DoSeR - A Knowledge-Base-Agnostic Framework for Entity Disambiguation Using Semantic Embeddings

Stefan Zwicklbauer, Christin Seifert, Michael Granitzer

University of Passau, Passau 94032, Germany,
forename.surname@uni-passau.de

Abstract. Entity disambiguation is the task of mapping ambiguous terms in natural-language text to its entities in a knowledge base. It finds its application in the extraction of structured data in RDF (Resource Description Framework) from textual documents, but equally so in facilitating artificial intelligence applications, such as Semantic Search, Reasoning and Question & Answering. In this work, we propose DoSeR (Disambiguation of Semantic Resources), a (named) entity disambiguation framework that is knowledge-base-agnostic in terms of RDF (e.g. DBpedia) and entity-annotated document knowledge bases (e.g. Wikipedia). Initially, our framework automatically generates semantic entity embeddings given one or multiple knowledge bases. In the following, DoSeR accepts documents with a given set of surface forms as input and collectively links them to an entity in a knowledge base with a graph-based approach. We evaluate DoSeR on seven different data sets against publicly available, state-of-the-art (named) entity disambiguation frameworks. Our approach outperforms the state-of-the-art approaches that make use of RDF knowledge bases and/or entity-annotated document knowledge bases by up to 10% F1 measure.

Keywords: Entity Disambiguation, Neural Networks, Linked Data, Semantic Web

1 Introduction

Entity disambiguation refers to the task of linking phrases in a text, also called surface forms, to a set of candidate meanings, referred to as the knowledge base (KB), by resolving the correct semantic meaning of the surface form. It is an essential task in combining unstructured with structured or formal information; a prerequisite for artificial intelligence applications such as Semantic Search, Reasoning and Question & Answering. Regarding Example 1, an accurate (named) entity recognition system (focusing on persons, organizations and locations only) would return the surface forms **TS** and **New York**. A (named) entity disambiguation system, basing on the well-known DBpedia KB [12], maps the surface forms **TS** and **New York** to the DBpedia resources *Times_Square* and *New_York_City*.

Example 1. The **TS** has been a **New York** attraction for over a century.

While entity disambiguation systems have been well-researched so far, most approaches, such as DBpedia Spotlight [14], TagMe2 [4] or Wikifier [2], have been designed to work on a particular type of KB [22], namely either RDF-based KBs (RDF-KB) like DBpedia [12] or YAGO3 [13], or entity-annotated document KBs (EAD-KB)

like Wikipedia. As a consequence these approaches might perform significantly worse after exchanging the underlying KB or do not work at all. Furthermore, RDF-KBs and EAD-KBs can complement each other in terms of entity coverage, i.e. the total number of entities available in a KB, and entity description, i.e. the completeness and quality of the description of one entity. In order to exploit the potential of different KBs, entity disambiguation approaches have to be knowledge-base-agnostic in terms of RDF-KBs and EAD-KBs, while at the same time maintaining simplicity, disambiguation accuracy and preprocessing/disambiguation performance.

In this work, we present DoSeR - (**D**isambiguation **o**f **Se**mantic **R**esources), a novel, knowledge-base-agnostic entity disambiguation framework. DoSeR achieves higher F-measures than the current, publicly available, state-of-the-art (named) entity disambiguation frameworks for DBpedia and Wikipedia entities. These results are achieved by first computing semantic entity embeddings using information from one or multiple underlying KBs and then applying the PageRank algorithm to an automatically created entity candidate graph.

Our **contributions** are the following:

- We present DoSeR, a (named) entity disambiguation framework that is knowledge-base-agnostic in terms of RDF and entity-annotated document KBs.
- We propose how to easily generate semantic entity embeddings to compute semantic similarities between entities regardless of the type of KBs available.
- We evaluate DoSeR on seven well-known and open source data sets showing that DoSeR outperforms current publicly available, state-of-the-art approaches by up to 10% F1 measure.

The remainder of the paper is structured as follows: In Section 2, we review related work. Section 3 introduces the problem formally and outlines our approach. Sections 4 and 5 explain the implementation of DoSeR as well as the process of generating semantic entity embeddings. In Section 6, we present the results of DoSeR attained on seven data sets. Finally, we conclude our paper in Section 7.

2 Related Work

Entity disambiguation has been studied extensively in the past 10 years. One of the first frameworks to annotate and disambiguate Linked Data Resources was **DBpedia Spotlight** [14]. The framework uses DBpedia as underlying KB, is based on the vector space model and cosine similarity and is publicly available as web service. Further, it is able to disambiguate all classes of the DBpedia ontology. Another framework to disambiguate Linked Data resources is **AGDISTIS** [22], a knowledge-base-agnostic approach from 2014. It is based on string similarity measures, an expansion heuristic for labels to cope with co-referencing and the graph-based Hypertext-Induced Topic Search (HITS) algorithm. Focusing on named entities only, it is the current publicly available, state-of-the-art approach in terms of disambiguating named entities while exclusively using DBpedia knowledge. Hoffert et al. proposed **AIDA** [7], a named entity disambiguation framework which is based on YAGO2. It unifies prior approaches into

a framework that combines the following three measures: prior probability of an entity, the similarity between the surface forms' context and entity candidates as well as the coherence among entity candidates for all surface forms together.

In contrast to the approaches mentioned before, *wikification* approaches link surface forms to Wikipedia pages. One of the first wikification approaches was proposed by Cucerzan et al. [3] who introduced topical coherence for entity disambiguation. The authors used the referent entity candidate and other entities within the same context to compute topical coherence by analyzing the overlap of categories and incoming links in Wikipedia. Milne and Witten [16] improved the exploitation of topical coherence using Normalized Google Distance and unambiguous entities in the context only. Further improvements on topical coherence were made by Kataria et al. [9] who proposed a new topic model (Wiki-based Pachinko Allocation Model). Their model uses all words of Wikipedia to learn entity-word associations and the Wikipedia category hierarchy to capture co-occurrence patterns among entities. Subsequent works are also based on topic models which perform well on data sets with sufficiently long textual contexts [5].

Another framework is Linden [21], an approach to link named entities in text with a knowledge base unifying Wikipedia and WordNet, by leveraging the rich semantic knowledge embedded in the Wikipedia and the taxonomy of the knowledge base. **Wikifier** [2], a well-known system from 2013, incorporates, along with statistical methods, richer relational analysis of the text. To our knowledge it is the current publicly available state-of-the-art system for linking surface forms to Wikipedia pages regarding the average accuracy across several data sets. Another publicly available framework that links to Wikipedia pages is **WAT** [17], which includes a redesign of TagMe [4] components and introduces two disambiguation families: graph-based algorithms for collective entity disambiguation and vote-based algorithms for local entity disambiguation [23].

The authors of [26] provide an evaluation of biomedical disambiguation systems with respect to three crucial properties: entity context, i.e. the way entities are described, user data, i.e. quantity and quality of externally disambiguated entities, and quantity and heterogeneity of entities to disambiguate.

Many state-of-the-art approaches rely on exhaustive data mining methods and compute, similar to us, semantic relatedness models for entity disambiguation [8]. However, in contrast to these models, DoSeR maintains simplicity, disambiguation accuracy on all data sets, preprocessing/disambiguation performance and extensibility (in terms of KBs). The bold highlighted frameworks are compared with DoSeR in Section 6.

3 Problem Statement and Approach

The goal of entity disambiguation is to find the correct semantic mapping between surface forms and entities in a KB. More formally, let $< m_1, ..., m_K >$ be a tuple of K surface forms in a document D, and $\Omega = \{e_1, ..., e_{|\Omega|}\}$ be a set of target entities in a KB. Let Γ be a possible entity configuration $< t_1, ..., t_K >$ with $t_i \in \Omega$, where t_i is the respective target entity for surface form m_i. In this definition, we assume that each entity in Ω is a possible candidate for a surface form m_i. The goal of entity linking can then be formalized as finding the optimal entity configuration Γ^*. Different to [18] we do not pose the optimization problem as maximizing the sum of the scores of a locality

function ϕ and a coherence function ψ, which has been proven to be NP-hard [3]. Instead, we approximate the solution using the PageRank algorithm with priors [1, 24] on a specially constructed graph G which encompasses both, the locality and the coherence function. In our work, the locality function reflects the likelihood that a target entity t_i is the correct disambiguation for m_i, whereas the coherence function computes a compatibility score describing the coherence between entities in Γ^*. The PageRank algorithm is a well-researched link-based ranking algorithm that simulates a random walk on the underlying graph and reflects the importance of each node. It has been shown to provide good performance for many applications [25], also in entity disambiguation tasks [6].

The graph we construct consists of one node for each entity candidate for all given surface forms. The graph is K-partite, with K being the number of surface forms, and only contains edges between entities of different surface forms. The edge weights are based on the similarity of the semantic embeddings of the entities. We show that semantic embeddings (vectors) can be estimated in a KB-agnostic way and thus, the graph can be constructed for any given KB.

The core steps of our solution are (i) the generation of entity candidates for the given surface forms, (ii) the estimation of semantic embeddings from EAD-KBs and RDF-KBs, (iii) the construction of the K-partite graph using priors and similarities of semantic embeddings, and (iv) the application of PageRank with priors on the graph.

The overall process is outlined in Section 4 and its subsections, whereas the estimation of semantic embeddings for different types of KB is presented in Section 5.

4 DoSeR - Disambiguation of Semantic Resources

In this section, we present the architecture of the DoSeR framework[1] (cf. Figure 1). DoSeR consists of the following three main steps: 1) index creation (Section 4.1), candidate generation (Section 4.2) and the assignment of entities to surface forms (Section 4.3). The first step in the index creation process is to define a set of *core* KBs. The set of core KBs (depicted with a continuous line in Figure 1) is used to specify the set of entities Ω which should be disambiguated by our framework. In the following, DoSeR processes the contents of all given (core and optional) KBs and stores available surface forms from entity-annotated documents, a semantic entity embedding as well as a prior probability for each entity (optional KBs are figured with a dashed line in Figure 1). This KB preprocessing step is executed only the first time or if the data of a new KB should be integrated. After preprocessing, DoSeR accepts documents with surface forms (e.g. manually marked by users) that should be linked to entities.

In the candidate generation step, we identify a set of possible entity candidates for each surface form and, thus, significantly reduce the number of possible target entities. To this end we apply several heuristics proposed in [22] or make use of known surface forms. In the final disambiguation step we use this set of candidates to create an entity candidate graph. By applying a PageRank algorithm we attempt to find the best possible entity configuration. More specifically, each entity candidate of a surface form that provides the highest PageRank score denotes the disambiguated target entity for that surface form. In the following, we present each of the steps of DoSeR in more detail.

[1] `https://github.com/quhfus/DoSeR/`

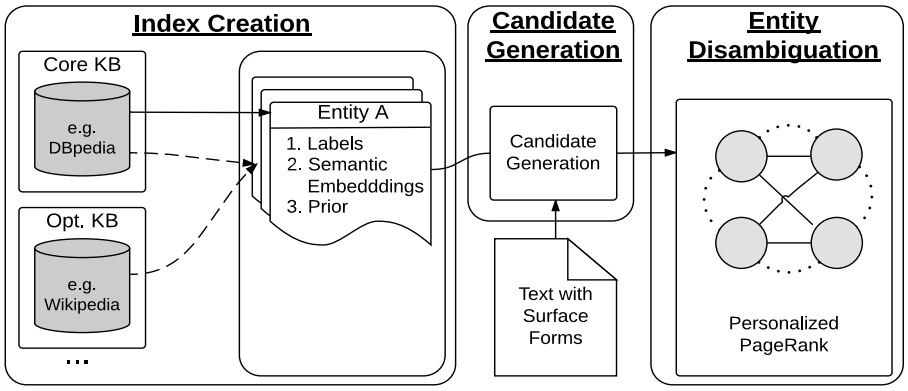

Fig. 1. Overview of DoSeR

4.1 Index Creation

Before starting the index creation process we first choose one or multiple source KBs that contain entity describing data. Basically, DoSeR accepts RDF-KBs (e.g. DBpedia, YAGO3) and EAD-KBs (e.g. Wikipedia). Since EAD-KBs do not have a standardized format, DoSeR requires a unified format for annotated entities. Example 2 shows an annotation of the surface form "New York", with the *id* denoting a unique entity id:

Example 2. ...been a <e id="dbr:New_York_City">New York</e> attraction...

To have EAD-KBs in the denoted format we have to parse them in the first step.

Next, given a set of source KBs in the appropriate format, we select a set of *core* KBs. The set of all entities specified or annotated in these core KBs specifies our target entity set Ω. If the core KBs provide information about the entities' classes (e.g. *rdf:type*), Ω can be restricted to named entities only (i.e. persons, organizations and places). After specifying Ω, we use all core and optional KBs as data sources for the entities in Ω. Optional KBs complement the core KBs in terms of completeness and quality of entity descriptions. Overall, our approach is fully knowledge-base-agnostic in terms of RDF-KBs and EAD-KBs. In the next step, DoSeR creates an index comprising the following three entity describing information:

Labels: By default DoSeR extracts the *rdfs:label* attribute of all given RDF-KBs and stores them in a label field. Our approach can be configured to use any set of properties as label. Further, DoSeR searches for EAD-KBs in our specified KB set and, if available, extracts and stores surface forms which have been used to address a specific entity.

Semantic Embeddings: DoSeR automatically creates a semantic embedding (vector) for all entities regardless of the underlying KBs. The resulting embeddings are used to compute a semantic similarity between entities. The creation of these embeddings on the basis of different KBs is explained in Section 5 in detail.

Prior: Generally, some entities occur more frequently than others. Thus, these popular entities provide a higher probability to reoccur in other documents [14, 26]. The prior $p(e_i)$ describes the a-priori probability that entity e_i occurs. We use the core KBs to compute entity priors by analyzing the number of its annotations in the documents (EAD-KB) or the number of in and outgoing edges (RDF-KB). In the latter case, we regard the KB as a directed graph, where the nodes V are resources, the edges E are properties and $x, y \in V, (x, y) \in E \Leftrightarrow \exists p : (x, p, y)$ is an RDF triple. For those entities, which have been annotated in a core EAD-KB, we use the number of its annotations in these documents to compute the prior. For the other entities, we use the number of in- and outgoing edges in the RDF-KBs as quantity during the prior computation.

Given these information in an index, we disambiguate entities by selecting relevant candidates (Section 4.2) and computing the optimal entity assignments (Section 4.3).

4.2 Candidate Generation

Given a constructed index, our framework accepts documents that contain one or multiple surface forms that should be linked to entities. DoSeR disambiguates all surface forms within a document using a collective approach. In our entity disambiguation chain, entity candidate generation is the first crucial step (cf. Figure 1). Our goal is to reduce the number of possible entity candidates for each input surface form by determining a set of relevant target entities. Hereby, we proceed as follows:

First, we compare the input surface form to those stored in the index. All entities in the index that provide an exact surface form matching serve as entity candidate.

Second, we use the candidate generation approach proposed by Usbeck et al. for AGDISTIS [22]. The authors apply a string normalization approach to the input text. It eliminates plural and genitive forms, removes common affixes such as postfixes for enterprise labels and ignores candidates with time information within their label. Similar to AGDISTIS [22], our system compares the normalized surface forms with the labels in our index by applying trigram similarity. The trigram similarity threshold ($\sigma = 0.82$) is constant in our system and experiments since it provides good results across all data sets and is the default setting in the AGDISTIS framework[2]. If an entity's label matches with the heuristically obtained label, while exceeding the trigram similarity threshold, and the entity is not yet a candidate for the surface form, the entity becomes a candidate.

4.3 Entity Disambiguation

After generating candidates for each surface form, DoSeR uses the set of candidates to create an entity candidate graph. On this graph we perform a random walk and determine the node relevance which can be seen as the average amount of its visits. The random walk is simulated by a PageRank algorithm that permits edge weights and non-uniformly-distributed random jumps [1, 24].

First, DoSeR creates a **complete, directed** K-partite graph whose set of nodes V is divided in K disjoint subsets $V_1, ..., V_K$. K denotes the amount of surface forms and V_i

[2] http://aksw.org/Projects/AGDISTIS.html

is the set of generated entity candidates $\{e^i_1, ..., e^i_{|V_i|}\}$ for surface form m_i. Since our graph is K-partite, there are only directed, weighted edges between entity candidates that belong to different surface forms. Connecting the entities that belong to the same surface form would be wrong since the correct target entities of surface forms are determined by the other surface forms' entity candidates (coherence). Consequently, using a complete graph instead results in an accuracy decrease of ≈ 2 percentage points F1.

The edge weights in our graph represent entity transition probabilities (ETP) which describe the likelihood to walk from a node (entity) to the adjacent node. We compute these probabilities by normalizing our semantic similarity measurement (cf. Equation 1). The semantic similarity between two entities is the cosine similarity (cos) of its semantic embeddings (vectors) $vec(e^i_u)$ and $vec(e^j_v)$ stored in the index.

$$ETP(e^i_u, e^j_v) = \frac{cos(vec(e^i_u), vec(e^j_v))}{\sum_{k \in (V \setminus V_i)} cos(vec(e^i_u), vec(k))} \tag{1}$$

Given the current graph, we additionally integrate a possibility to jump from any node to any other node in the graph during the random walk with probability $\alpha = 0.1$. Typical values for alpha (according to the original paper [24]) are in the range $[0.1, 0.2]$. We did not manually integrate jump edges in the graph (as in the transition case), but our PageRank algorithm simulates edges between all node pairs during PageRank computation. We compute a probability for each entity candidate being the next jump target. For this purpose we use the already precomputed prior stored in our index.

Figure 2 shows a possible entity candidate graph of our introducing example. The surface form "TS" has only one entity candidate and consequently has already been linked to the entity "Time Square". The second surface form "New York" is still ambiguous, providing two entity candidates. We omit the jump probability values in the figure to improve visualization.

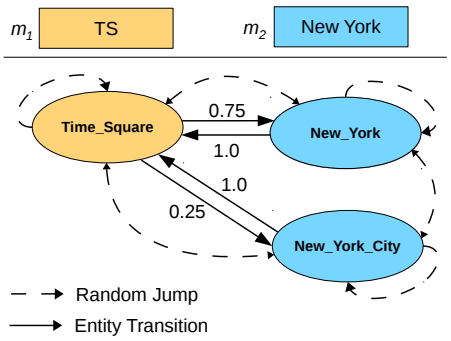

Fig. 2. Entity candidate graph with candidates for the surface forms "TS" and "New York".

After constructing the disambiguation graph, we need to identify the correct entity candidate node. By applying the PageRank algorithm we compute a relevance score for each entity candidate. We empirically choose $it = 50$ PageRank iterations, which is the best trade-off between performance and accuracy in our experiments. Afterwards, the entity candidate e^i_j of a surface form candidate set V_i that provides the highest relevance score is our entity result for surface form m_i. To improve performance, we automatically thin out our disambiguation graph by removing 25% of those edges, whose source and target entities have the lowest semantic similarity. Despite a loss of information, our disambiguation results nearly stay the same.

We also evaluated the graph-based HITS-algorithm [10]. In our experiments, this algorithm performs worse if the input documents contain a bulk of surface forms or the number of generated entity candidates is high (e.g. when using Wikipedia as KB).

5 Semantic Embedding

Embeddings are n-dimensional vectors of concepts which describe the similarities between these concepts via cosine similarity. This has already been well researched for words in literature [15]. In this work, we show how entity embeddings can be generated for the different types of source KBs. First, we briefly introduce Word2Vec, a set of models that are used to produce word embeddings, in Section 5.1. Second, we propose two algorithms to generate appropriate training corpora for Word2Vec in Section 5.2.

5.1 Learning Semantic Embeddings Using Word2Vec

Word2Vec is a group of state-of-the-art, unsupervised algorithms to create word embeddings from (textual) documents initially presented by Mikolov et al. [15]. To train these embeddings, Word2Vec uses a two-layer neural network to process non-labeled documents. The neuronal network architecture is based either on the continuous bag of words (CBOW) or the skip-gram architecture. Using CBOW, the input to the model could be $w_i - 2, w_i - 1, w_i + 1, w_i + 2$, the preceding and following words of the current word w_i. The output of the network is the probability of w_i being the correct word. The task can be described as predicting a word given its context. The skip-gram model works vice-versa: the input to the model is a word w_i and Word2Vec predicts the surrounding context words $w_i - 2, w_i - 1, w_i + 1, w_i + 2$. In contrast to other natural language neuronal network models, the training speed of Word2Vec is very fast and can be further significantly improved by using parallel training. The training time on the Wikipedia corpus (without tables) takes $\approx$ 90 minutes on our personal computer with a 4x3.4GHz Intel Core i7 processor and 16 GB RAM.

An important property of Word2Vec is that it groups the vectors of similar words together in the vector space. If enough data is used for training, Word2Vec makes highly accurate guesses about a word's meaning based on its past appearances. Additionally, the resulting word embeddings capture linguistic regularities, for instance the vector operation vec("President") $-$ vec("Power") $\approx$ vec("Prime Minister"). However, the semantic similarity between two words, which is important in the context of our work, denotes the cosine similarity between the words' Word2Vec vectors.

Since the current version of DoSeR considers the semantic similarity between two entities, we treat entities similar to words. Further, we build our semantic entity embeddings with the help of an entity corpus instead of a textual corpus containing sentences and paragraphs (cf. Section 5.2). To train our entity model, we apply the skip-gram architecture that performs better with infrequent words (less popular entities) [15].

5.2 Corpus Generation

Word2Vec typically accepts a set of corpora containing natural language text as input and trains its word vectors according to the words' order in the corpora. Since we want to learn entity representations only, we have to create an appropriate Word2Vec input corpus file that exclusively comprises entities. The entities' order in the corpus file reflects how entities occur in RDF-KBs or EAD-KBs. In the following, we present how DoSeR creates a suitable Word2Vec corpus basing on one or multiple KBs. Hereby, the outputs of both algorithms are concatenated to create a single corpus file.

EAD-KB: To create a Word2Vec corpus out of entity-annotated natural language documents, we assume to have the entity annotations in the format described in Section 4.1 (Example 2). Next, DoSeR iterates over all documents in the underlying corpus and replaces all available, linked surface forms with its respective target entity identifier. Further, all non-entity identifiers like words and punctuations are removed so that all documents consist of entity identifiers separated by whitespaces only. However, the collocation of entities is still maintained as given by the original document. The resulting documents are concatenated to create a single Word2Vec corpus file. The corpus creation approach for EAD-KBs is explicated in Algorithm 1.

Algorithm 1: Creating a Word2Vec corpus out of a EAD-KB

input : document corpus C, output file
output: word2vec corpus file
forall the $D \in C$ **do**
  $D \longleftarrow replaceSFwithTargetID(D)$
  $D \longleftarrow removeAllNonTargetIDs(D)$
  $appendToOutputFile(D)$

RDF-KB: Originally, Word2Vec trains its word (entity) embeddings based on the order given in a document. Since an RDF-KB does not contain this kind of entity sequence, we model this sequence by random walks between resources in the KB. Hereby, we assume that resources that are directly connected via relations or connected via short relation paths provide cohesiveness. In our case, we create a sequence of those resources that are in our entity target set Ω. For this purpose, we regard an RDF-KB as an **undirected** graph $G_{KB} = (V, E)$ where the nodes V are resources of the KB, the edges E are properties of KB and $x, y \in V, (x, y) \in E \Leftrightarrow \exists p : (x, p, y) \vee \exists p : (y, p, x)$ is an RDF triple in the KB. After that, we perform a random walk on the graph G_{KB}. Whenever the random walk visits a node $x \in V$ we append the entity identifier of node x to the output corpus file, if $x \in \Omega$. The succeeding node $succ(x)$ of x is randomly selected by choosing an adjacent node of x, with probability $\frac{1}{edgesOf(x)}$. Hereby, the function $edgesOf$ counts the number of edges that contact node x. Additionally, we introduce a random variable X_x that provides probabilities to jump to a specific node if a random jump is performed:

$$X_x = P(X_x = x) = \frac{IEF(x)}{\sum_{k \in V} IEF(k)}, \text{ with } IEF(x) = log(\frac{|E|}{edgesOf(x)}) \quad (2)$$

We compute the jump probability from any node to a specific node x by normalizing the respective inverse edge frequency IEF of node x. In our experiments, we use the parameter $\alpha = 0.1$ to perform a random jump. However, values of $0.05 < \alpha < 0.25$ do not significantly affect the resulting Word2Vec model. Furthermore, the parameter θ specifies the number of random walks on the graph. We suggest to use $\theta = 5 * |E|$, which results in $\approx 50M$ random walks for DBpedia. Higher values of θ do not improve the entity embeddings but increase the training time. The corpus creation approach for RDF-KBs is explicated in Algorithm 2.

Algorithm 2: Creating a Word2Vec corpus out of a RDF knowledge base

input : undirected graph G = (V, E), jump random variable X_x, output file
output : word2vec corpus file
parameter: α node jump probability, θ number of walks
$x \longleftarrow drawRandomNode(X_x)$
$walks \longleftarrow 0$
while $walks < \theta$ **do**
  if $x \in \Omega$ **then**
    $appendToOutputFile(x)$
  if $randomInt(100) > (\alpha * 100)$ **then**
    $x \longleftarrow chooseNextNode(x)$; // adjacent node with $p(succ(x)) = \frac{1}{edgesOf(x)}$
  else
    $x \longleftarrow drawRandomNode(X_x)$
  $walks \longleftarrow walks + 1$;

6 Evaluation

Many disambiguation systems rely on RDF-KBs and additionally leverage knowledge from Wikipedia to significantly improve disambiguation accuracy (e.g. DBpedia Spotlight, AIDA). Thus, the aim in our evaluation is two-fold.

First, we compare DoSeR to the current state-of-the-art named entity disambiguation framework AGDISTIS [22] that exclusively makes use of RDF data by default. Therefore, we use the same KB in form of DBpedia and the same entity target set as in AGDISTIS, consisting of named entities only (cf. left column in Table 1).

Second, we compare DoSeR to publicly available entity disambiguation systems that rely on knowledge from Wikipedia. These are DBpedia Spotlight [14], AIDA [7], Wikifier [2] and WAT [17]. Wikifier and WAT use Wikipedia as underlying KB and link surface forms directly to Wikipedia pages (wikification). In contrast, DBpedia Spotlight and AIDA rely on the RDF-KBs DBpedia and YAGO2, while additionally making use of Wikipedia knowledge (e.g. entity priors). Since entities within these three KBs (DBpedia, Wikipedia and YAGO2) provide *sameAs* relations, we can easily compare the disambiguation accuracy while using the same data sets.

To evaluate DoSeR as well as the competitive disambiguation systems we use the GERBIL - General Entity Annotator Benchmark [23] which offers an easy-to-use platform for the agile comparison of annotators using multiple data sets. In GERBIL, we make use of the D2KB task, which evaluates entity disambiguation only. We report the F1, Precision and Recall, aggregated across surface forms (micro-averaged). Spotlight and WAT are integrated in GERBIL by default, whereas we manually downloaded Wikifier and AIDA and installed them on our server with its best settings. For AIDA we downloaded the default entity repository that is suggested as reference for comparison.

Within our experiment we train the Word2Vec model with Gensim [19], an open-source vector space modeling and topic modeling toolkit. On overview of our parameter settings, links to downloadable resources of other systems and some GERBIL result sheets of our experiments can be found here[3]. In the following Section 6.1, we briefly describe the data sets which are used in our experiments. In Section 6.2, we show how DoSeR performs against the other disambiguation systems.

[3] https://github.com/quhfus/DoSeR/wiki/Configurations

Table 1. Class constraints for named entities (persons, organizations and places) only in DBPedia and YAGO3. Prefix **dbo** stands for http://dbpedia.org/ontology/, **foaf** for http://xmlns.com/foaf/0.1/ and **yago** for http://yago-knowledge.org/resource/.

Class	DBpedia	YAGO3
Person	dbo:Person, foaf:Person	yago:yagoLegalActorGeo
Organization	dbo:Organization, dbo:WrittenWork	yago:yagoLegalActorGeo
Place	dbo:Place, yago:YagoGeoEntity	yago:yagoLegalActorGeo

6.1 Data Sets

In the following, we present seven well-known and publicly available data sets which are used in our evaluation. All data sets are integrated in GERBIL and strongly differ in document length and amount of entities per document. Table 2 shows the statistics of our test corpora.

Table 2. Statistics of our test corpora

Data set	Topic	#Doc.	#Ent.	Ent./Doc.
ACE2004	news	57	253	4.44
AIDA/CO-NLL-TestB	news	231	4458	19.40
AQUAINT	news	50	727	14.50
MSNBC	news	20	658	32.90
N3-Reuters	news	128	621	4.85
IITB	mixed	103	11245	109.01
Microposts-2014 Test	tweets	1165	1440	1.24

1. **ACE2004** This data set from Ratinov et al. [18] is a subset of the ACE2004 coreference documents and contains 57 news articles comprising 253 surface forms.
2. **AIDA/CO-NLL-TestB** The AIDA data set [7] was derived from the CO-NLL 2003 task and contains 1.393 news articles. The corpora was split into a training and two test corpora. The second test set has 231 documents with 19.40 entities on average.
3. **AQUAINT** Compiled by Milne and Witten [16], the data set contains 50 documents and 727 surface forms from a news corpus from the Xinhua News Service, the New York Times, and the Associated Press.
4. **MSNBC** The corpus was presented in 2007 by Cucerzan et al. [3] and contains 20 news documents with 32.90 entities per document.
5. **N3-Reuters** This corpus is based on the well-known Reuters-21578 corpus which contains economic news articles. Roeder et al. proposed this corpus in [20] which contains 128 short documents with 4.85 entities on average.
6. **IITB** This manually created data set by Kulkarni et al. [11] with 123 documents displays the highest entity/document-density of all data sets ($\approx$ 109 entities/document).
7. **Microposts-2014 Test** The tweet data set was introduced for the "Making Sense of Microposts" challenge and has very few entities per document on average [23].

6.2 Results

1. Experiment: We compare DoSeR against AGDISTIS, the current state-of-the-art named entity disambiguation framework from 2014, on DBpedia (i.e. DBpedia as core KB and no optional KBs). AGDISTIS is able to disambiguate all entity classes but achieves its best results on named entities [22]. To the best of our knowledge, AGDISTIS is the only available approach that is able to perform named entity disambiguation by using **only** DBpedia knowledge without implementation effort and significant

accuracy drop. We also investigate whether DoSeR performs better on the up-to-date YAGO3 KB compared to the DBpedia KB. To provide a fair comparison in our experiment, we exclusively regard the same named entities (persons, organizations and places) as used in AGDISTIS (cf. Table 1). Further, we present the results after omitting the DBpedia category system during the training process of the semantic embeddings. (i.e. creating the Word2Vec corpus without including http://purl.org/dc/terms/subject). DoSeR performs best on six out of seven data sets (cf. Table 3), with and without using the DBpedia category system (denoted as *DoSeR* and *DoSeR - No Categories*). Both variants attain similar results, but using the DBpedia categories further improves the F-measure by up to 3 percentage points. Despite applying the same candidate generation approach as proposed in AGDISTIS (because no external surface forms are available), DoSeR outperforms AGDISTIS by up to 10% F1 measure (IITB data set). On the other data sets (except MSNBC) the advantage is $\approx$ 5-6% F1 measure. We assume that the groundtruth entities in the MSNBC data set perfectly fit to available relations between entities in DBpedia. A look at the precision values shows that DoSeR links surface forms to entities more accurate (up to 18% precision measure on Microposts-

Table 3. Performance of DoSeR using relations only, DoSeR using relations and categories, and AGDISTIS on seven different data sets using micro F-measure (**F1**).

Data set	Corpus	Approach	F1-Measure	Precision	Recall
ACE2004	DBpedia	DoSeR	0.702	0.795	0.629
		DoSeR - No Categories	**0.706**	**0.800**	**0.632**
		AGDISTIS	0.658	0.696	0.624
	YAGO3	DoSeR	0.679	0.778	0.602
AIDA/CONLL-TestB	DBpedia	DoSeR	**0.616**	**0.697**	**0.552**
		DoSeR - No Categories	0.602	0.684	0.537
		AGDISTIS	0.582	0.628	0.541
	YAGO3	DoSeR	0.608	0.662	0.550
AQUAINT	DBpedia	DoSeR	**0.646**	**0.820**	**0.533**
		DoSeR - No Categories	0.637	0.809	0.525
		AGDISTIS	0.596	0.739	0.499
	YAGO3	DoSeR	0.611	0.754	0.513
MSNBC	DBpedia	DoSeR	0.725	0.763	0.690
		DoSeR - No Categories	0.727	0.765	0.692
		AGDISTIS	**0.751**	**0.772**	**0.730**
	YAGO3	DoSeR	0.735	0.773	0.700
N3 Reuters	DBpedia	DoSeR	**0.731**	**0.817**	**0.661**
		DoSeR - No Categories	0.713	0.791	0.649
		AGDISTIS	0.658	0.721	0.605
	YAGO3	DoSeR	0.725	0.805	0.656
IITB	DBpedia	DoSeR	**0.515**	**0.773**	**0.386**
		DoSeR - No Categories	0.488	0.751	0.362
		AGDISTIS	0.412	0.637	0.304
	YAGO3	DoSeR	0.454	0.697	0.333
Microposts-2014 Test	DBpedia	DoSeR	**0.489**	**0.763**	**0.360**
		DoSeR - No Categories	0.478	0.750	0.351
		AGDISTIS	0.428	0.584	0.337
	YAGO3	DoSeR	0.465	0.703	0.347

2014 Test). The bottle neck which prevents achieving higher F-measures is the absence of surface forms (resulting in a low recall) in the index. The results attained with the YAGO3 KB are sightly worse than those attained in DBpedia ($\approx$ 2-3% F1 measure). However, we still outperform AGDISTIS on six data sets by $\approx$ 4-5% F1 measure.

2. Experiment: We evaluate DoSeR against the entity disambiguation systems Wikifier, DBpedia Spotlight, AIDA and WAT which all leverage Wikipedia knowledge. Similar to the previous experiment we use DBpedia as core KB, but let DoSeR disambiguate all entities in DBpedia (all entities belonging to the *owl:thing* class) instead of named entities only. We present the results of DoSeR using DBpedia only (denoted as *DoSeR*), using DBpedia and surface forms extracted from Wikipedia (denoted as *DoSeR +WikiSF*) as well as using DBpedia as core KB and Wikipedia as optional KB (denoted as *DoSeR + Wiki*). On all variants, we make use of the DBpedia category system.

Table 4 shows the F1 values of DoSeR using different data sources compared to other disambiguation approaches. We also provide the average F1 values across all data sets. Overall, the results of *DoSeR* are slightly worse than those of the previous experiment. This is because our index does not only contain named entities and thus, the entity target set Ω comprises more entities to be disambiguated [26]. Using surface forms from the Wikipedia corpus (*DoSeR +WikiSF*) improves the results from 61.4% F1 to 67.4% F1 on average, which is mainly caused by increased recall values. In this configuration, DoSeR outperforms Spotlight and AIDA by $\approx$ 7% F1 measure on average.

Using Wikipedia as optional KB in DoSeR (*DoSeR +Wiki*) further increases the average F1 values by $\approx$ 10% F1 percentage points and significantly outperforms the other approaches. It also beats the current state-of-the-art approach Wikifier on four out of seven data sets (ACE2004, MSNBC, N3-Reuters Microposts2014-Test). Considering the AIDA-TestB data set, DoSeR performs comparatively poor with 72.2% F1 compared to 84.3% F1 by the WAT system. Analyzing the results on this data set shows that an analysis of the surface forms' textual context is necessary to perform better. In contrast, on the ACE2004 and MSNBC data sets our approach performs exceptionally well with 86.4% F1 and 88.1% F1 respectively. We also performed the same evaluation with disambiguating Wikipedia entities only (Wikipedia as core KB only). The results are very similar to *DoSeR +Wiki*, achieving an average F1 measure of 76.0%.

Table 4. F1 values of DoSeR, DBpedia Spotlight, AIDA, WAT and Wikifier on seven data sets.

Data set	DoSeR	DoSeR (+WikiSF)	DoSeR (+Wiki)	Wikifier	Spotlight	AIDA	WAT
ACE2004	0.681	0.768	**0.864**	0.824	0.713	0.741	0.800
AIDA/CONLL-TestB	0.597	0.735	0.722	0.776	0.593	0.806	**0.843**
AQUAINT	0.638	0.709	0.820	**0.862**	0.713	0.534	0.768
MSNBC	0.719	0.796	**0.881**	0.851	0.511	0.796	0.777
N3-Reuters	0.700	0.718	**0.727**	0.694	0.577	0.571	0.644
IITB	0.497	0.525	0.713	**0.755**	0.447	0.277	0.611
Microposts-2014 Test	0.469	0.464	**0.639**	0.586	0.623	0.412	0.595
Average	0.614	0.674	**0.767**	0.764	0.597	0.591	0.720

Discussion: In our experiments, we show that the semantics of entity relations and categories in RDF-KBs as well as entity annotations in EAD-KBs can be optimally captured by Word2Vec's entity embeddings. In contrast to binary relations, the semantic embeddings allow to compute an accurate entity similarity even there is no direct relation in the KB between these entities. Additionally, these embeddings are robust against noisy information within these KBs. In contrast to other works, our approach does not regard the surrounding contextual words of the surface forms to disambiguate entities. This is a crucial issue especially on the AIDA test data set. For instance, given a set of location names as surface forms, our approach is not able to decide whether the surface forms refer to locations or football clubs. However, we are going to tackle this deficit in the near future. In terms of performance, DoSeR practically disambiguates as fast as AGDISTIS if only a moderate number of entity candidates is available (e.g. Experiment 1). The disambiguation performance decreases in our second experiment, but we will further optimize the PageRank computation by heuristic computations [25].

7 Conclusion

In this work, we present DoSeR a (named) entity disambiguation framework that is knowledge-base-agnostic in terms of RDF-KBs (e.g. DBpedia) and EAD-KBs (e.g. Wikipedia). In this context, we propose how to easily generate semantic entity embeddings with Word2Vec to compute semantic similarities between entities regardless of the type of KBs available (RDF-KBs or EAD-KBs). Our collective disambiguation algorithm relies on the PageRank algorithm, which is applied on an automatically created entity candidate graph. DoSeR outperforms the current state-of-the-art approach for named entity disambiguation on RDF-KBs on DBpedia by up to 10% F1 measure. Further, our approach outperforms AIDA, DBpedia Spotlight, WAT and Wikifier (the current publicly available state-of-the-art disambiguation system for Wikipedia entities), when leveraging Wikipedia knowledge.

8 Acknowledgments

The presented work was developed within the EEXCESS project funded by the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement number 600601.

References

1. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: 7th WWW. pp. 107–117. Elsevier Science Publishers B. V., Amsterdam, The Netherlands (1998)
2. Cheng, X., Roth, D.: Relational inference for wikification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. EMNLP '13 (2013)
3. Cucerzan, S.: Large-scale named entity disambiguation based on Wikipedia data. In: EMNLP-CoNLL. pp. 708–716. ACL, Prague, Czech Republic (June 2007)
4. Ferragina, P., Scaiella, U.: Fast and accurate annotation of short texts with wikipedia pages. IEEE Softw. 29(1), 70–75 (Jan 2012)

5. Han, X., Sun, L.: An entity-topic model for entity linking. In: EMNLP-CoNLL. pp. 105–115. ACL, Stroudsburg, PA, USA (2012)
6. Han, X., Sun, L., Zhao, J.: Collective entity linking in web text: A graph-based method. In: SIGIR. pp. 765–774. ACM, New York, NY, USA (2011)
7. 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. pp. 782–792. ACL, Stroudsburg, PA, USA (2011)
8. Huang, H., Heck, L., Ji, H.: Leveraging deep neural networks and knowledge graphs for entity disambiguation. CoRR abs/1504.07678 (2015)
9. Kataria, S.S., Kumar, K.S., Rastogi, R.R., Sen, P., Sengamedu, S.H.: Entity disambiguation with hierarchical topic models. In: 17th SIGKDD. pp. 1037–1045. ACM, NY, USA (2011)
10. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM 46(5), 604–632 (Sep 1999)
11. Kulkarni, S., Singh, A., Ramakrishnan, G., Chakrabarti, S.: Collective annotation of wikipedia entities in web text. In: 15th SIGKDD. pp. 457–466. ACM, New York, USA (2009)
12. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal (2014)
13. Mahdisoltani, F., Biega, J., Suchanek, F.M.: Yago3: A knowledge base from multilingual wikipedias (2015)
14. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: Shedding light on the web of documents. In: 7th I-Semantics. pp. 1–8. ACM, New York, NY, USA (2011)
15. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
16. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: 17th CIKM. pp. 509–518. ACM, New York, NY, USA (2008)
17. Piccinno, F., Ferragina, P.: From tagme to wat: A new entity annotator. In: First Int. Workshop on Entity Recognition/Disambiguation. pp. 55–62. ERD '14, ACM, NY, USA (2014)
18. Ratinov, L., Roth, D., Downey, D., Anderson, M.: Local and global algorithms for disambiguation to wikipedia. In: ACL. pp. 1375–1384. Stroudsburg, PA, USA (2011)
19. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proc of the LREC 2010 Workshop. pp. 45–50. ELRA, Valletta, Malta (May 2010)
20. Röder, M., Usbeck, R., Hellmann, S., Gerber, D., Both, A.: N3 - a collection of datasets for named entity recognition and disambiguation in the nlp interchange format. In: 9th LREC, 26-31 May, Reykjavik, Iceland (2014)
21. Shen, W., Wang, J., Luo, P., Wang, M.: Linden: Linking named entities with knowledge base via semantic knowledge. In: 21st WWW. pp. 449–458. ACM, New York, NY, USA (2012)
22. Usbeck, R., Ngonga Ngomo, A.C., Röder, M., Gerber, D., Coelho, S.A., Auer, S., Both, A.: Agdistis - graph-based disambiguation of named entities using linked data. In: 14th ISWC. pp. 457–471. Springer-Verlag New York, Inc., New York, NY, USA (2014)
23. Usbeck, R., Röder, M., Ngonga Ngomo, A.C., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., Ferragina, P., Lemke, C., Moro, A., Navigli, R., Piccinno, F., Rizzo, G., Sack, H., Speck, R., Troncy, R., Waitelonis, J., Wesemann, L.: GERBIL – general entity annotation benchmark framework. In: 24th WWW conference (2015)
24. White, S., Smyth, P.: Algorithms for estimating relative importance in networks. In: 9th SIGKDD. pp. 266–275. ACM, New York, NY, USA (2003)
25. Xie, W., Bindel, D., Demers, A., Gehrke, J.: Edge-weighted personalized pagerank: Breaking a decade-old performance barrier. In: 21th SIGKDD. pp. 1325–1334. ACM, NY, USA (2015)
26. Zwicklbauer, S., Seifert, C., Granitzer, M.: From general to specialized domain: Analyzing three crucial problems of biomedical entity disambiguation. In: 26th DEXA 2015, Valencia, Spain. pp. 76–93 (2015)