# Entity Suggestion Ranking via Context Hashing

Rima Türker[1,2], Maria Koutraki[1,2], Jörg Waitelonis[3], and Harald Sack[1,2]

[1] FIZ Karlsruhe, Karlsruhe, Germany
firstname.lastname@fiz-karlsruhe.de
[2] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
firstname.lastname@kit.de
[3] Hasso Plattner Institute for IT Systems Engineering, Potsdam, Germany
joerg.waitelonis@hpi.de

## 1 Introduction

In text-based semantic analysis the task of named entity linking (NEL) establishes the fundamental link between unstructured data elements and knowledge base entities. The increasing number of applications complementing web data via knowledge base entities has led to a rich toolset of NEL frameworks [4,7]. To resolve linguistic ambiguities, NEL relates available context information via statistical analysis, as e.g. term co-occurrences in large text corpora, or graph analysis, as e.g. connected component analysis on the contextually induced knowledge subgraph. The semantic document annotation achieved via NEL algorithms can furthermore be complemented, upgraded or even substituted via manual annotation, as e.g. in [5]. For this manual annotation task, a popular approach suggests a set of potential entity candidates that fit to the text fragment selected by the user, who decides about the correct entity for the annotation. The high degree of natural language ambiguity causes the creation of a huge sets of entity candidates to be scanned and evaluated. To speed up this process and to enhance its usability, we propose a pre-ordering of the entity candidates set for a predefined context. The complex process of NEL context analysis often is too time consuming to be applied in an online environment. Thus, we propose to speed up the context computation via approximation based on the offline generation of context weight vectors. For each entity, a context vector is computed beforehand and is applied like a hash for quickly computing the most likely entity candidates with respect to a given context. In this paper, the process of entity hashing via context weight vectors is introduced. Context evaluation via weight vectors is evaluated on the test case of *SciHi*[1], a web blog on the history of science providing blog posts semantically annotated with DBpedia entities.

## 2 Creating Context Vectors for Entity Hashing

**Conceptual Idea.** Most of the DBpedia entities are linked to related Wikipedia categories[2].Categories represent precious characteristics about the resources and

[1] http://scihi.org/
[2] https://en.wikipedia.org/wiki/Help:Category

they are hierarchically related to form a taxonomy via the `skos:broader` property. However, to identify the relation between a resource and a more general category such as e.g., between Wolfgang A. Mozart and `dbc:Music`, a path has to be determined from `dbc:Composers_for_piano`, the category directly connected to Mozart, via a sequence of `skos:broader` connections to `dbc:Music`. In contrast to other studies [6,2,1], in this paper, DBpedia resources are characterized based on their Wikipedia category associations. Note that related NEL approaches make use of all the Wikipedia categories. However, in this study a small number of categories is used to reduce the complexity and to make the approach scalable.

**Weight Vector Heuristics.** For a manual semantic annotation use case, usually many potential entity candidates are displayed to the user. In order to display the most likely entity candidates first, we propose a sorting based on the computation of a weight vector for each DBpedia entity. Weight vectors reflect the characteristics of the entities with respect to predefined Wikipedia categories. Each DBpedia entity will be represented as an $m$-dimensional vector of weights $(w_1, \ldots, w_m)$ that indicate the connectedness of the entity to the predefined categories $c_1, \ldots, c_m$. Since our use case *SciHi* provides 35 science related categories, entities will be represented by a 35 dimensional weight vector ranging from $c_1 =$ Archeology to $c_{35} =$ Zoology. The single weights $w_i$ of a weight vector $wv_e = (w_1, \ldots, w_m)$ are computed based on the number of paths from the entity $e$ to the categories $(c_1, \ldots, c_m)$ in the RDF graph via `dct:subject` and `skos:broader` with shorter paths indicating a stronger relation. The following formula for weight calculation has been determined empirically:

$$w_c(e) = \begin{cases} 0, & \text{if there is no path from } e \text{ to } c, \\ \sum_{l=l_{min}}^{maxDepth} \frac{\#p_l(e,c)}{2^{l-l_{min}}} \cdot log\left(\frac{N}{n_l}\right), & \text{otherwise} \end{cases}$$

where $w_c(e)$ is the single weight value for category $c$ and entity $e$, *maxDepth* is the maximum path length, which was empirically determined as $maxDepth = 6$, since longer path lengths cause the introduction of too much noise. $l_{min}$ designates the shortest discovered path length from entity $e$ to any category, $\#p_l(e,c)$ is the number of paths from entity $e$ to category $c$ with length $l$ , $N$ is the total number of entities, and $n_l$ is the number of entities with paths to category $c$ of length $l$ .

The longer the path length, the less relevant is the connection of entity $e$ to category $c$ and the more noise is introduced. Therefore the number of paths $\#p_l(e,c)$ is divided by $2^{l-l_{min}}$, i.e. for each hop starting with the shortest found path at length $l_{min}$ the weight is cut in halve, and further multiplied by term frequency-inverse document frequency(tf-idf), since more general categories will always have a high number of paths and should not be overemphasized [7]. Finally, the weight vectors are normalized to enable a fair comparison.

Table 1: Evaluation results

| | Baseline | Approach |
|---|---|---|
| Total items | 28,516 | 28,516 |
| Top 1 | 2,336 (8.2 %) | 2,699 (9.5 %) |
| Top 5 | 4,609 (16.2 %) | 6,481 (22.7 %) |
| Top 10 | 5,381 (18.9 %) | 8,116 (28.5 %) |

## 3 Experiments And Results

**Evaluation Requirements.** The general purpose of the weight vectors is to improve semantic text annotation systems. If the general domain of the text is known, this information might assist the subsequent entity linking processes. With knowledge about the domain the disambiguation process can be refined by putting more focus on entities belonging to the domain. Another application is the entity lookup problem as for example solved in auto-completion systems [3]. By entering a search string, a list of knowledge base entities is presented, usually in a string distance based order, from which the user selects the best suited candidate. With focus on a given domain, the ranking of candidate entities in the auto-completion list could be re-organized so that entities belonging to the domain are preferred over others. The hypothesis is, that re-ranking provides better suggestions so that the work with the system is more efficient when incorporating domain information. With this application scenario in mind, we have conducted an evaluation experiment, incorporating the auto-suggestion system for entity lookup proposed in [3]. For the evaluation the ground truth dataset requires to include potential search strings (e.g. surface forms of entity mentions), the associated entities, as well the categories. Datasets for entity linking should fit perfectly for this purpose. Unfortunately, those datasets often don't provide information about a specific domain or category. Only the WES2015[8] dataset created from the *SciHi* blog articles provides categories for each annotated documents, which contains 331 science related articles with a total number of 28,516 annotations with 8,582 distinct entities and 11,990 different surface forms.

**Experiment Design.** For the evaluation, the auto-suggestion system was adapted to also take into account the weight vectors for the entities. For each annotation of the dataset, the list of entities was generated with the surface form as input and the annotation entity was localized in the list. If the entity was within the top $n$ positions, a true positive was ascertained. The following versions of the ranking were evaluated: (a) order by string distance & in-degree (cf. [3]) (baseline) and (b) order by weight vector category.

The baseline is the combined ranking implemented in [3]. First, the results are ordered by JaroWinkler distance [9] between the surface form and the entity's main label. Subsequently the top $n = 50$ items were ordered according to entities' in-degree derived from the Wikipedia link graph to approximate entity popularity. Thus, popular entities should be ranked higher than unpopular enti-

ties. The order by weight vector category was realized by sorting the result list by the values at the corresponding category index in the weight vector.

**Evaluation Results.** Tab. 1 presents the evaluation results. The first row ("Total items") shows the total number of entities found, the next rows show the number of entities found within the $n$ topmost positions, e.g. with the baseline 2,336 out of 28,516 entities were suggested at the first position, thus 8.2 % were a perfect hit.

One can see that the number of hits was increased by applying the weight vector based ordering. Whereas the improvement on the top 1 hits from 8.2 % to 9.5 % was only slight (1.3 %), the top 5 (6.5 %) and top 10 (9.6 %) improvements are more clear.

## 4 Conclusion And Future Work

In this paper, we have shown how to approximate contextual information using weight vectors. The approach has been evaluated in an entity suggestion use case and has shown a clear improvement compared with the baseline that only takes into account string similarity and popularity measures. To be applicable in a more general use case, weight vector dimensions have to be adapted (a) to the domain in focus or (b) to the general case without domain restrictions, as e.g. categories from a top level ontology. Current research is focused on adapting the presented approach with weight vector hashing also to named entity linking.

## References


1. Hoffart, J., Yosef, M.A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Weikum, G.: Robust disambiguation of named entities in text. In: EMNLP (2011)
2. Ichinose, S., Kobayashi, I., Iwazume, M., Tanaka, K.: Ranking the results of dbpedia retrieval with sparql query. In: JIST (2013)
3. Osterhoff, J., Waitelonis, J., Sack, H.: Widen the Peepholes! Entity-Based Auto-Suggestion as a rich and yet immediate Starting Point for Exploratory Search. In: IVDW (2012)
4. Shen, W., Wang, J., Han, J.: Entity linking with a knowledge base: Issues, techniques, and solutions. Transactions on Knowledge & Data Engineering (2015)
5. Tietz, T., Jäger, J., Waitelonis, J., Sack, H.: Semantic annotation and information visualization for blogposts with refer. In: VOILA (2016)
6. Tonon, A., Catasta, M., Prokofyev, R., Demartini, G., Aberer, K., Cudré-Mauroux, P.: Contextualized ranking of entity types based on knowledge graphs. Web Semant. (2016)
7. Usbeck, R., et al.: Gerbil: General entity annotator benchmarking framework. In: WWW (2015)
8. Waitelonis, J., Exeler, C., Sack, H.: Linked Data Enabled Generalized Vector Space Model to Improve Document Retrieval. In: Proceedings of NLP & DBpedia 2015 workshop in ISWC (2015)
9. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: SRMS (1990)