. With both type paths we can calculate the hierarchical metrics. In the example, all the types predicted by the system were included in the labelled data, thus is has a perfect hierarchical precision (hP = 1). Concerning recall, the prediction misses the type *Soccer Player*, getting only three out of four hits (hR = 0.75). The hierarchical F-measurement is the harmonic mean on both.



$$hP = \frac{\Sigma |P \cap L|}{\Sigma |P|} \qquad hP = \frac{3}{3} = 1$$

$$hR = \frac{\Sigma |P \cap L|}{\Sigma |L|} \qquad hR = \frac{3}{4} = 0.75$$

$$hF = \frac{2 * hP * hR}{hP + hR} \qquad hF = \frac{2 * 1 * 0.75}{1 + 0.75} = \frac{1.5}{1.75} = 0.86$$

Fig. 5. Formula and examples of hierarchical precision, recall and F-measure. White graph on the right represents labelled data and the corresponding type path. The black one represents the predicted specific type and its type path

6.2. DBpedia Evaluation

We have evaluated the English version of the DBpedia dataset (2016-10 release[10]). The data files used were instance_types_en.ttl, which contains the most specific type for each resource and long_abstracts_en.ttl, containing the abstract text extracted from the corresponding Wikipedia entries. These files contain 5.150.434 types and 4.935.281 abstracts for DBpedia resources. When both files are combined, joining types and abstracts by their associated resource, we obtain a total of 3.048.742 resources. It is worth to clarify that all the resources that are typed only with owl:Thing are not considered, as inferring this type is trivial and does not add any information for the classification problem.

Following the approach described in Section 3, we trained our system using the type and abstract information available. The evaluation comprises a 5-fold process, in which each iteration randomly picks 80% of the data, and reserves 20% for test. The training data is fed to the classifier on the training phase. We then used the trained classifier to predict the types for the test data.

We have executed this approach over three different data sizes, to compare how the performance of the system evolves as more data is available. The results obtained are depicted in Table 1. As we can see, the more data we are able to use for training the system, the more precise it gets, obtaining around a 95% of hierarchical F-measure when using the full training set.

As described in Section 3, the system can be configured to apply different NLP techniques, such as NER and text pre-processing. Table 1 show how our system performs on our best setup. Table 2 shows the results of combining different techniques over the data[11]. The three main elements that can be considered are: (1) the abstract text (ABS), (2) the inclusion of named entity types (NER), and (3) the pre-processing techniques (PP). As shown in the table, the best results (in bold) are obtained when combining abstract and named entities extracted from it, without applying any normalization (pre-processing).

Table 1
5-Fold evaluation results.

Resources	Accuracy	hPrec	hRecall	hF-measure
10K	0,734	0,894	0,886	0,890
1M	0,825	0,944	0,939	0,942
Full (3M)	0,835	0,952	0,949	**0,950**

The use of NER increases the performance of the classifier, although the text of the abstract is the most prominent feature for classifying. If we only use named entities for classifying, not considering the abstract, results drop from a 94% to a 78% hierarchical F-measure (as shown in the the last row of the table), which is reasonable taking into account how much information we are wasting. It is worth to notice that applying pre-processing techniques actually reduces the accuracy of the system. This is most likely due to the fact that processed tokens are more common among different documents, having less discriminatory TF-IDF scores.

6.3. Gold Standard Evaluation

Gold standards provide a benchmark for measuring the performance of a system and how it compares to

[10]http://downloads.dbpedia.org/2016-10/core-i18n/en/

[11]Due to time restrictions, in this paper we have evaluated our approaches with one million entries for this comparison

Table 2

5-Fold evaluation results from different NLP pipelines for one million entries. Plus (+) denote that the technique has been applied, where minus (-) indicates that it has not been applied.

Resources			Accuracy	hPrec	hRecall	hF-measure
ABS	NER	PP				
+	+	-	**0.825**	**0.944**	**0.939**	**0.942**
+	+	+	0.811	0.937	0.932	0.935
+	-	-	0.790	0.937	0.930	0.933
+	-	+	0.779	0.929	0.921	0.925
-	+	-	0.568	0.782	0.782	0.782

other existing ones in the area. In this section we describe how we have evaluated our system using the LHD Gold Standard dataset, described in Section 2. As explained in [10], the gold standard is divided into three datasets (namely GS1, GS2, and GS3), containing a total of 2092 resources and their curated types. The only condition we require for them to be used in our system is that they have an abstract associated. From the total of 2.092 resources, 1.825 meet this requirement. This is due to either changes on the URIs of the resources or due to some of them being removed in the DBpedia version used in this paper.

For this evaluation, we have trained our system with the DBpedia dataset, removing the resources from the gold standard during the training phase, and then obtained the predictions for them. The results obtained using different amounts of training data, as in the previous case, are shown in Table 3. As we can see, the overall results are lower than those obtained in the 5-fold evaluation. This was expected, as the types included in the gold standard are manually curated and do not necessarily follow the typing schema from DBpedia. That is, our system learns from DBpedia data and produces types mimicking it, including the potential errors on the type assignment, whereas the manually annotated ones might diverge, thus lowering the accuracy of the predictions.

By using the gold standard dataset, we can compare our system to those existing in the state of the art. Table 3 includes the results reported in [10], in which the hSVM and SDType systems are compared[12]. As we can see, in general, our system outperforms both hSVM and SDType over the gold standard resources when trained with enough resources.

[12]We have included only the highest results reported, shown in Table 6 of the cited paper by Kliegr et. al.

Table 3
Gold Standard Evaluation

Resources	Accuracy	hPrec	hRecall	hF-measure
10K	0.205	0.669	0.624	0.645
1M	0.420	0.811	0.807	0.809
Full	0.449	0.827	**0.822**	**0.825**
hSVM	**0.548**	**0.890**	0.665	0.761
SDType	0.338	0.809	0.641	0.715

7. Conclusions and Future Work

In this paper we have explored how NLP analysis can be used for inferring types over knowledge bases, applying it to the (English) DBpedia, one of the main semantic datasets available. We have shown that by embedding textual data in a vector space model using a TF-IDF vectorization process, it is possible to guess types for resources with high precision and recall.

Our results show that the pipeline outlined in this paper is able to achieve up to a 94% of precision and recall, and around 82% when using a gold standard for evaluation, being around 6 points better than relevant tools in the state of the art. As discussed before, the test sets used under our k-fold approach were extracted from DBpedia, whereas the test set used for the gold standard evaluation is manually generated and curated one. That is, the system is better at predicting types over the DBpedia extracted test sets as they are more similar to the ones used for training, containing the same kind of type errors, which are common in DBpedia [17]. The gold standard set avoid some of those errors, which make is slightly different from the DBpedia one. Thus the performance difference shown in the results when comparing both types of evaluation. This results show and interesting behaviour when using collaboratively annotated data for building system such as the one presented in this work. We continue on researching this topic, exploring how to exploit the advantages of using large datasets, such as DBpedia, while avoiding the issues related to the inner quality of the data they contain.

The study carried out shows that, despite using Named Entity Recognition techniques over resource abstracts increases the performance of the prediction system, this improvement is not highly significant. As well, it shows that applying normalization (preprocessing) to the input text reduces the overall performance. This is due to the fact that processed words are not so discriminatory when classifying.

This work sets the foundation for more complex future systems, in which new steps can be added to the existing NLP pipeline. In this way, we are currently exploring how n-grams increases the classification capabilities of the system, by generating more discriminatory tokens. We are also working on producing a more comprehensive evaluation in which we tune the different parameters at the different steps of the process, such as the confidence threshold and weights for named entities, using different tools for text normalization, or comparing TF-IDF with other vectorization techniques. In the near future we also plan to study other classifiers for the prediction task, such as Maximun Entropy classifiers[18], which has shown promising result for text classification.

Acknowledgements

This work was partially funded by projects RTC-2016-4952-7 (esTextAnalytics), and TIN2016-78011-C4-4-R (Datos4.0), from the Spanish State Investigation Agency of the MINECO and FEDER Funds.

References

[1] V. Yadav and S. Bethard, A Survey on Recent Advances in Named Entity Recognition from Deep Learning models, 2019.

[2] M. Habibi, L. Weber, M. Neves, D.L. Wiegandt and U. Leser, Deep learning with word embeddings improves biomedical named entity recognition, *Bioinformatics* **33**(14) (2017), i37–i48. doi:10.1093/bioinformatics/btx228.

[3] H. Paulheim and C. Bizer, Improving the quality of linked data using statistical distributions, *International Journal on Semantic Web and Information Systems (IJSWIS)* **10**(2) (2014), 63–86.

[4] H. Paulheim and C. Bizer, Type Inference on Noisy RDF Data, in: *The Semantic Web – ISWC 2013*, 2013, pp. 510–525. ISBN 978-3-642-41335-3. doi:10.1007/978-3-642-41335-3_32.

[5] G. Tsoumakas and I. Vlahavas, Random k-Labelsets: An Ensemble Method for Multilabel Classification, in: *Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings*, J.N. Kok, J. Koronacki, R.L.d. Mantaras, S. Matwin, D. Mladenič and A. Skowron, eds, 2007, pp. 406–417. ISBN 978-3-540-74958-5. doi:10.1007/978-3-540-74958-5_38.

[6] M.-L. Zhang and Z.-H. Zhou, Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization, *IEEE Transactions on Knowledge and Data Engineering* **18**(10) (2006), 1338–1351. doi:10.1109/TKDE.2006.162.

[7] O. Luaces, J. Díez et al., Binary relevance efficacy for multilabel classification, *Progress in Artificial Intelligence* **1**(4) (2012), 303–313. doi:10.1007/s13748-012-0030-x.

[8] P. Ristoski and H. Paulheim, Feature Selection in Hierarchical Feature Spaces, in: *Discovery Science: 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, S. Džeroski, P. Panov, D. Kocev and L. Todorovski, eds, Springer International Publishing, Cham, 2014, pp. 288–300. ISBN 978-3-319-11812-3. doi:10.1007/978-3-319-11812-3_25.

[9] A. Melo, J. Völker and H. Paulheim, Type prediction in noisy RDF knowledge bases using hierarchical multilabel classification with graph and latent features, *Int. J. on Artificial Intelligence Tools* **26**(02) (2017).

[10] T. Kliegr and O. Zamazal, LHD 2.0: A text mining approach to typing entities in knowledge graphs, *Web Semantics: Science, Services and Agents on the World Wide Web* **39** (2016), 47–61.

[11] T. Kliegr, Linked hypernyms: Enriching DBpedia with Targeted Hypernym Discovery, *Web Semantics: Science, Services and Agents on the World Wide Web* **31** (2015), 59–69. doi:https://doi.org/10.1016/j.websem.2014.11.001. http://www.sciencedirect.com/science/article/pii/S1570826814001048.

[12] A. Gangemi, A.G. Nuzzolese et al., Automatic Typing of DBpedia Entities, in: *International Semantic Web Conference Proc.*, 2012, pp. 65–81.

[13] S. Faralli and S.P. Ponzetto, A hearst-like pattern-based approach to hypernym extraction and class induction, in: *Semantic Web Evaluation Challenge*, 2016, pp. 48–60.

[14] P.N. Mendes, M. Jakob, A. García-Silva and C. Bizer, DBpedia spotlight: shedding light on the web of documents, in: *Proceedings of the 7th international conference on semantic systems*, ACM, 2011, pp. 1–8.

[15] E. Loper and S. Bird, NLTK: The Natural Language Toolkit, in: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, Association for Computational Linguistics, Stroudsburg, PA, USA, 2002, pp. 63–70. doi:10.3115/1118108.1118117.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research* **12** (2011), 2825–2830.

[17] M. Rico, I. Santana-Pérez, P. Pozo-Jiménez and A. Gómez-Pérez, Inferring Types on Large Datasets Applying Ontology Class Hierarchy Classifiers: The DBpedia Case, in: *Knowledge Engineering and Knowledge Management*, C. Faron Zucker, C. Ghidini, A. Napoli and Y. Toussaint, eds, Springer Interna-

tional Publishing, Cham, 2018, pp. 322–337. ISBN 978-3-030-03667-6.

[18] A.L. Berger, V.J.D. Pietra and S.A.D. Pietra, A Maximum Entropy Approach to Natural Language Processing, *Comput. Linguist.* **22**(1) (1996), 39–71. http://dl.acm.org/citation.cfm?id=234285.234289.

[19] M. Rico, N. Mihindukulasooriya and A. Gómez-Pérez, Data-Driven RDF Property Semantic-Equivalence Detection Using NLP Techniques, in: *EKAW Proceedings, LNCS 10024*, Springer International Publishing, 2016, pp. 797–804. ISBN 978-3-319-49004-5. doi:10.1007/978-3-319-49004-5_51.

[20] F. Marini, A.L. Magrì and R. Bucci, Multilayer feed-forward artificial neural networks for class modeling **88** (2007), 118–124.

[21] N. Mihindukulasooriya, M. Rico et al., Repairing Hidden Links in Linked Data: Enhancing the quality of RDF knowledge graphs, in: *K-CAP proceedings*, 2017. doi:10.1145/3148011.3148020.

[22] K. Gunaratna, K. Thirunarayan, P. Jain, A. Sheth and S. Wijeratne, A Statistical and Schema Independent Approach to Identify Equivalent Properties on Linked Data, in: *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, ACM, New York, NY, USA, 2013, pp. 33–40. ISBN 978-1-4503-1972-0. doi:10.1145/2506182.2506187.

[23] B. Hachey, W. Radford, J. Nothman, M. Honnibal and J.R. Curran, Evaluating entity linking with Wikipedia, *Artificial intelligence* **194** (2013), 130–150.

[24] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann and S. Auer, Quality assessment for linked data: A survey, *Semantic Web* **7**(1) (2016), 63–93.

[25] A. Dimou, D. Kontokostas, M. Freudenberg, R. Verborgh, J. Lehmann, E. Mannens, S. Hellmann and R. Van de Walle, Assessing and refining mappingsto rdf to improve dataset quality, in: *International Semantic Web Conference*, Springer, 2015, pp. 133–149.

[26] D. Kontokostas, M. Brümmer, S. Hellmann, J. Lehmann and L. Ioannidis, NLP data cleansing based on linguistic ontology constraints, in: *European Semantic Web Conference*, Springer, 2014, pp. 224–239.

[27] H. Paulheim, Data-driven Joint Debugging of the DBpedia Mappings and Ontology, in: *European Semantic Web Conference*, 2017, pp. 1–15.

[28] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I.H. Witten, The WEKA data mining software: an update, *ACM SIGKDD explorations newsletter* **11**(1) (2009), 10–18.

[29] D. Fleischhacker, H. Paulheim, V. Bryl, J. Völker and C. Bizer, Detecting errors in numerical linked data using cross-checked outlier detection, in: *International Semantic Web Conference*, Springer, 2014, pp. 357–372.

[30] H. Paulheim, Identifying Wrong Links between Datasets by Multi-dimensional Outlier Detection., in: *WoDOOM*, 2014, pp. 27–38.

[31] J. Debattista, S. Londoño, C. Lange and S. Auer, Quality assessment of linked datasets using probabilistic approximation, in: *European Semantic Web Conference*, Springer, 2015, pp. 221–236.

[32] J. Debattista, C. Lange and S. Auer, A Preliminary Investigation Towards Improving Linked Data Quality Using Distance-Based Outlier Detection, in: *Joint International Semantic Technology Conference*, Springer, 2016, pp. 116–124.

[33] D. Gerber, D. Esteves, J. Lehmann, L. Bühmann, R. Usbeck, A.-C.N. Ngomo and R. Speck, DeFacto—Temporal and multilingual Deep Fact Validation, *Web Semantics: Science, Services and Agents on the World Wide Web* **35** (2015), 85–101.

[34] P.N. Mendes, H. Mühleisen and C. Bizer, Sieve: linked data quality assessment and fusion, in: *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, ACM, 2012, pp. 116–123.

[35] M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, S. Auer and J. Lehmann, Crowdsourcing linked data quality assessment, in: *International Semantic Web Conference*, Springer, 2013, pp. 260–276.

[36] M. Ketterl, L. Knipping, N. Ludwig, R. Mertens, J. Waitelonis, N. Ludwig, M. Knuth and H. Sack, Whoknows? evaluating linked data heuristics with a quiz that cleans up dbpedia, *Interactive Technology and Smart Education* **8**(4) (2011), 236–248.

[37] N. Mihindukulasooriya, M. Poveda-Villalón, R. García-Castro and A. Gómez-Pérez, Collaborative Ontology Evolution and Data Quality-An Empirical Analysis, in: *International Experiences and Directions Workshop on OWL*, Springer, 2016, pp. 95–114.

[38] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, DBpedia: A Nucleus for a Web of Open Data, *The Semantic Web* (2007), 722–735.

[39] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, A Review of Relational Machine Learning for Knowledge Graphs, in: *Proceedings of the IEEE*, Vol. 104, IEEE, 2015, pp. 11–33.

[40] M.-L. Zhang and Z.-H. Zhou, A review on Multi-Label Learning Algorithms, *IEEE Transactions on Knowledge and Data Engineering* (2014), 1819–1837.

[41] P. Probst, Q. Au, G. Casalicchio, C. Stachl and B. Bischl, Multilabel Classification with R Package mlr, *arXiv preprint arXiv:1703.08991* (2017).

[42] L. Breiman, Random Forests, *Machine Learning* **45**(1) (2001), 5–32.

[43] K.P. Murphy, Naive bayes classifiers, *University of British Columbia* (2006).

[44] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

[45] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda et al., Top 10 algorithms in data mining, *Knowledge and information systems* **14**(1) (2008), 1–37.

[46] S. Faralli and S.P. Ponzetto, DWS at the 2016 open knowledge extraction challenge: a hearst-like pattern-based approach to hypernym extraction and class induction, in: *Semantic Web Evaluation Challenge*, Springer, 2016, pp. 48–60.

[47] N. Mihindukulasooriya, M. Rico et al., An Analysis of the Quality Issues of the Properties Available in the Spanish DBpedia, in: *16th Conference AEPIA*, 2015, pp. 198–209.