Domain-aware Matching of Events to DBpedia

Kristian Slabbekoorn, Laura Hollink, and Geert-Jan Houben

Web Information Systems Group
Delft University of Technology, The Netherlands
k.slabbekoorn@student.tudelft.nl,{l.hollink, g.j.p.m.houben}@tudelft.nl

Abstract. In this paper, we present our work on the enrichment of the EventMedia dataset as provided by the DeRiVE data challenge with links to DBpedia. Our main contribution is an exploration into the use of domain knowledge in the matching process. As a starting point we take DBpedia Spotlight, an off-the-shelf tool for matching textual resources to DBpedia. We present a bootstrap method to automatically derive the needed domain knowledge from an initial set of high confidence matches, and compare this to a baseline method without any domain knowledge, and an 'oracle' method with perfect domain knowledge.

1 Introduction

In this paper, we present our work on the enrichment of the EventMedia dataset as provided by the DeRiVE data challenge with links to DBpedia. Tools and algorithms have emerged that automate the task of matching two ontologies or datasets. Most of these systems use string similarity measures and/or structural measures to determine the similarity between a pair of resources. However, little is known about how one can include the domain of the data into the matching process. In our case, we know that the EventMedia dataset is about events, performing artists and venues. The main contribution of this paper is an exploration into the use of this knowledge of the domain to produce better or more matches. In addition, the resulting matches are made publicly available for download.

As a starting point we take DBpedia Spotlight, an off-the-shelf tool for matching textual resources to DBpedia. DBpedia Spotlight has been shown to be able to compete with established annotation systems while remaining largely configurable [1]. The configurability allows us to include various forms of domain knowledge, and test the effect on the resulting matches. It also means, however, that we have to choose values for a relatively large number of parameters that potentially influence the results. To minimize this effect, we set the parameters systematically and transparently in section 2.1.

We present a bootstrap method to automatically derive the needed domain knowledge from an initial set of high confidence matches, and compare this to a baseline method without any domain knowledge, and an 'oracle' method with perfect domain knowledge. To explore the generalizability of the derived domain knowledge, we perform an evaluation in which we derive the domain knowledge from one dataset and use it to find matches in another dataset.

Several bootstrapping methods to derive links between Linking Open Data (LOD) datasets have been proposed previously. [2] matches concepts by finding candidates in DBpedia, then comparing classifications of their own concepts to the classes and categories of the DBpedia candidate concepts. In our case, we do not assume to have a classification of the source data available. BLOOMS+ [3] uses the Wikipedia category hierarchy to bootstrap the process of finding schema-level links between LOD datasets. We exploit the Wikipedia category hierarchy in a similar fashion; not to find matches directly but to find categories (and classes) that effectively describe our domain of interest.

1.1 Dataset and Reference Alignment

All experiments are performed on the EventMedia dataset provided as part of the DeRiVE data challenge, containing RDF statements about events, artists and venues from the websites Last.fm, Eventful.com and Upcoming.yahoo.com. We have chosen to focus on matching artists as they are more likely to have pages dedicated to them on Wikipedia than venues and events do - pages can be found for roughly 35% of the artists contained in the Last.fm dataset, and 45% of the artists contained in the Eventful dataset. Upcoming does not contain explicit mentions of artists. We evaluate our approach by comparison to a manually composed reference alignment of 1500 randomly picked artists (1000 from Last.fm and 500 from Eventful.com) to DBpedia resources.

1.2 DBpedia Spotlight

Throughout this work we have used DBpedia Spotlight [1], a powerful tool for automatically annotating natural language texts with links to DBpedia resources. It does so by first finding surface forms in the text that could be mentions of DBpedia resources (the ‘spotting’ function), then disambiguating to link to the right DBpedia resources based on context similarity measures (the ‘disambiguation’ function). Its results can be directed towards high precision or high recall by setting two parameters: a ‘support’ threshold for minimum popularity of the Wikipedia page (i.e. the number of inlinks from other Wikipedia pages) and a ‘confidence’ threshold for minimum similarity between source text and context associated with DBpedia surface forms. The latter has been normalized to a range of 0..1. In addition, Spotlight’s ‘black- and whitelists’ allow one to filter the results to exclude/include only members of certain classes and categories that correspond to the domain of the source text.

2 Approach

In this section we present our approach to domain-aware matching. We compare our results to a baseline approach, where we run Spotlight without any domain filters, and an ‘oracle’ approach, where the optimal classes and categories are chosen as a domain filter, based on the best matching results in hindsight.

We do our matching in two passes. In the first pass we attempt to match the full label. We do this by marking the full `rdfs:label` of an artist for disambiguation and appending the `dc:description` value, if available, as context for Spotlight's 'disambiguate' function. We attempt to increase the number of links by running a second pass with Spotlight's 'spotting' function on artists not matched initially to search for surface forms 'hidden' inside the label (for instance, some labels include more than one artist).

Bootstrap approach: deriving domain filters. We bootstrap the selection of domain filters by first running DBpedia Spotlight without any knowledge of the domain, with parameters set towards high precision, to obtain an initial set of links from our data to DBpedia resources. From these resources we gather the associated DBpedia classes, YAGO classes and Wikipedia categories and use these as a domain knowledge filter to find further matches.

To get a set of classes and categories that concisely describes our domain, we first gather all DBpedia and YAGO classes of the matched DBpedia resources, including all super-classes up to the root of their respective hierarchies. For categories, we gather only up to 4 ancestors of each, as due to the size and messy structure of the Wikipedia category hierarchy the set will quickly become too large and broad to be useful.

Second, we select the appropriate classes and categories from this long list as follows. We count the number of occurrences of each class or category. We filter out all classes that occur less than $r\%$ of the total number of classes found. General (super-)classes will occur more frequently then specific (sub-)classes. Therefore, the higher the value of r, the more general our list of classes will be. For categories, this effect is less strong since we do not gather super-categories up to the root. Therefore we simply select the top t categories that occur the most. To avoid too much redundancy in the list of classes and categories, i.e. to avoid including a super-class plus all its sub-classes, we filter out all super-classes where the sum of the numbers of occurrence of their sub-classes is more than 90% of the number of occurrence of the super-class. The same procedure is followed for categories. The resulting list of DBpedia classes, YAGO classes and categories represents our domain filter.

2.1 Spotlight parameter optimization

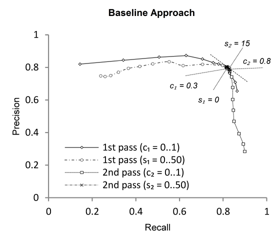
In this paper we assume an application that values precision and recall equally, and therefore we optimize the parameters for high F-measure (F). To allow a fair comparison between the three approaches, we set the parameters of each approach independently to the values that are optimal for that approach.

For each approach, we need to optimize 'confidence' c and 'support' s for pass 1 and pass 2, resulting in the four parameters c_1, c_2, s_1 and s_2. We determine the best setting of all parameters by evaluating the resulting matches against our reference alignment. First, we keep s_1 fixed at 0 and vary c_1 between 0 and 1 in steps of 0.1. Next, we take values around and including the value of c_1 that provided the highest F and vary s_1 between 0 and 50 in steps of 5. We take this relatively low range of inlinks due to the nature of our dataset, which largely consists of obscure entities that are likely not often linked to. We settle on whichever combination of c_1 and s_1 gives the best F. To set the parameters c_2 and s_2 of the second pass, analogous experiments are performed, this time varying c_2 and s_2 respectively.

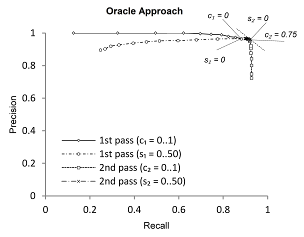
Baseline approach Figure 1a shows that the highest F-measure (F_{max}) is obtained when parameters are set as follows: $c_1 = 0.3$, $s_1 = 0$, $c_2 = 0.8$ and $s_2 = 15$.

Oracle approach Optimal parameters for this approach are $c_1 = 0.0$, $s_1 = 0$, $c_2 = 0.75$ and $s_2 = 0$. See figure 1b.

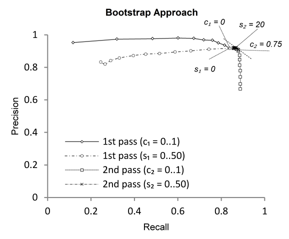
Bootstrap approach Our aim is to evaluate our bootstrap approach with varying amounts of classes and categories to specify the domain. The optimal parameters for each variant could be different and hence they need to be determined independently. For space reasons, figure 1c only shows the parameters for the approach that gave the best results: $c_1 = 0.0$, $s_1 = 0$, $c_2 = 0.75$ and $s_2 = 20$.



(a) The baseline approach. $F_{max} = 0.811$



(b) The oracle approach. $F_{max} = 0.939$



(c) The bootstrap approach ($r = 1, t = 0$). $F_{max} = 0.895$

Fig. 1: The effect of different parameter settings on precision and recall. The optimal values for each parameter are denoted in the graphs. The highest F-measure scores F_{max} correspond to the values at the top-right corners of the graphs.

3 Experimental results

Table 1 show the results of our experiments on the Last.fm artist dataset. We see that our domain-aware bootstrap method gives results that are better than the baseline, and close to the 'oracle'. The results are slightly better when we do not consider categories at all ($t = 0$). A reason for this is because it is difficult to detect and filter out overly general categories (for example, `Category:People` is always part of our result). There is also an expected inherent trade-off to be seen between precision and recall when we choose r as either 1 or 2. We additionally run the baseline, oracle and best-performing bootstrap approach on the Eventful artist dataset and evaluate based on a ground truth of 500 artist labels derived in a similar way to the Last.fm ground truth. These results again show the value

Table 1: Results for each approach sorted by maximum F-measure.

Approach	*Precision*	*Recall*	F_{max}
Oracle	0.966	0.914	0.939
Bootstrap: $r = 1, t = 0$	0.921	0.870	0.895
Bootstrap: $r = 2, t = 0$	0.869	0.916	0.892
Bootstrap: $r = 1, t = 10$	0.877	0.905	0.891
Bootstrap: $r = 2, t = 10$	0.846	0.902	0.873
Bootstrap: $r = 1, t = 5$	0.835	0.902	0.867
Bootstrap: $r = 2, t = 5$	0.828	0.902	0.863
Baseline	0.799	0.824	0.811

of domain knowledge, with $F_{max} = 0.726$ for the baseline, $F_{max} = 0.905$ for the oracle, and $F_{max} = 0.901$ for the derived approach.

The DBpedia links created with the oracle approach for both datasets are available for download[1]. Also included are interlinks between Last.fm and Eventful artists if they have all of their links in common. For Last.fm, we have 50120 entities in total and end up with 17116 DBpedia links. For Eventful, we have 6540 entities and 2724 links. There are 2450 interlinks made between datasets.

4 Discussion and Future Work

In this paper we presented a bootstrapping method to improve the matching of concepts within a particular domain to DBpedia resources. We found that our bootstrapping method performs better than a general domain-independent matching, and that the F-measure associated to our best derived model is consistent across two datasets. It is not yet clear to what extent our proposed method is applicable to other domains of a different nature. We are currently exploring how robust our method is against different sets of initial high confidence matches, varying the size as well as the domain.

We found that we often end up with rather generic classes/categories, such as YAGO class `LivingPeople` and category `People`, in our final selections for a domain filter. Our future work focusses on improving the class and category selection algorithm in order to filter out these general cases.

References

1. Mendes, P., Jakob, M., Garca-Silva, A., Bizer, C.: DBpedia Spotlight: Shedding Light on the Web of Documents. In the Proc. of the 7th International Conference on Semantic Systems (I-Semantics). Graz, Austria, 7-9 September 2011. (to appear)
2. Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., Lee, R.: Media Meets Semantic Web - How the BBC Uses DBpedia and Linked Data to Make Conections. Proc. of ESWC 2009, Heraklion, Crete.
3. Jain P., Yeh P. Z., Verma K., Vasquez R. G., Damova M., Hitzler P., Sheth A. P.: Contextual Ontology Alignment of LOD with an Upper Ontology: A Case Study with Proton. In G. Antoniou (Ed.), Proc. of ESWC 2011, Heraklion, Crete.

[1] http://wis.ewi.tudelft.nl/iswc2011/derive/