On the Reproducibility of the TAGME Entity Linking System

Faegheh Hasibi[1(✉)], Krisztian Balog[2], and Svein Erik Bratsberg[1]

[1] Norwegian University of Science and Technology, Trondheim, Norway
{faegheh.hasibi,sveinbra}@idi.ntnu.no
[2] University of Stavanger, Stavanger, Norway
krisztian.balog@uis.no

Abstract. Reproducibility is a fundamental requirement of scientific research. In this paper, we examine the repeatability, reproducibility, and generalizability of TAGME, one of the most popular entity linking systems. By comparing results obtained from its public API with (re)implementations from scratch, we obtain the following findings. The results reported in the TAGME paper cannot be repeated due to the unavailability of data sources. Part of the results are reproducible through the provided API, while the rest are not reproducible. We further show that the TAGME approach is generalizable to the task of entity linking in queries. Finally, we provide insights gained during this process and formulate lessons learned to inform future reducibility efforts.

1 Introduction

Recognizing and disambiguating entity occurrences in text is a key enabling component for semantic search [14]. Over the recent years, various approaches have been proposed to perform automatic annotation of documents with entities from a reference knowledge base, a process known as *entity linking* [7,8,10,12,15,16]. Of these, TAGME [8] is one of the most popular and influential ones. TAGME is specifically designed for efficient ("on-the-fly") annotation of short texts, like tweets and search queries. The latter task, i.e., annotating search queries with entities, was evaluated at the recently held Entity Recognition and Disambiguation Challenge [1], where the first and second ranked systems both leveraged or extended TAGME [4,6]. Despite the explicit focus on short text, TAGME has been shown to deliver competitive results on long texts as well [8]. TAGME comes with a web-based interface and a RESTful API is also provided.[1] The good empirical performance coupled with the aforementioned convenience features make TAGME one of the obvious must-have baselines for entity linking research. The influence and popularity of TAGME is also reflected in citations; the original TAGME paper [8] (from now on, simply referred to as the TAGME paper) has been cited around 50 times according to the ACM digital library and nearly 200 times according to Google scholar, at the time of writing. The authors

[1] http://tagme.di.unipi.it/.

N. Ferro et al. (Eds.): ECIR 2016, LNCS 9626, pp. 436–449, 2016.
DOI: 10.1007/978-3-319-30671-1_32

have also published an extended report [9] (with more algorithmic details and experiments) that has received over 50 citations according to Google scholar.

Our focus in this paper is on the repeatability, reproducibility, and generalizability of the TAGME system; these are obvious desiderata for reliable and extensible research. The recent SIGIR 2015 workshop on Reproducibility, Inexplicability, and Generalizability of Results (RIGOR)[2] defined these properties as follows:

- *Repeatability*: "Repeating a previous result under the original conditions (e.g., same dataset and system configuration)."
- *Reproducibility*: "Reproducing a previous result under different, but comparable conditions (e.g., different, but comparable dataset)."
- *Generalizability*: "Applying an existing, empirically validated technique to a different IR task/domain than the original."

We address each of these aspects in our study, as explained below.

Repeatability. Although TAGME facilitates comparison by providing a publicly available API, it is not sufficient for the purpose of repeatability. The main reason is that the API works much like a black-box; it is impossible to check whether it corresponds to the system described in [8]. Actually, it is acknowledged that the API deviates from the original publication,[3] but the differences are not documented anywhere. Another limiting factor is that the API cannot be used for efficiency comparisons due to the network overhead. We report on the challenges around repeating the experiments in [8] and discuss why the results are not repeatable.

Reproducibility. TAGME has been re-implemented in several research papers, see, e.g., [2,3,11], these, however, do not report on the reproducibility of results. In addition, there are some technical challenges involved in the TAGME approach that have not always been dealt with properly in the original paper and accordingly in these re-implementations (as confirmed by some of the respective authors).[4] We examine the reproducibility of TAGME, as introduced in [8], and show that some of the results are not reproducible, while others are reproducible only through the TAGME API.

Generalizability. We test generalizability by applying TAGME to a different task: entity linking in queries (ELQ). This task has been devised by the Entity Recognition and Disambiguation (ERD) workshop [1], and has been further elaborated on in [11]. The main difference between conventional entity linking and ELQ is that the latter accepts that a query might have multiple interpretations, i.e., the output in not a single annotation, but (possibly multiple) sets of entities that are semantically related to each other. Even though TAGME has been developed for a different problem (where only a single interpretation is returned), we show that it is generalizable to the ELQ task.

[2] https://sites.google.com/site/sigirrigor/.
[3] http://tagme.di.unipi.it/tagme_help.html and is also mentioned in [5,18].
[4] Personal communication with authors of [3,8,11].

Before we proceed let us make a disclaimer. In the course of this study, we made a best effort to reproduce the results presented in [8] based on the information available to us: the TAGME papers [8,9] and the source code kindly provided by the authors. Our main goal with this work is to learn about reproducibility, and is in no way intended to be a criticism of TAGME. The communication with the TAGME authors is summarized in Sect. 6. The resources developed within this paper as well as detailed responses from the TAGME authors (and any possible future updates) are made publicly available at http://bit.ly/tagme-rep.

2 Overview of TAGME

In this section, we provide an overview of the TAGME approach, as well as the test collections and evaluation metrics used in the TAGME papers [8,9].

2.1 Approach

TAGME performs entity linking in a pipeline of three steps: (i) parsing, (ii) disambiguation, and (iii) pruning (see Fig. 1). We note that while Ferragina and Scaiella [8] describe multiple approaches for the last two steps, we limit ourselves to their final suggestions; these are also the choices implemented in the TAGME API.

Before describing the TAGME pipeline, let us define the notation used throughout this paper. Entity linking is the task of annotating an input text T with entities E from a reference knowledge base, which is Wikipedia here. T contains a set of entity mentions M, where each mention $m \in M$ can refer to a set of candidate entities $E(m)$. These need to be disambiguated such that each mention points to a single entity $e(m)$.

Parsing. In the first step, TAGME parses the input text and performs mention detection using a dictionary of entity surface forms. For each entry (surface form) the set of entities recognized by that name is recorded. This dictionary is built by extracting entity surface forms from four sources: anchor texts of Wikipedia articles, redirect pages, Wikipedia page titles, and variants of titles (removing parts after the comma or in parentheses). Surface forms consisting of numbers only or of a single character, or below a certain number of occurrences (2) are discarded. Further filtering is performed on the surface forms with low *link probability* (i.e., < 0.001). Link probability is defined as:

$$\text{lp}(m) = P(\text{link}|m) = \frac{link(m)}{freq(m)}, \tag{1}$$

Fig. 1. Annotation pipeline in the TAGME system.

where $freq(m)$ denotes the total number of times mention m occurs in Wikipedia (as a link or not), and $link(m)$ is the number of times mention m appears as a link.

To detect entity mentions, TAGME matches all n-grams of the input text, up to $n = 6$, against the surface form dictionary. For an n-gram contained by another one, TAGME drops the shorter n-gram, if it has lower link probability than the longer one. The output of this step is a set of mentions with their corresponding candidate entities.

Disambiguation. Entity disambiguation in TAGME is performed using a voting schema, that is, the score of each mention-entity pair is computed as the sum of votes given by candidate entities of all other mentions in the text. Formally, given the set of mentions M, the relevance score of the entity e to the mention m is defined as:

$$\text{rel}(m, e) = \sum_{m' \in M - \{m\}} \text{vote}(m', e), \tag{2}$$

where $\text{vote}(m', e)$ denotes the agreement between entities of mention m' and the entity e, computed as follows:

$$\text{vote}(m', e) = \frac{\sum_{e' \in E(m')} \text{relatedness}(e, e') \cdot \text{commonness}(e', m')}{|E(m')|}. \tag{3}$$

Commonness is the probability of an entity being the link target of a given mention [13]:

$$\text{commonness}(e', m') = P(e'|m') = \frac{link(e', m')}{link(m')}, \tag{4}$$

where $link(e', m')$ is the number of times entity e' is used as a link destination for m' and $link(m')$ is the total number of times m' appears as a link. *Relatedness* measures the semantic association between two entities [17]:

$$\text{relatedness}(e, e') = \frac{\log(\max(|in(e)|, |in(e')|)) - \log(|in(e) \cap in(e')|)}{\log(|E|) - \log(\min(|in(e)|, |in(e')|))}, \tag{5}$$

where $in(e)$ is the set of entities linking to entity e and $|E|$ is the total number of entities.

Once all candidate entities are scored using Eq. (2), TAGME selects the best entity for each mention. Two approaches are suggested for this purpose: (i) disambiguation by classifier (DC) and (ii) disambiguation by threshold (DT), of which the latter is selected as the final choice. Due to efficiency concerns, entities with commonness below a given threshold τ are discarded from the DT computations. The set of commonness-filtered candidate entities for mention m is $E_\tau(m) = \{e \in M(e) | \text{commonness}(m, e) \geq \tau\}$. Then, DT considers the top-ϵ

entities for each mention and then selects the one with the highest commonness score:

$$m(e) = \arg\max_{e} \{\text{commonness}(m, e) : e \in E_\tau(m) \wedge e \in \text{top}_\epsilon[\text{rel}(m, e)]\}. \quad (6)$$

At the end of this stage, each mention in the input text is assigned a single entity, which is the most pertinent one to the input text.

Pruning. The aim of the pruning step is to filter out non-meaningful annotations, i.e., assign *NIL* to the mentions that should not be linked to any entity. TAGME hinges on two features to perform pruning: *link probability* (Eq. (1)) and *coherence*. The coherence of an entity is computed with respect to the candidate annotations of all the other mentions in the text:

$$\text{coherence}(e, T) = \frac{\sum_{e' \in E(T) - \{e\}} \text{relatedness}(e, e')}{|E(T)| - 1}, \quad (7)$$

where $E(T)$ is the set of distinct entities assigned to the mentions in the input text. TAGME takes the average of the link probability and the coherence score to generate a ρ score for each entity, which is then compared to the pruning threshold ρ_{NA}. Entities with $\rho < \rho_{\text{NA}}$ are discarded, while the rest of them are served as the final result.

2.2 Test Collections

Two test collections are used in [8]: Wiki-Disamb30 and Wiki-Annot30. Both comprise of snippets of around 30 words, extracted from a Wikipedia snapshot of November 2009, and are made publicly available.[5] In Wiki-Disamb30, each snippet is linked to a single entity; in Wiki-Annot30 all entity mentions are annotated. We note that the sizes of these test collections (number of snippets) deviate from what is reported in the TAGME paper: Wiki-Disamb30 and Wiki-Annot30 contain around 2M and 185K snippets, while the reported numbers are 1.4M and 180K, respectively. This suggests that the published test collections might be different from the ones used in [8].

2.3 Evaluation Metrics

TAGME is evaluated using three variations of precision and recall. The so-called *standard* precision and recall (P and R), are employed for evaluating the disambiguation phase, using the Wiki-Disamb30 test collection. The two other metrics, *annotation* and *topics* precision and recall are employed for measuring the end-to-end performance on the Wiki-annot30 test collection. The annotation metrics (P_{ann} and R_{ann}) compare both the mention and the entity against

[5] http://acube.di.unipi.it/tagme-dataset/.

the ground truth, while the topics metrics (P_{topics} and R_{topics}) only consider entity matches. The TAGME papers [8,9] provide little information about the evaluation metrics. In particular, the computation of the *standard* precision and recall is rather unclear; we discuss it later in Sect. 4.2. Details are missing regarding the two other metrics too: (i) How are overall precision, recall and F-measure computed for the annotation metrics? Are they micro- or macro-averaged? (ii) What are the matching criteria for the annotation metrics? Are partially matching mentions accepted or only exact matches? In what follows, we formally define the annotation and topics metrics, based on the most likely interpretation we established from the TAGME paper and from our experiments.

We write $\mathcal{G}(T) = \{(\hat{m}_1, \hat{e}_1), \ldots, (\hat{m}_m \hat{e}_m)\}$ for ground truth annotations of the input text T, and $\mathcal{S}(T) = \{(m_1, e_1), \ldots, (m_n, e_n)\}$ for the annotations identified by the system. Neither $\mathcal{G}(T)$ nor $\mathcal{S}(T)$ contains NULL annotations. The TAGME paper follows [12], which uses macro-averaging in computing annotation precision and recall:[6]

$$P_{ann} = \frac{|\mathcal{G}(T) \cap \mathcal{S}(T)|}{|\mathcal{S}(T)|}, \quad R_{ann} = \frac{|\mathcal{G}(T) \cap \mathcal{S}(T)|}{|\mathcal{G}(T)|}. \tag{8}$$

The annotation $(\hat{m}, \hat{e})$ matches (m, e) if two conditions are fulfilled: (i) entities match ($\hat{e} = e$), and (ii) mentions match or contain each other ($\hat{m} = m$ or $\hat{m} \in m$ or $m \in \hat{m}$). We note that the TAGME paper refers to "perfect match" of the mentions, while we use a more relaxed version of matching (by considering containment matches). This relaxation results in the highest possible P_{ann} and R_{ann}, but even those are below the numbers reported in [8] (cf. Sect. 4.2).

The topics precision and recall (P_{topics} and R_{topics}) [16] only consider entity matches ($\hat{e} = e$) and are micro-averaged over the set of all texts $\mathcal{F}$:

$$P_{topics} = \frac{\sum_{T \in \mathcal{F}} |\mathcal{G}(T) \cap \mathcal{S}(T)|}{\sum_{T \in \mathcal{F}} |\mathcal{S}(T)|}, \quad R_{topics} = \frac{\sum_{T \in \mathcal{F}} |\mathcal{G}(T) \cap \mathcal{S}(T)|}{\sum_{T \in \mathcal{F}} |\mathcal{G}(T)|}. \tag{9}$$

For all metrics the overall F-measure is computed from the overall precision and recall.

3 Repeatability

By definition (cf. Sect. 1), *repeatability* means that a system should be implemented under the same conditions as the reference system. In our case, the repeatability of the TAGME experiments in [8] is dependent on the availability of (i) the knowledge base and (ii) the test collections (text snippets and gold standard annotations).

[6] As explained later by the TAGME authors, they in fact used micro-averaging. This contradicts the referred paper [12], which explicitly defines P_{ann} and R_{ann} as being macro-averaged.

The reference knowledge base is Wikipedia, specifically, the TAGME paper uses a dump from November 2009, while the API employs a dump from July 2012. Unfortunately, neither of these dumps is available on the web nor could be provided by the TAGME authors upon request. We encountered problems with the test collections too. As we already explained in Sect. 2.2, there are discrepancies between the number of snippets the test collections (Wiki-Disamb30 and Wiki-Annot30) actually contain and what is reported in the paper. The latter number is higher, suggesting that the results in [8] are based only on subsets of the collections.[7] Further, the Wiki-Disamb30 is split into training and test sets in the TAGME paper, but those splits are not available.

Due to these reasons, which could all be classified under the general heading of *unavailability of data*, we conclude that the TAGME experiments in [8] are not repeatable. In the next section, we make a best effort at establishing the most similar conditions, that is, we attempt to reproduce their results.

4 Reproducibility

This section reports on our attempts to reproduce the results presented in the TAGME paper [8]. The closest publicly available Wikipedia dump is from April 2010,[8] which is about five months newer than the one used in [8]. On a side note we should mention that we were (negatively) surprised by how difficult it proved to find Wikipedia snapshots from the past, esp. from this period. We have (re)implemented TAGME based on the description in the TAGME papers [8,9] and, when in doubt, we checked the source code. For a reference comparison, we also include the results from (i) the TAGME API and (ii) the Dexter entity linking framework [3]. Even though the implementation in Dexter (specifically, the parser) slightly deviates from the original TAGME system, it is still useful for validation, as that implementation is done by a third (independent) group of researchers. We do not include results from running the source code provided to us because it requires the Wikipedia dump in a format that is no longer available for the 2010 dump we have access to; running it on a newer Wikipedia version would give results identical to the API. In what follows, we present the challenges we encountered during the implementation in Sect. 4.1 and then report on the results in Sect. 4.2.

4.1 Implementation

During the (re)implementation of TAGME, we encountered several technical challenges, which we describe here. These could be traced back to differences between the approach described in the paper and the source code provided by

[7] It was later explained by the TAGME authors that they actually used only 1.4M out of 2M snippets from Wiki-Disamb30, as Weka could not load more than that into memory. From Wiki-Annot30 they used all snippets, the difference is merely a matter of approximation.

[8] https://archive.org/details/enwiki_20100408.

the authors. Without addressing these differences, the results generated by our implementation are far from what is expected and are significantly worse than those by the original system.

Link probability computation. Link probability is one of the main statistical features used in TAGME. We noticed that the computation of link probability in TAGME deviates from what is defined in Eq. (1): instead of computing the denominator $freq(m)$ as the number of occurrences of mention m in Wikipedia, TAGME computes the number of documents that mention m appears in. Essentially, document frequency is used instead of term (phrase) frequency. This is most likely due to efficiency considerations, as the former is much cheaper to compute. However, a lower denominator in Eq. (1) means that the resulting link probability is a higher value than it is supposed to be. In fact, this change in the implementation means that it is actually not link probability, but more like *keyphraseness* that is being computed. Keyphraseness [15] is defined as:

$$\text{kp}(m) = P(\text{keyword}|m) = \frac{key(m)}{df(m)}, \tag{10}$$

where $key(m)$ denotes number of Wikipedia articles where mention m is selected as a keyword, i.e., linked to an entity (any entity), and $df(m)$ is the number of articles containing the mention m. Since in Wikipedia a link is typically created only for the first occurrence of an entity ($link(m) \approx key(m)$), we can assume that the numerator of link probability and keyphraseness are identical. This would mean that TAGME as a matter of fact uses keyphraseness. Nevertheless, as our goal in this paper is to reproduce the TAGME results, we followed their implementation of this feature, i.e., $link(m)/df(m)$.[9]

Relatedness computation. We observed that the *relatedness* score, defined in Eq. (5), is computed as $1 - \text{relatedness}(e, e')$, furthermore, for the entities with zero inlinks or no common inlinks, the score is set to zero. These details are not explicitly mentioned in the paper, while they have significant impact on the overall effectiveness of TAGME.

Pruning based on commonness. In addition to the filtering methods mentioned in the parsing step (cf. Sect. 2.1), TAGME filters entities with commonness score below 0.001, but it is not documented in the TAGME papers. We followed this filtering approach, as it makes the system considerably faster.

4.2 Results

We report results for the intermediate disambiguation phase and for the end-to-end entity linking task. For all reproducibility experiments, we set the ρ_{NA} threshold to 0.2, as it delivers the best results and is also the recommended value in the TAGME paper.

[9] The proper implementation of link probability would result in lower values (as the denominator would be higher) and would likely require a different threshold value than what is suggested in [8]. This goes beyond the scope of our paper.

Table 1. Results of TAGME repeatability on the Wiki-Disamb30 test collection.

Method	P	R	F
Original paper [8]	0.915	0.909	0.912
TAGME API	0.775	0.775	0.775

Table 2. Results of TAGME repeatability on the Wiki-Annot30 test collection.

Method	P_{ann}	R_{ann}	F_{ann}	P_{topics}	R_{topics}	F_{topics}
Original paper [8]	0.7627	0.7608	0.7617	0.7841	0.7748	0.7794
TAGME API	0.6945	0.7136	0.7039	0.7017	0.7406	0.7206
TAGME-wp10 (our)	0.6143	0.4987	0.5505	0.6499	0.5248	0.5807
Dexter	0.5722	0.5959	0.5838	0.6141	0.6494	0.6313

Disambiguation phase. For evaluating the disambiguation phase, we submitted the snippets from the Wiki-Disamb30 test collection to the TAGME API, with the pruning threshold set to 0. This setting ensures that no pruning is performed and the output we get back is what is supposed to be the outcome of the disambiguation phase. We tried different methods for computing precision and recall, but we were unable to get the results that are reported in the original TAGME paper (see Table 1). We therefore relaxed the evaluation conditions in the following way: if any of the entities returned by the disambiguation phase matches the ground truth entity for the given snippet, then we set both precision and recall to 1; otherwise they are set to 0. This gives us an upper bound for the performance that can be achieved on the Wiki-disamb30 test collection; any other interpretation of precision or recall would result in a lower number. What we found is that even with these relaxed conditions the F-score is far below the reported value (0.775 vs. 0.912). One reason for the differences could be the discrepancy between the number of snippets in the test collection and the ones used in [8]. Given the magnitude of the differences, even against their own API, we decided not to go further to get the results for our implementation of TAGME. We conclude that this set of results is not reproducible, due to *insufficient experimental details* (test collection and metrics).

End-to-end performance. Table 2 shows end-to-end system performance according to the following implementations: the TAGME API, our implementation using a Wikipedia snapshot from April 2010, and the Dexter implementation using a Wikipedia snapshot from March 2013. For all experiments, we compute the evaluation metrics described in Sect. 2.3. We observe that the API results are lower than in the original paper, but the difference is below 10 %. We attribute this to the fact that the ground truth is generated from a 2009 version of Wikipedia, while the API is based on the version from 2012.

Concerning our implementation and Dexter (bottom two rows in Table 2) we find that they are relatively close to each other, but both of them are lower than

the TAGME API results; the relative difference to the API results is –19 % for our implementation and –12 % for Dexter in F_{topics} score. Ceccarelli et al. [3] also report on deviations, but they attribute these to the processing of Wikipedia: "we observed that our implementation always improves over the WikiMiner online service, and that it behaves only slightly worse then TAGME after the top 5 results, probably due to a different processing of Wikipedia." The difference between Dexter and our implementation stems from the parsing step. Dexter relies on its own parsing method and removes overlapping mentions at the end of the annotation process. We, on the other hand, follow TAGME and delete overlapping mentions in the parsing step (cf. Sect. 2.1). By analyzing our results, we observed that this parsing policy resulted in early pruning of some correct entities and led accordingly to lower results.

Our experiments show that the end-to-end results reported in [8] are reproducible through the TAGME API, but not by (re)implementation of the approach by a third partner. This is due to *undocumented deviations from the published description.*

5 Generalizability

To test the generalizability of TAGME, we apply it to a (slightly) different entity linking task: entity linking in queries (ELQ). As discussed in [1,11], the aim of this task is to detect all possible entity linking *interpretations* of the query. This is different from conventional entity linking, where a single annotation is created. Let us consider the query "france world cup 98" to get a better understanding of the differences between the two tasks. In this example, both France and France national football team are valid entities for the mention "france." In conventional entity linking, we link each mention to a single entity, e.g., "france" ⇒ France, "world cup" ⇒ FIFA World Cup. For the ELQ task, on the other hand, we detect all entity linking interpretations of the query, where each interpretation is a set of semantically related entities, e.g., {France, FIFA World Cup} and {France national football team, FIFA World Cup}. In other words, the output of conventional entity linking systems is a set of mention-entity pairs, while entity linking in queries returns set(s) of entity sets.

Applying a conventional entity linker to the ELQ task restricts the output to a single interpretation, but can deliver solid performance nonetheless [1]. TAGME has great potential to be generalized to the ELQ task, as it is designed to operate with short texts. We detail our experimental setup in Sect. 5.1 and report on the results in Sect. 5.2.

5.1 Experimental Setup

Implementations. We compare four different implementations to assess the generalizability of TAGME to the ELQ task: the TAGME API, our implementation of TAGME with two different Wikipedia versions, one from April 2010

and another from May 2012 (which is part of the ClueWeb12 collection), and Dexter's implementation of TAGME. Including results using the 2012 version of Wikipedia facilitates a better comparison between the TAGME API and our implementation, as they both use similar Wikipedia dumps. It also demonstrates how the version of Wikipedia might affect the results.

Datasets and evaluation metrics. We use two publicly available test collections developed for the ELQ task: ERD-dev [1] and Y-ERD [11]. ERD-dev includes 99 queries, while Y-ERD offers a larger selection, containing 2398 queries. The annotations in these test collections are confined to proper noun entities from a specific Freebase snapshot.[10] We therefore remove entities that are not present in this snapshot in a post-filtering step. In all the experiments, ρ_{NA} is set to 0.1, as it delivers the highest results both for the API and for our implementations, and is also the recommendation of the TAGME API. Evaluation is performed in terms of precision, recall, and F-measure (macro-averaged over all queries), as proposed in [1]; this variant is referred to as *strict* evaluation in [11].

5.2 Results

Table 3 presents the TAGME generalizability results. Similar to the reproducibility experiments, we find that the TAGME API provides substantially better results than any of the other implementations. The most fair comparison between Dexter and our implementations is the one against TAGME-wp12, as that has the Wikipedia dump closest in date. For ERD-dev they deliver similar results, while for Y-ERD Dexter has a higher F-score (but the relative difference is below 10 %). Concerning different Wikipedia versions, the more recent one performs better on the ERD-dev test collection, while the difference is negligible for Y-ERD. If we take the larger test collection, Y-ERD, to be the more representative one, then we find that TAGME API > Dexter > TAGME-wp10, which

Table 3. TAGME results for the entity linking in queries task.

Method	ERD-dev			Y-ERD		
	P_{strict}	R_{strict}	F_{strict}	P_{strict}	R_{strict}	F_{strict}
TAGME API	0.8352	0.8062	0.8204	0.7173	0.7163	0.7168
TAGME-wp10 (our)	0.7143	0.7088	0.7115	0.6518	0.6515	0.6517
TAGME-wp12 (our)	0.7363	0.7234	0.7298	0.6535	0.6532	0.6533
Dexter	0.7363	0.7073	0.7215	0.6989	0.6979	0.6984

is consistent with the reproducibility results in Table 2. However, the relative differences between the approaches are smaller here. We thus conclude that the TAGME approach can be generalized to the ELQ task.

[10] http://web-ngram.research.microsoft.com/erd2014/Datasets.aspx.

6 Discussion and Conclusions

TAGME is an outstanding entity linking system. The authors offer invaluable resources for the reproducibility of their approach: the test collections, source code, and a RESTful API. In this paper we have attempted to (re)implement the system described in [8], reproduce their results, and generalize the approach to the task of entity linking in queries. Our experiments have shown that some of the results are not reproducible, even with the API provided by the authors. For the rest of the results, we have found that (i) the results reported in the paper are higher than what can be reproduced using their API, and (ii) the TAGME API gives higher numbers than what is achievable by a third-party implementation (not only ours, but also that of Dexter [2]). Based on these findings, we recommend to use the TAGME API, much like a black-box, when entity linking is performed as part of a larger task. For a reliable and meaningful comparison between TAGME and a newly proposed entity linking method, the TAGME approach should be (re)implemented, like it has been done in some prior work, see, e.g., [2,11].

Post-acceptance responses from the TAGME authors. Upon the acceptance of this paper, the TAGME authors clarified some of the issues that surfaced in this study. This information came only after the paper was accepted, even though we have raised our questions during the writing of the paper (at that time, however, the reply we got only included the source code and the fact that they no longer have the Wikipedia dumps used in the paper). We integrated their responses throughout the paper as much as it was possible; we include the rest of them here. First, it turns out that the public API as well as the provided source code correspond to a newer, updated version ("version 2") of TAGME. The source code for the original version ("version 1") described in the TAGME papers [8,9] is no longer available. This means that even if we managed to find the Wikipedia dump used in the TAGME papers and ran their source code, we would have not been able to reproduce their results. Furthermore, TAGME performs additional non-documented optimizations when parsing the spots, filtering inappropriate spots, and computing relatedness, as explained by the authors. Another reason for the differences in performance might have to do with how links are extracted from Wikipedia. TAGME uses wiki page-to-page link records, while our implementation (as well as Dexter's) extracts links from the body of the pages. This affects the computation of relatedness, as the former source contains 20 % more links than the latter. (It should be noted that this file was not available for the 2010 and 2012 Wikipedia dumps.) The authors also clarified that all the evaluation metrics are micro-averaged and explained how the disambiguation phase was evaluated. We refer the interested reader to the online appendix of this paper for further details.

Lessons learned. Even though we have only partially succeeded in reproducing the TAGME results, we have gained invaluable insights about reproducibility requirements during the process. Lessons learned include the following: (i) all

technical details that affect effectiveness or efficiency should be explained (or at least mentioned) in paper; sharing the source code helps, but finding answers in a large codebase can be highly non-trivial; (ii) if there are differences between the published approach and publicly made available source code or API (typically, the latter being an updated version), those should be made explicit; (iii) it is encouraged that authors keep all data sources used in a published paper (in particular, historical Wikipedia dumps, esp. in some specific format, are more difficult to find than one might think), so that these can be shared upon requests from other researchers; (iv) evaluation metrics should be explained in detail. Maintaining an "online appendix" to a publication is a practical way of providing some of these extra details that would not fit in the paper due to space limits, and would have the additional advantage of being easily editable and extensible.

Acknowledgement. We would like to thank Paolo Ferragina and Ugo Scaiella for sharing the TAGME source code with us and for the insightful discussions and clarifications later on. We also thank Diego Ceccarelli for the discussion on link probability computation and for providing help with the Dexter API.

References

1. Carmel, D., Chang, M.-W., Gabrilovich, E., Hsu, B.-J.P., Wang, K.: ERD'14: Entity recognition and disambiguation challenge. SIGIR Forum **48**(2), 63–77 (2014)
2. Ceccarelli, D., Lucchese, C., Orlando, S., Perego, R., Trani, S.: Dexter: An open source framework for entity linking. In: Proceedings of the Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval, pp. 17–20 (2013)
3. Ceccarelli, D., Lucchese, C., Orlando, S., Perego, R., Trani, S.: Learning relatedness measures for entity linking. In: Proceedings of CIKM 2013, pp. 139–148 (2013)
4. Chiu, Y.-P., Shih, Y.-S., Lee, Y.-Y., Shao, C.-C., Cai, M.-L., Wei, S.-L., Chen, H.-H.: NTUNLP approaches to recognizing and disambiguating entities in long and short text at the ERD challenge 2014. In: Proceedings of Entity Recognition & Disambiguation Workshop, pp. 3–12 (2014)
5. Cornolti, M., Ferragina, P., Ciaramita, M.: A framework for benchmarking entity-annotation systems. In: Proceedings of WWW 2013, pp. 249–260 (2013)
6. Cornolti, M., Ferragina, P., Ciaramita, M., Schütze, H., Rüd, S.: The SMAPH system for query entity recognition and disambiguation. In: Proceedings of Entity Recognition & Disambiguation Workshop, pp. 25–30 (2014)
7. Cucerzan, S.: Large-scale named entity disambiguation based on Wikipedia data. In: Proceedings of EMNLP-CoNLL 2007, pp. 708–716 (2007)
8. Ferragina, P., Scaiella, U.: TAGME: On-the-fly annotation of short text fragments (by Wikipedia entities). In: Proceedings of CIKM 2010, pp. 1625–1628 (2010)
9. Ferragina, P., Scaiella, U.: Fast and accurate annotation of short texts with Wikipedia pages. CoRR (2010). abs/1006.3498
10. Han, X., Sun, L., Zhao, J.: Collective entity linking in web text: A graph-based method. In: Proceedings of SIGIR 2011, pp. 765–774 (2011)
11. Hasibi, F., Balog, K., Bratsberg, S.E.: Entity linking in queries: tasks and evaluation. In: Proceedings of the ICTIR 2015, pp. 171–180 (2015)

12. Kulkarni, S., Singh, A., Ramakrishnan, G., Chakrabarti, S.: Collective annotation of Wikipedia entities in web text. In: Proceedings of KDD 2009, pp. 457–466 (2009)
13. Medelyan, O., Witten, I.H., Milne, D.: Topic indexing with Wikipedia. In: Proceedings of the AAAI WikiAI Workshop, pp. 19–24 (2008)
14. Meij, E., Balog, K., Odijk, D.: Entity linking and retrieval for semantic search. In: Proceedings of WSDM 2014, pp. 683–684 (2014)
15. Mihalcea, R., Csomai, A.: Wikify!: Linking documents to encyclopedic knowledge. In: Proceedings of CIKM 2007, pp. 233–242 (2007)
16. Milne, D., Witten, I.H.: Learning to link with Wikipedia. In: Proceedings of CIKM 2008, pp. 509–518 (2008)
17. Milne, D., Witten, I.H.: An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In: Proceedings of AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy, pp. 25–30 (2008)
18. 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 annotator benchmarking framework. In: Proceedings of WWW 2015, pp. 1133–1143 (2015)