Automatic Generation of Benchmarks for Entity Recognition and Linking

Axel-Cyrille Ngonga Ngomo[1], Michael Röder[1], Diego Moussallem[1], and Ricardo Usbeck[1]

AKSW Group, University of Leipzig, Germany
{lastname}@informatik.uni-leipzig.de

Abstract. The creation of gold standards for named entity recognition and entity linking is a time-intensive, costly and error-prone task. We hence investigate the automatic generation of benchmark texts with entity annotations for these tasks from structured data, i.e., from Linked Data. The main advantage of automatically benchmarks is that they can be readily generated at any time, are cost-effective while being guaranteed to achieve gold-standard quality. We compare the performance of 11 tools on the benchmarks we generate with their performance on 16 benchmarks that were created manually. Our results suggest that our automatic benchmark generation approach can create benchmarks that have characteristics similar to those of existing benchmarks. The results achieved by existing tools on our automatically generated benchmarks also point to these benchmarks being universally usable for evaluating the performance of modern entity recognition and entity linking systems.

1 Introduction

Benchmarking is of central importance for the objective assessment and development of approaches all around computer science. For example, developments in the database area suggest that benchmarks such as TPC-H were instrumental for the increase of the performance of relational databases by orders of magnitude [29]. In newer times, benchmarking campaigns such as BioASQ [23] have pushed the F-measure achieved by bio-medical question answering systems by more than 5%.

This work was motivated by discrepancies in the performance of tools within the GERBIL [25] platform, which led us to evaluate a sample of 25 documents from the ACE2004 benchmark manually. This evaluation revealed that 195 annotations were missing and 14 annotations were incorrect. A manual evaluation of AIDA/CONLL [21] and OKE2015 [12] revealed similar even if less drastic mistakes in existing benchmarks. Currently, creating benchmarks for named entity recognition (NER) and entity linking (EL), also known as entity disambiguation, is a mostly manual effort. The manual approach has the advantage of yielding benchmarks which reflect human processing. However, this approach to creating benchmarks exhibits significant disadvantages (1) human annotators have to read through every sentence in the corpus and often miss annotations (fatigue, unclear annotation rules), (2) manually created benchmarks are usually small (commonly $< 2,500$ documents, see Table 2); hence (3) they do not allow benchmarking the scalability of existing solutions. Moreover, (4) manual benchmark generation approaches lead to static corpora which tend not to reflect the newest terminology or knowledge bases. For example, several of the benchmarks presented in GERBIL [25] link to outdated versions of Wikipedia or DBpedia and hence contain resource that do not exist anymore.

We argue that automatic methods are a *viable and supplementary* approach for the generation of benchmarks for entity recognition and linking, especially as they address some of the weaknesses of the manual benchmark creation process. In particular, automatic benchmark creation methods (1) alleviate the human annotation error problem by relying on structured data (here, Linked Data in RDF)[1] which explicitly contain the entities to find. In addition, they (2) solve the size problem by being able to generate arbitrarily large benchmarks. Hence, in addition to allowing the measurement of the accuracy of approaches, they also (3) allow benchmarking the scalability of current solutions. Moreover, they (4) can be easily updated to reflect the newest terminology and knowledge bases. *The main contribution of our paper is thus a novel approach for the automatic generation of benchmarks for NER and EL* dubbed BENGAL. Our approach relies on the abundance of structured data in RDF on the Web and is based on verbalizing such data to generate automatically annotated natural-language statements.

The rest of this paper is structured as follows: We begin with an overview of the state of the art in benchmarking NER and EL. Then, in Section 3, we explain our approach towards verbalizing RDF graphs and show how verbalized RDF can be used to create NER, EL and even relation extraction (RE) benchmarks. In Section 4, we compare the features of our generated benchmarks as well as the results achieved by 11 state-of-the-art NER and EL frameworks with the features and results of manually crafted benchmarks. Finally, we discuss the insights provided by our evaluation and possible extensions of our approach in Section 5.

2 Related Work

2.1 Current benchmark datasets for NER and EL

The 2003 CoNLL shared task [21] is probably the most used benchmark dataset. The corpus contains 1,393 manually annotated documents and is usually split into two testing and one training corpus. The ACE2004 and the MSNBC news datasets were used by Ratinov et al. [13] to evaluate their seminal work on linking to Wikipedia. Another often used corpus is AQUAINT, e.g., used by Milne and Witten [8]. The applied human-driven annotations allow for NER, EL and co-reference resolution [10] where annotators manually disambiguated pre-recognized entities. Additionally, the IITB dataset compiles 104 documents from famous websites. Here, the interrater agreement after volunteer annotation was very low since annotators rarely tagged the same tokens. Detailed dataset statistics on some of these benchmarks can be found in Table 2.

Recently, there has been an uptake of publicly available corpora [16, 20]. The Spotlight corpus and the KORE 50 dataset were proposed to show case the usability of RDF-based annotations [7]. The multilingual N3 collection [16] was introduced to widen the

[1] Resource Description Format, see `https://www.w3.org/RDF/`.

scope and diversity of NIF-based corpora. It has shown its usability while the evaluation of disambiguation tools [24] or ensemble-learning based NER tools [19]. Another recent observation is the shift towards micropost documents like tweets. The Microposts2014 corpus [2] which has been used to evaluate NERD-ML [15]. The Open Knowledge Extraction challenge [12] released open, manually created datasets containing NIF-based annotations for RDF entities and classes. Recently, van Erp et al. [26] presented an analysis of existing benchmarks for NER and EL showing that these benchmarks tend to rely on very popular entities while entities with a low pagerank in the knowledge graph are not considered.

2.2 Benchmarking NER and EL

Several benchmarking platforms for NER and EL evaluation have been introduced recently. The BAT-framework [3] is designed to facilitate the benchmarking of NER, EL and concept tagging approaches. BAT compares seven existing entity annotation approaches using Wikipedia. Rizzo et al. [15] present a study of NER and EL systems for annotating newswire and micropost documents. The GERBIL framework [25] addresses the lack of treatment of NIL[2] values within the BAT-framework, is knowledge base agnostic and provides additional wrapping approaches for annotators and datasets.

2.3 Automatic generation of gold standards for NER and EL

Crowd-based approaches for the generation of gold standards/benchmarks for NER and EL commonly annotate existing text using one or more recognizers and then hand over the tasks of refinement and/or linking to crowd workers to improve the quality. For example, Voyer et al. [27] use a pre-trained annotator which annotates each entity. Afterwards, naïve crowd workers take a binary judgement to indicate the type of an entity. Semi-automatic approaches for the automatic generation of benchmarks use existing annotation tools to generate a coherent annotation of existing text. For example, the CALBC silver standard [14] provides biomedical text corpora based on the annotations of several existing tools which are then aligned using a voting scheme.

To the best of our knowledge, BENGAL is the first approach that makes use of structured data and EL benchmarks. In contrast to the approaches presented in van Erp et al. [26], our framework is not biased towards popular resources as it chooses entities following a uniform distribution.

3 The BENGAL approach

Our approach is based on the observation that more than 30 billion facts pertaining to more than 3 billion entities are available in machine-readable form on the Web (i.e., in RDF triples).[3] The basic intuition behind our approach is simple: *Given that NER and*

[2] That is, entities that exist in the real world but not in the reference knowledge base.

[3] Here we use the expression machine-readable in the sense of the Semantic Web, see `https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225`.

EL are often used in pipelines for the extraction of machine-readable facts from text, we can simply invert the pipeline and go from facts to text, thereby using the information in the facts to produce a gold standard that is *guaranteed to contain no errors*. In the following, we begin by giving a more formal overview of RDF. Thereafter, we present how we use RDF to automatically generate NER and EL benchmarks at scale.

3.1 Preliminaries and Notation

RDF. The notation presented herein is based on the RDF 1.1 specification. An RDF graph G is a set of facts. Facts are represented as triples $t = (s, p, o) \in (R \cup B) \times P \times (R \cup B \cup L)$ where R is the set of all resources (i.e., things of the real world), P is the set of all predicates (binary relations), B is the set of all blank nodes (which basically express existential quantification) and L is the set of all literals (i.e., of datatype values). We call the set $R \cup P \cup L \cup B$ our universe and call its elements entities. A fragment of the RDF knowledge base DBpedia[4] is shown below. We will use this fragment in our examples.

```
:AlbertEinstein dbo:birthPlace :Ulm .
:AlbertEinstein dbo:deathPlace :Princeton .
:AlbertEinstein rdf:type dbo:Scientist .
:AlbertEinstein dbo:field :Physics .
:Ulm dbo:country :Germany.
:AlbertEinstein rdfs:label "Albert␣Einstein"@en.
```

Listing 1. Example RDF dataset

Benchmarks. We define a benchmark as a set C of independent and annotated documents D_i. Each document D_i is a sequence of characters $s_{i1} \dots s_{in}$. Each subsequence $s_{ij} \dots s_{ik}$ (with $j < k$) of the document D_i which stands for a resource $r \in R$ is assumed to be marked as such. We model the marking of resources by the function $m : C \times \mathbb{N} \times \mathbb{N} \rightarrow R$ and write $m(D_i, j, k) = r$ to signify that the substring $s_{ij} \dots s_{ik}$ stands for the resource r. In case the substring $s_{ij} \dots s_{ik}$ does not stand for a resource, we write $m(i, j, k) = \epsilon$. Let D_0 be the example shown in Listing 2. We would write $m(D_0, 0, 14) =$ `:AlbertEinstein`.

```
Albert Einstein was born in Ulm.
```

Listing 2. Example sentence

Verbalization. To the best of our knowledge, there are two main works on verbalizing SPARQL, i.e., SPARTIQULATION [5] and SPARQL2NL [9]. Our approach to verbalizing RDF is based on SPARQL2NL because it is extensible by virtue of being bottom-up, i.e., of specifying reusable rules to verbalize atomic constructs (e.g., RDF triples) and to combine their verbalization to sentences. This increases our syntactic variability over purely rule-based approaches. In contrast, SPARTIQULATION [5] assumes the structure of the sentence to be generated to be known and fits the verbalization of the

[4] `http://dbpedia.org`

Table 1. Dependencies (Dep.) used by BENGAL.

Dependency	**Explanation**
cc	Stands for the relation between a conjunct and a given conjunction (in most cases and or or). For example in the sentence John eats an apple and a pear, cc(PEAR,AND) holds. We mainly use this construct to specify reduction and replacement rules.
conj*	Used to build the *conjunction* of two phrase elements, e.g. conj(subj(EAT,JOHN), subj(DRINK,MARY)) stands for John eats and Mary drinks. conj is not to be confused with the logical conjunction $\wedge$, which we use to state that two dependencies hold in the same sentence. For example subj(EAT,JOHN) $\wedge$ dobj(EAT,FISH) is to be read as John eats fish.
dobj	Dependency between a verb and its *direct object*, for example dobj(EAT,APPLE) expresses to eat an/the apple.
nn	The *noun compound modifier* is used to modify a head noun by the means of another noun. For instance nn(FARMER,JOHN) stands for farmer John.
poss	Expresses a possessive dependency between two lexical items, for example poss(JOHN,DOG) expresses John's dog.
subj	Relation between *subject* and verb, for example subj(BE,JOHN) expresses John is.

components into the template. The notation and formal framework for verbalization in BENGAL is presented below is hence also based on SPARQL2NL [9].

Let W be the set of all words in the dictionary of our target language. We define the realization function $\rho : R \cup P \cup L \rightarrow W^*$ as the function which maps each entity to a word or sequence of words from the dictionary. Formally, the goal of the verbalization is to devise the extension of ρ to conjunctions of RDF triples. This extension maps all triples t to their realization $\rho(t)$ and defines how these atomic realizations are to be combined. We denote the extension of ρ by the same label ρ for the sake of simplicity. We adopt a rule-based approach to devise the extension of ρ, where the rules extending ρ to RDF triples are expressed in a conjunctive manner. This means that for premises $P_1, \ldots, P_n$ and consequences $K_1, \ldots, K_m$ we write $P_1 \wedge \ldots \wedge P_n \Rightarrow K_1 \wedge \ldots \wedge K_m$. The premises and consequences are explicated by using an extension of the Stanford dependencies.[5] We rely especially on the constructs explained in Table 1. For example, a possessive dependency between two phrase elements e_1 and e_2 is represented as $\texttt{poss}(e_1, e_2)$. For the sake of simplicity, we sometimes reduce the construct subj(y,x) $\wedge$ dobj(y,z) to the triple (x,y,z) $\in W^3$.

3.2 Overview of the Approach

BENGAL assumes that it is given (1) a RDF graph $G \subseteq (R \cup B) \times P \times (R \cup B \cup L)$, (2) a number of documents to generate, (3) a minimal resp. maximal document size (i.e., number of triples to use during the generation process) d_{min} resp. d_{max},

[5] For a complete description of the vocabulary, see http://nlp.stanford.edu/software/dependencies_manual.pdf.

(4) a set of restrictions pertaining to the resources to generate and (5) a strategy for generating single documents. Given the graph G, BENGAL begins by selecting a set of *seed resources* from G based on the restrictions set using parameter (4). Thereafter, it uses the strategy (5) defined via parameter to select a subgraph of G which contains a randomly selected number d of triples with $d_{min} \leq d \leq d_{max}$. The subgraph is then verbalized. The verbalization is annotated automatically and finally returned as document. This process is repeated as many times as necessary to reach the predefined number of documents. In the following, we present the details of each step underlying our benchmark generation process.

3.3 Seed Selection

Given that we rely on RDF, we model the seed selection by means of a SPARQL SELECT query[6] with one projection variable. Note that we can use the wealth of SPARQL to devise seed selection strageties of arbitrary complexity. However, given that NER and EL frameworks commonly focus on particular classes of resources, we are commonly confronted with the condition that the seeds must be instances of a set of classes, e.g., `:Person`, `:Organization` or `:Place`. The SPARQL query for our example dataset would be as follows:

```
SELECT ?x WHERE { {?x a :Person.} UNION {?x a :Organization.}
    UNION {?x a :Place.} }
```

Listing 3. Example seed selection query.

3.4 Subgraph Generation

Our approach to generating subgraphs is reminiscent of SPARQL query topologies as available in SPARQL query benchmarks such as DBPSB, BSBM, FEASIBLE and FedBench [17]. As these queries (especially the DBPSB[7] and FEASIBLE[8] queries) describe real information needs, their topology must stand for the type of information that are necessitated by applications and humans. We thus distinguish between three main types of subgraphs to be gathered from RDF data: (1) *star graphs* provide information about a particular entity, most commonly a resource (e.g, the short biography of a person); (2) *path graphs* describe the relation between two entities (e.g., the relation between a gene and a side-effect); (3) *hybrid graphs* are a mix of both and commonly describe a specialized subject matter involving several actors (e.g., a description of the cast of a movie).

Star Graph Generation. For each $s_i \in S$, we simply gather all triples of the form $t = (s_i, p, o) \in R \times P \times (R \cup L)$. Note that we do not consider blank nodes as they cannot be verbalized due to the existential quantification they stand for. The triples

[6] SPARQL is the query language for RDF data. The specification can be found at `https://www.w3.org/TR/rdf-sparql-query/`.
[7] `http://aksw.org/Projects/DBPSB`
[8] `http://aksw.org/Projects/Feasible`

are then added to a list $L(s_i)$ sorted in descending order according to a hash function h. After randomly selecting a document size d between d_{min} and d_{max}, we select d random triples from $L(s_i)$. For the dataset shown in Listing 1 and $d = 2$, we would for example get Listing 4.

```
:AlbertEinstein :birthPlace :Ulm .
:AlbertEinstein :deathPlace :Princeton .
```

Listing 4. Example RDF dataset.

Symmetric Star Graph Generation. Same as above with $t = (s_i, p, o)$ or $t = (o, p, s_i)$.

Path Graph Generation. For each $s_i \in S$, we begin by computing list $L(s_i)$ as in the symmetric star graph generation. Then, we pick a random triple (s_i, p, o) or (o, p, s_i) from $L(s_i)$ that is such that o is a resource. We then use o as seed and repeat the operation until we have generated d triples, where d is randomly generated as above. For the example dataset shown in Listing 1 and $d = 2$, we would for example get Listing 5.

```
:AlbertEinstein :birthPlace :Ulm .
:Ulm :country :Germany .
```

Listing 5. Example RDF dataset.

Hybrid Graph Generation. This is a 50/50-mix of the star and path graph generation approaches. In each iteration, we chose and apply one of the two strategies above randomly.

Summary Graph Generation. This last strategy is a specialization of the star graph generation where the set of triples to a resource is not chosen randomly. Instead, for each class (e.g., `:Person`) of the input knowledge base, we begin by filtering the set of properties and only consider properties that (1) have the said class as domain and (2) achieve a coverage above a user-set threshold (60% in our experiments) (e.g., `:birthPlace`, `:deathPlace`, `:spouse`). We then build a property co-occurence graph for the said class in which the nodes are the properties selected in the precedent step and the co-occurence of two properties p_1 and p_2 is the instance r of the input class where $\exists o_1, o_2 : (r, p_1, o_1) \in K \wedge (r, p_2, o_2) \in K$. The resulting graph is then clustered (e.g., by using the approach presented in Ngonga [11]). We finally select the clusters which contain the properties with the highest frequencies in K that allow selecting at least d triples from K. For example, if `:birthPlace` (frequency = 10), `:deathPlace` (frequency = 10) were in the same cluster while `:spouse` (frequency = 8) were in its own cluster, we would chose the pair (`:birthPlace`, `:deathPlace`) and return the corresponding triples for our input resource. Hence, we would return Listing 4 for our running example.

3.5 Verbalization

The verbalization strategy for the first four strategies consists of verbalizing each triple as a single sentence and is derived from SPARQL2NL[9]. To verbalize the subject of the triple $t = (s, p, o)$, we use one of its labels according to Ell et al [4] (e.g., the `rdfs:label`). If the object o is resource, we follow the same approach as for the subject. However, the verbalization of a triple $t = (s, p, o)$ depends mostly on the verbalization of the predicate `p`. If `p` can be realized as a noun phrase, then a possessive clause can be used to express the semantics of (s, p, o). For example, if p can be verbalized as a nominal compound like `birth place`, then the triple can be verbalized as shown in equation 1. In case `p`'s realization is a verb, then the triple can be verbalized as in equation 2.

1. ρ`(s p o)` $\Rightarrow$ `poss(`ρ`(p),`ρ`(s))` $\wedge$ `subj(BE,`ρ`(p))` $\wedge$`dobj(BE,`ρ`(o))`

2. ρ`(s p o)` $\Rightarrow$ `subj(`ρ`(p),`ρ`(s))` $\wedge$`dobj(`ρ`(p),`ρ`(o))`

In our running example, verbalizing (`:AlbertEinstein`, `dbo:birthDate`, `:Ulm`) would thus lead to `Albert Einstein's birth place is Ulm.`, as `birth place` is a noun. In the case of summary graphs, we go beyond the verbalization of single sentences and merge sentences that were derived from the same cluster. For example, if p_1 and p_2 can be verbalized as nouns, then we apply the following rule:

3. ρ`(s p`$_1$ `o`$_1$`)` $\wedge\rho$`(s p`$_2$ `o`$_2$`)` $\Rightarrow$ `conj(poss(`ρ`(p`$_1$`),`ρ`(s))`$\wedge$
   `subj(BE`$_1$`,`ρ`(p`$_1$`)`$\wedge$ `dobj(BE`$_1$`,`ρ`(o`$_1$`))`$\wedge$
   `poss(`ρ`(p`$_2$`),`ρ`(pronoun(s)))`$\wedge$
   `subj(BE`$_2$`,`ρ`(p`$_2$`))`$\wedge$ `dobj(BE`$_2$`,`ρ`(o`$_2$`))`

Note that `pronoun(s)` returns the correct pronoun for a resource based on its type and gender. Therewith, we can generate `Albert Einstein's birthplace is Ulm and his death place is Princeton`.

3.6 Paraphrasing

With this step, BENGAL avoids the generation of large number of sentences that share the same terms and the same structure [28]. To this end, we implemented the paraphrasing steps as proposed in Androutsopoulos et al. [1] as follows: (1) change the structure of the sentence, (2) change the voice from active to passive and (3) look for synonyms based on the context. For step (1) and (2), BENGAL relies on the framework presented in Gatt et al. [6] combined with the Stanford POS tagger [22]. For step (3), we use a round-trip translations according to Somers et al. [18], which points out that the results of a round-trip translation are commonly syntactical different from original text and thus enable the computation of paraphrased text through synonyms across languages. We implemented the round-trip translation by using the Bing translator.

4 Experiments and Results

We generated 13 datasets to evaluate our approach.[9] The first 10 were generated by running our five sub-graph generation methods with and without paraphrasing. The number

[9] The datasets are available at `http://titan.informatik.uni-leipzig.de/mroeder/bengal_datasets.zip`.

of documents was set to 100 while (d_{min}, d_{max}) was set to $(1, 5)$. The 11th dataset tries to show how that BENGAL can be used to evaluate the scalability of approaches. Here, we used the hybrid generation strategy to generate 10,000 documents. The 12th and 13th generated datasets comprise 10 documents each with d_{min} set to 90. For the 12th dataset, we focused on generating a high number of entities in the documents while the 13th dataset contains less entities but the same number of documents. We then compared the 13 generated with 16 manually created gold standards presented below. The comparison was carried out in two ways. First, we assessed the features of the datasets. Then, we compared the micro F-measure of 11 NER and EL frameworks on the manually and automatically generated datasets. We chose to use these 11 frameworks because they are included into GERBIL and this inclusion ensures that their interfaces are compatible and their results comparable.

4.1 Dataset features

Table 2 shows the features of the datasets, where E is the set of entities in the corpus C. The distribution of values for the different features is very diverse across the different manually created datasets (see Figure 1). This is mainly due to (1) different ways to annotate entities and (2) the domains of the datasets (news, description of entities, microposts). As shown in Table 2 and Figure 1, BENGAL can be easily configured to generate a wide variety of datasets with features similar to those of real datasets. This is due to our approach being able to generate benchmarks ranging from (1) benchmarks with sentences containing up to two entities without any unnecessary filler terms in between (high entity density) to (2) benchmarks which contain more datatype information and hence yield a low entity density.

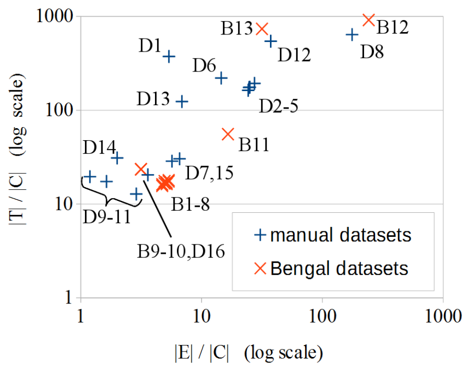

Fig. 1. Average entities per document and average tokens ($|T|$) per document for each dataset.

Table 2. Features of datasets. The datasets B4, B6, B8 and B10 are paraphrased versions of B3, B5, B7 resp. B9 and share similar characteristics.

ID	Name	Doc. $\vert C\vert$	Tokens $\vert T\vert$	Entities $\vert E\vert$	$\vert T\vert/\vert C\vert$	$\vert E\vert/\vert C\vert$	$\vert E\vert/\vert T\vert$
D1	ACE2004	57	21312	306	373.9	5.4	0.01
D2	AIDA/CoNLL-Complete	1393	245008	34929	175.9	25.1	0.14
D3	AIDA/CoNLL-Test A	216	41757	5917	193.3	27.4	0.14
D4	AIDA/CoNLL-Test B	231	37687	5616	163.1	24.3	0.15
D5	AIDA/CoNLL-Training	946	165564	23396	175.0	24.7	0.14
D6	AQUAINT	50	11024	727	220.5	14.5	0.07
D7	DBpediaSpotlight	58	1661	330	28.6	5.7	0.20
D8	IITB	104	66531	18308	639.7	176.0	0.28
D9	KORE50	50	640	144	12.8	2.9	0.23
D10	Microposts2014-Test	1055	20648	1256	19.6	1.2	0.06
D11	Microposts2014-Train	2340	40684	3822	17.4	1.6	0.09
D12	MSNBC	20	10877	747	543.9	37.4	0.07
D13	N3-Reuters-128	128	15842	880	123.8	6.9	0.06
D14	N3-RSS-500	500	15504	1000	31.0	2.0	0.06
D15	OKE 2015 Task 1 evaluation	101	3064	664	30.3	6.6	0.22
D16	OKE 2015 Task 1 train	95	1946	341	20.5	3.6	0.18
B1	BENGAL Path 100	100	1640	514	16.4	5.1	0.31
B2	BENGAL Path Para 100	100	1697	514	17.0	5.1	0.30
B3	BENGAL Star 100	100	1659	494	16.6	4.9	0.30
B4	BENGAL Star Para 100	100	1754	494	17.5	4.9	0.28
B5	BENGAL Sym 100	100	1743	534	17.4	5.34	0.31
B6	BENGAL Sym Para 100	100	1795	534	18.0	5.34	0.30
B7	BENGAL Hybrid 100	100	1563	472	15.6	4.72	0.30
B8	BENGAL Hybrid Para 100	100	1621	472	16.2	4.72	0.29
B9	BENGAL Summary 100	100	2340	315	23.4	3.15	0.13
B10	BENGAL Summary Para 100	100	2356	315	23.6	3.15	0.14
B11	BENGAL Hybrid 10000	10000	556483	165254	55.6	16.5	0.30
B12	BENGAL Hybrid Long 10	10	9162	2417	241.7	916.2	0.26
B13	BENGAL Star Long 10	10	7369	316	31.6	736.9	0.04

4.2 Annotator performance

We used GERBIL [25] to evaluate the performance of 11 annotators on the manually created as well as the BENGAL datasets.The complete result set can be found online.[10] We tested the annotator within an A2KB (annotation to knowledge base) experiment setting: Each document of the corpora was sent to each annotator. The annotator had to find and link all entities to a reference knowledge base (here Wikipedia or DBpedia). We measured both the performance of the NER and the EL steps. For the evaluation of the annotators' performance, we used the weak annotation match. Thus, if an annotator produced $m(D_i, j, k) = r$, it was counted as correct if it overlapped with the marking in the gold standard and if it was linked to the correct entity. Table 3 shows the micro

[10] `http://w3id.org/gerbil/experiment?id=201603140002`

Table 3. Micro F1-scores of the annotators for the A2KB and NER experiments on chosen datasets. N/A means that the annotator stopped with an error.

Experiment	Dataset ID	AIDA	Babelfy	Spotlight	Dexter	E.eu	FOX	FREME	KEA	NERD-ML	TagMe 2	WAT
A2KB	D15	0.536	0.385	0.428	0.441	0.280	0.526	0.510	0.420	0.488	0.484	0.502
	D16	0.585	0.492	0.499	0.576	0.187	0.453	0.603	0.504	0.607	0.696	0.613
	B1	0.678	0.706	0.614	0.716	0.190	0.467	0.718	0.680	0.649	0.693	0.644
	B2	0.669	0.705	0.621	0.717	0.191	0.473	0.699	0.701	0.617	0.690	0.657
	B3	0.675	0.687	0.586	0.669	0.217	0.483	0.649	0.661	0.584	0.707	0.614
	B5	0.652	0.716	0.525	0.704	0.141	0.395	0.609	0.660	0.704	0.715	0.540
	B7	0.710	0.631	0.657	0.700	0.217	0.481	0.672	0.632	0.714	0.699	0.663
	B12	0.713	0.507	0.780	0.837	0.399	0.726	0.814	N/A	0.759	0.525	0.774
	B13	0.400	N/A	0.352	0.457	0.115	0.246	0.503	N/A	0.535	0.539	0.495
NER	D2	0.857	0.608	0.706	0.638	0.842	0.829	0.814	0.510	0.571	0.677	0.896
	D3	0.861	0.612	0.695	0.634	0.818	0.851	0.834	0.565	0.654	0.675	0.889
	D4	0.833	0.621	0.715	0.649	0.844	0.847	0.803	0.531	0.583	0.699	0.881
	D5	0.861	0.604	0.705	0.634	0.848	0.836	0.811	0.497	0.550	0.673	0.900
	D7	0.288	0.323	0.446	0.360	0.344	0.236	0.344	0.721	0.675	0.666	0.300
	D12	0.838	0.554	0.558	0.452	0.845	0.277	0.876	0.305	0.632	0.580	0.811
	D13	0.503	0.413	0.460	0.441	0.607	0.885	0.540	0.343	0.391	0.469	0.604
	D14	0.619	0.495	0.515	0.508	0.760	0.695	0.674	0.317	0.484	0.480	0.730
	D15	0.671	0.542	0.612	0.563	0.610	0.736	0.696	0.638	0.679	0.613	0.717
	D16	0.708	0.597	0.742	0.691	0.622	0.674	0.739	0.665	0.770	0.762	0.767
	B1	0.809	0.718	0.844	0.883	0.718	0.726	0.862	0.723	0.794	0.717	0.798
	B2	0.792	0.721	0.857	0.884	0.702	0.726	0.598	0.744	0.764	0.723	0.804
	B3	0.763	0.692	0.814	0.865	0.730	0.730	0.582	0.828	0.709	0.687	0.784
	B5	0.742	0.727	0.772	0.854	0.690	0.702	0.801	0.701	0.857	0.731	0.735
	B7	0.835	0.637	0.864	0.870	0.747	0.768	0.850	0.720	0.868	0.722	0.814
	B12	0.904	0.535	0.914	0.911	0.776	0.860	0.900	N/A	0.893	0.575	0.904
	B13	0.595	N/A	0.499	0.628	0.418	0.536	0.667	N/A	0.673	0.764	0.701

F1-score of the different annotators on chosen datasets. The manually created datasets showed diverse results. We analyzed the results further by using the F1-scores of the annotators as features of the datasets. Based on these feature vectors we calculated the Pearson correlations between the datasets to identify datasets with similar characteristics.

We used the Pearson correlation instead of the Spearman correlation or Kendalls Tau since they do not take the distance between the ranks into account. Using only ranks is however problematic since some of the annotators achieve similar results on the different datasets but create different orders. Hence, a rank correlation would fail to differentiate between these results and lead to a decreased correlation value. The Pearson correlation on the other hand is more robust against these artefacts.

For example, the Pearson correlations of the F-measures achieved by the different annotators on the AIDA/CoNLL datasets (D2–D5) to each other are very high (0.95–1.00) while the correlation between the results on the Spotlight corpus (D7) and N3-Reuters-128 (D13) is around -0.62. The results on D1 and D12–D15 have a correlation to the AIDA/CoNLL results (D2–D5) that is higher than 0.5. In contrast, the correlations of D7 and D8 to the AIDA/CoNLL datasets range from -0.54 to -0.36. These correlations show the diversity of the manually created datasets and suggest that creating an approach which emulates all datasets is non-trivial.

Like the correlations between the manually created datasets, the correlations between the results achieved on BENGAL datasets and hand-crafted datasets vary. The results on BENGAL correlate most with the results on the OKE 2015 data. The highest correlations are achieved with the OKE 2015 Task 1 dataset and range between 0.89 and 0.92. This suggests that our benchmark can emulate entity-centric benchmarks. The correlation of BENGAL with OKE is however reduced to 0.82 in D13, suggesting that BENGAL can be parametrized so as to diverge from such benchmarks. A similar observation can be made for the correlation D12 and ACE2014, where the correlation increased with the size of the documents in the benchmark. The correlation between the results across BENGAL datasets varies between 0.54 and 1, which suggests that BENGAL can generate a wide range of diverse datasets.

4.3 Scalability

A significant advantage of BENGAL is the scalability of the generated datasets. The number and size of generated documents are only limited by the size of the knowledge base. We created the dataset B11 comprising 10,000 documents using the hybrid graph generation. Every document had between 3 and 20 sentences. The dataset features can be found at the bottom of Table 2.[11] While FOX[12] achieves micro F-measures similar to those on the B7 datasets (0.494 for the A2KB task and 0.763 for the NER task), the experiments on the larger dataset allow deriving that the average runtime of FOX grows from 902 ms/document to 1745 ms/document when the average document size grows from 3 to 12 sentences. While the scaling of other tools will clearly be different from FOX's, this experiment confirms (1) that it is possible to scale the size of the generated datasets, (2) that the quality of the generated dataset and, thus, the performance of the annotators remain stable and (3) we can evaluate how the performance of tools changes depending on document and corpus sizes.

5 Discussion and Conclusion

Overall, our investigation suggests that our approach can automatically generate diverse benchmarks with characteristics similar to those of a large proportion of existing benchmarks. Still, our results also show the limitations of the current implementation

[11] The complete experiment results can be found at `http://gerbil.aksw.org/gerbil/experiment?id=201603140003`.

[12] We chose FOX because it can be run locally and has no restrictions of the number of API calls.

of BENGAL, e.g., the combination of verbalization and paraphrasing has little effect on the performance of the tools, which might be due to BENGAL verbalizing predicates and resources in a limited way. Thus, adding additional surface forms for properties and resources is a future task. However, BENGAL allows studying other aspects of frameworks (such as scalability) which are hard to analyze with current benchmarks. Additionally, we are looking forward to adapt BENGAL to be used in multilingual benchmarking environments. Finally, our results suggest that BENGAL benchmarks can ease the development of NER and EL tools by providing developers with insights into their performance at virtually no cost. Hence, BENGAL (like other benchmarks) can facilitate the push towards better frameworks.

References

1. Ion Androutsopoulos and Prodromos Malakasiotis. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, pages 135–187, 2010.
2. Amparo Elizabeth Cano Basave, Giuseppe Rizzo, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. Making sense of microposts (#microposts2014) named entity extraction & linking challenge. In *Proceedings of 4th Workshop on Making Sense of Microposts (#Microposts2014)*, volume 1141, pages 54–60, 2014.
3. Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. A framework for benchmarking entity-annotation systems. In *22nd WWW Conference*, pages 249–260, 2013.
4. Basil Ell, Denny Vrandečić, and Elena Simperl. Labels in the web of data. In *The Semantic Web–ISWC 2011*, pages 162–176. 2011.
5. Basil Ell, Denny Vrandečić, and Elena Simperl. Spartiqulation: Verbalizing sparql queries. In *The Semantic Web: ESWC 2012 Satellite Events*, pages 117–131. 2012.
6. Albert Gatt and Ehud Reiter. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93, 2009.
7. Pablo N. Mendes, Max Jakob, Andres Garcia-Silva, and Christian Bizer. DBpedia Spotlight: Shedding Light on the Web of Documents. In *7th International Conference on Semantic Systems (I-Semantics)*, pages 1–8, 2011.
8. David Milne and Ian H Witten. Learning to link with wikipedia. In *17th ACM CIKM*, pages 509–518, 2008.
9. Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. Sorry, i don't speak sparql — translating sparql queries into natural language. In *Proceedings of WWW*, pages 977–988, 2013.
10. Axel-Cyrille Ngonga Ngomo, Michael Röder, and Ricardo Usbeck. Cross-document coreference resolution using latent features. In *LD4IE—Linked Data for Information Extraction at ISWC 2014*, 2014.
11. Axel-Cyrille Ngonga Ngomo and Frank Schumacher. Borderflow: A local graph clustering algorithm for natural language processing. In *Computational Linguistics and Intelligent Text Processing*, pages 547–558. Springer, 2009.
12. Andrea-Giovanni Nuzzolese, AnnaLisa Gentile, Valentina Presutti, Aldo Gangemi, Darío Garigliotti, and Roberto Navigli. Open knowledge extraction challenge. In *Semantic Web Evaluation Challenges*, volume 548, page 3. 2015.
13. Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1375–1384, 2011.

14. Dietrich Rebholz-Schuhmann, Antonio José Jimeno Yepes, Erik M Van Mulligen, Ning Kang, Jan Kors, David Milward, Peter Corbett, Ekaterina Buyko, Elena Beisswanger, and Udo Hahn. Calbc silver standard corpus. *Journal of bioinformatics and computational biology*, 8(01):163–179, 2010.
15. Giuseppe Rizzo, Marieke van Erp, and Raphaël Troncy. Benchmarking the extraction and disambiguation of named entities on the semantic web. In *9th LREC*, pages 4593–4600, 2014.
16. Michael Röder, Ricardo Usbeck, Daniel Gerber, Sebastian Hellmann, and Andreas Both. N^3 - A Collection of Datasets for Named Entity Recognition and Disambiguation in the NLP Interchange Format. In *LREC*. European Language Resources Association (ELRA), 2014.
17. Muhammad Saleem, Yasar Khan, Ali Hasnain, Ivan Ermilov, and Axel-Cyrille Ngonga Ngomo. A fine-grained evaluation of sparql endpoint federation systems. *Semantic Web*, pages 1–26, 2015.
18. Harold Somers. Round-trip translation: What is it good for. In *Proceedings of the Australasian Language Technology Workshop*, pages 127–133, 2005.
19. René Speck and Axel-Cyrille Ngonga Ngomo. Ensemble learning for named entity recognition. In *The Semantic Web – ISWC 2014*, pages 519–534. 2014.
20. Nadine Steinmetz, Magnus Knuth, and Harald Sack. Statistical analyses of named entity disambiguation benchmarks. In *1st Workshop on NLP&DBpedia 2013*, pages 91–102, 2013.
21. Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, pages 142–147, 2003.
22. Kristina Toutanova and Christopher D Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70, 2000.
23. George Tsatsaronis, Michael Schroeder, Georgios Paliouras, Yannis Almirantis, Ion Androutsopoulos, Eric Gaussier, Patrick Gallinari, Thierry Artieres, Michael R. Alvers, Matthias Zschunke, and Axel-Cyrille Ngonga Ngomo. BioASQ: A challenge on large-scale biomedical semantic indexing and question answering. In *Proceedings of AAAI Information Retrieval and Knowledge Discovery in Biomedical Text*, 2012.
24. Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Sören Auer, Daniel Gerber, and Andreas Both. AGDISTIS – Graph-Based Disambiguation of Named Entities using Linked Data. In *International Semantic Web Conference (ISWC)*, pages 457–471. 2014.
25. Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, Paolo Ferragina, Christiane Lemke, Andrea Moro, Roberto Navigli, Francesco Piccinno, Giuseppe Rizzo, Harald Sack, René Speck, Raphaël Troncy, Jörg Waitelonis, and Lars Wesemann. GERBIL – General Entity Annotation Benchmark Framework. In *24th WWW conference*, 2015.
26. Marieke van Erp, Pablo Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Joerg Waitelonis. Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job. In *Proceedings of LREC*, 2016.
27. Robert Voyer, Valerie Nygaard, Will Fitzgerald, and Hannah Copperman. A hybrid model for annotating named entity training corpora. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 243–246, 2010.
28. Matt Young et al. *Technical writer's handbook*. University Science Books, 2002.
29. Jianyong Zhang, Anand Sivasubramaniam, Hubertus Franke, Natarajan Gautam, Yanyong Zhang, and Shailabh Nagar. Synthesizing representative i/o workloads for tpc-h. In *Software, IEE Proceedings-*, pages 142–142. IEEE, 2004.