A Hybrid Approach for Entity Recognition and Linking

Julien Plu, Giuseppe Rizzo, Raphaël Troncy

EURECOM, Sophia Antipolis, France
{julien.plu,giuseppe.rizzo,raphael.troncy}@eurecom.fr

Abstract. Numerous research efforts are tackling the entity recognition and entity linking tasks resulting in a large body of literature. One could roughly categorize the proposed approaches in two different strategies: linguistic-based and semantic-based methods. In this paper, we present our participation to the OKE challenge, where we experiment with a hybrid approach, which combines the strength of a linguistic-based method augmented by a high coverage in the annotation obtained by using a large knowledge base as entity dictionary. The main goal of this hybrid approach is to improve the extraction and recognition level to get the best recall in order to apply a pruning step. On the training set, the results are promising and the breakdown figures are comparable with the state of the art performance of top ranked systems. Our hybrid approach has been ranked first to the OKE Challenge on the test set.

Keywords: Entity Recognition, Entity Linking, Entity Filtering, Learning to Rank, OKE Challenge

1 Introduction

The first task of the Open Knowledge Extraction (OKE) challenge organized at ESWC 2015 aims to advance research in entity extraction, typing (recognition) and linking for Knowledge Base population. In this paper, we present a hybrid approach for extracting, typing and linking entities from textual documents, to a targeted knowledge base that has been indexed beforehand.

Following the challenge requirements, we make use of the 2014 snapshot of DBpedia as the targeted knowledge base. Our proposed workflow is broken down into three tasks: entity recognition, entity linking and entity pruning. Entity recognition is composed of two subtasks: *i)* entity extraction that refers to the task of spotting mentions that can be entities in the text and *ii)* entity typing that refers to the task of assigning them a proper type. In the following, we use the terms extraction and typing to refer to those two subtasks. Entity linking refers to the task of disambiguating the mention in a targeted knowledge base, and it is also often composed of two subtasks: generating candidates and ranking them according to various scoring functions. Entity pruning aims to filter out candidates that are unlikely to be relevant for the domain considered.

The remainder of this paper is structured as follows. We first present some recent related work for both the entity recognition and the entity linking tasks (Section 2). Next,

we describe the modular architecture (Section 3) and we detail its current implementation (Section 4). We present our experiment settings and we provide preliminary results on the OKE challenge training dataset (Section 5). Finally, we conclude and outline some future work (Section 6).

2 Related Work

In this section, we present several top-performing systems that recognize and link entities in text. We distinguish the approaches proposed for the entity recognition (Section 2.1) and the entity linking (Section 2.2) steps.

2.1 Entity Recognition

Numerous approaches have been proposed to tackle the task of recognizing entities in a text. Amongst the recent and best performing systems, WAT builds on top of TagME algorithms and follows the three steps approach we are advocating: extraction, linking and pruning [9]. For the extraction, a gazetteer that contains wiki-anchors, titles and redirect pages with a list of all their possible links ranked according to a probability score is used. The extraction performance can also be tuned with an optional binary classifier (SVM with linear or RBF kernel) using statistics (features) for each entity referenced in the gazetteer. For typing the entities, WAT relies on OpenNLP NER with the types `PERSON`, `LOCATION` and `ORGANIZATION`. One limitation of this method is that anything matching an entry in the gazetteer, including terms that are common words such as verbs or prepositions, is extracted. Furthermore, the recognition of an entity is limited to the kind of mentions that can be typed by OpenNLP NER. If a mention does not exist in Wikipedia, it cannot be extracted by the WAT system.

Similar to WAT, DBpedia Spotlight uses a gazetteer containing a set of labels from the DBpedia lexicalization dataset for the extraction step [7]. More precisely, the LingPipe Exact Dictionary-Based Chunker with the Aho-Corasick string distance measure is being used. Extracts that only contain verbs, adjectives, adverbs and prepositions can be detected using the LingPipe part-of-speech tagger and then discarded. For the typing step, DBpedia Spotlight re-uses the type of the link provided by DBpedia. The limitation of this method is again the DBpedia dependency, since mentions that do not exist in this knowledge base cannot be extracted nor recognized.

A different kind of approach is the one developed by AIDA [5]. For the recognition step, AIDA uses Stanford NER. A limitation of this approach is that it becomes dependent of the specific model used by the CRF algorithm of Stanford NER. A comparable approach to ours is the one used in Babelfy [8]. For the extraction step, a part-of-speech tagger is used in order to identify the segments in the text which contain at least one noun and that are substring of the entities referenced in BabelNet. For the typing step, the Babelnet categories are used. The limitation of this approach is that only entities appearing in BabelNet can be extracted which prevents to recognize “emerging” entities [4].

2.2 Entity Linking

Once recognized (extracted and typed), entities are linked (disambiguated) according to a reference knowledge base. Various approaches are again reported in the literature. The WAT system uses two methods, namely voting-based and graph-based algorithms [9]. The voting-based approach assigns one score to each entity. The entity having the highest score is then selected. The graph-based approach builds a graph where the nodes correspond to mentions or candidates (entities) and the edges correspond to either mention-entity or entity-entity relationships, each of these two kinds of edges being weighted with three possible scores: *i)* identity, *ii)* commonness that is the prior probability $Pr(e|m)$ and *iii)* context similarity that is the BM25 similarity score used by Lucene[1]. The goal is to find the subgraph that interlinks all the mentions.

AIDA uses a similar approach than the graph-based method of WAT. The graph is built in the same way but only one score for each kind of edge (mention-entity or entity-entity) is proposed. The score used to weight the mention-entity edges is a combination of similarity measure and popularity while the score used to weight the entity-entity edges is based on a combination of Wikipedia-link overlap and type distance [5]. Another graph-based approach is the one used by Babelfy. Two main algorithms have been developed: random walk and a heuristic for finding the subgraph that contains most of the relations between the recognized mentions and candidates. The nodes are pairs (mention,entity) and the edges correspond to existing relationships in BabelNet which are scored. The semantic graph is built using word sense disambiguation (WSD) that extracts lexicographic concepts and entity linking for matching strings with resources described in a knowledge base.

In contrast, DBpedia Spotlight relies on the so-called TF*ICF (Term Frequency-Inverse Candidate Frequency) score computed for each entity. The goal of this score is to show that the discriminative strength of a mention is inversely proportional to the number of candidates it is associated with. This means that a mention that commonly co-occurs with many candidates is less discriminative.

The limitation of systems such as AIDA, Babelfy and DBpedia Spotlight is that they do not include a pruning step that would remove possible false positive candidates. This requires a strong entity recognition system since precision and recall can only fall down at the linking stage.

3 A Hybrid Approach for Entity Recognition and Linking

Our proposed system implements a three steps approach: *i)* named entity recognition, *ii)* named entity linking, and *iii)* named entity pruning. In the following, we detail each of those steps.

3.1 Named Entity Recognition

We rely on a linguistic approach for the first stage of the system. More precisely, we rely on the grammatical meaning of the mentions that are spotted and typed in a text. This ensures a robust performance for well-written texts. Those linguistic approach are:

[1] http://lucene.apache.org/

1. Part-Of-Speech tagging system where we will only keep the singular and plural proper nouns.
2. Named Entity Recognition system to extract and type the named entities.
3. Gazetteers to re-enforce the extraction bringing a robust spotting for well-known mentions.

3.2 Named Entity Linking

This step is composed of three sub-tasks: *i)* entity generation, where an index is built on top of both DBpedia2014[2] and a dump of the Wikipedia articles[3] dated from October 2014 to get possible candidates; *ii*) filtering candidates based on direct inbound and outbound links from Wikipedia; *iii)* entity ranking based on a proposed ranking function. If an entity does not have an entry in the knowledge base, we normally link it to NIL following the TAC KBP convention [6]. However, for the OKE challenge, we do not make use of NIL but we instead create a new URI to describe not-in-the-knowledge-base entities in order to populate DBpedia.

The core of this part grounds on the index created on top of the DBpedia 2014 Knowledge Base and the Wikipedia dump. Each record of the index has a key which corresponds to a DBpedia resource, while the features are listed in Table 1.

ID	Feature	Definition
1	title	the title of the entity
2	URI	the URI associated to the entity
3	redirects	the list of all the redirect pages associated to the entity
4	disambiguation	the title of the disambiguation pages associated to the entity if there is at least one
5	types	the full type hirarchy of the entity, from the highest to the fine-grained type
6	pageRank	the PageRank score of the DBpedia resource corresponding to the entity
7	hits	the HITS score of the DBpedia resource corresponding to the entity
8	inlinks	the number of inLinks of the DBpedia resource corresponding to the entity
9	outlinks	the number of outLinks of the DBpedia resource corresponding to the entity
10	length	the length in number of characters of the associated Wikipedia page of the entity
11	numRedirects	the number of redirects links associated to the entity
12	surfaceForms	the different surface forms used to call the entity in all the Wikipedia articles
13	quotes	the direct outbound links and the number of time they appear in the article of the corresponding entity.

Table 1. List of features contained in the index and used by the pruning algorithm

For each mention, we have potentially many candidates, while some of them have to be filtered out because they are not related to the context. With all the candidates

[2] `http://wiki.dbpedia.org/services-resources/datasets/datasets2014`

[3] `https://dumps.wikimedia.org/enwiki/`

of each mention, we create a graph and we find the densest graph between all of these candidates, similarly to [8]. Our approach is, however, slightly different: we use the feature number 13 (quotes) described in the Table 1 and not BabelNet in order to build the graph. The edges of the graph are weighted according to the number of occurrence of the link between each candidates. For example, given the Wikipedia article describing the Eiffel Tower, if there is one outbound link to Paris in Texas and three to Paris in France, both candidates (Paris in Texas and Paris in France) will be kept. However, the weight of Paris in France will be higher than the one of Paris in Texas. In case all candidates of a mention do not have any relation with any other candidate of the other mentions, all its candidates are kept.

To create those pairs, we used an in-house library to parse the Wikipedia dump. We first tried several libraries that parse Wikipedia such as Sweble[4], GWTWiki[5] and wikipedia-parser[6]. However, these libraries are either too complex to use for the simple extraction we need or too greedy in terms of memory. We have therefore developed our own library in order to extract the pairs (Wikipedia article title, number of times the title appears in the article).

Given an extracted mention, we implement a ranking algorithm based on a string similarity measure between the extracted mention and the title of the link, the set of redirect and the set of disambiguation pages. The rank score of the link, computed by the rank function $r(l)$, is then weighted by a Page Rank score of the referenced resources.

$$r(l) = (a \cdot L(m, title) + b \cdot max(L(m, R)) + c \cdot max(L(m, D))) \cdot PR(l) \qquad (1)$$

where $a = \frac{4}{7}, b = \frac{1}{7}, c = \frac{2}{7}$ are empirically defined. In our experiment, L corresponds to the Levenshtein distance, R and D are respectively the set of redirect and disambiguation pages and PR refers to the PageRank score of the link.

3.3 Named Entity Pruning

At this stage, each extracted mention has been either linked to a DBpedia resource or to NIL. Applying a supervised learning approach, we plan to increase the precision of the system by discarding mentions that are not in the scope of the ones observed in the labeled data. The prediction model is built using the features from 6 to 11 listed in Table 1 and the rank function $r(l)$.

4 System Implementation

We derived three different pipelines of the proposed system that we name respectively **Pipeline_1**, **Pipeline_2** and **Pipeline_3**. In the reminder of this section, we describe the different configurations of those pipelines. The entity linking and entity pruning steps are the same for all the three pipelines while the entity recognition step has a

[4] `http://sweble.org/`
[5] `https://code.google.com/p/gwtwiki/`
[6] `https://github.com/Stratio/wikipedia-parser`

different configuration for the three pipelines: **Pipeline_1** favors a linguistic approach with gazetteer, **Pipeline_2** uses a supervised NER model and **Pipeline_3** combines the two approaches.

4.1 Pipeline_1

We generate candidates by selecting the proper noun in the singular (NNP) and the plural form (NNPS) of the part-of-speech tagging. Our POS tagging system is the Stanford NLP POS Tagger [10] with the model *english-bidirectional-distsim* trained on WSJ[7] sections 0-18 using a bidirectional architecture and including word shape and distributional similarity features.

To increase the coverage of the system in correctly annotating `dul:Role` entities, the current pipeline implements two gazetteers for job names and nationalities. The two gazetteers are built using the list of jobs and nationalities from the corresponding English Wikipedia pages. This process generates the types of the extracted entities, and they are linked to them.

Each proper noun is then looked up in the index, and a set of matching links are retrieved. Those links are then sorted according to the ranking function $r(l)$, and the first one is considered to be the entity link of the mention. The index is built using Lucene v5 and requires 44 hours to be built on a 20 core CPU at 2.5Ghz with 64GB RAM machine. A possible improvement could be to parallelize, or distribute this process in order to decrease the index building time.

At this stage, the entities typed are the `dul:Role` ones. For the others, a set of manual alignment drives the typing process. The alignments are meant to map the DBpedia types with the entities typed as `dul:Person`, `dul:Location` and `dul:Organization`. Often, in retrieving the whole hierarchy of the fine-grained type from a DBpedia resource, the type given is this hierarchy. Traversing the T-Box, we label the entity with the type in the hierarchy learned from the manual alignment process. The `dul` types used have the same name than the one used by DBpedia so the alignment is quite simple: `dul:Place` $\sim$ `dbpedia:Place`, `dul:Person` $\sim$ `dbpedia:Person` and `dul:Organization` $\sim$ `dbpedia:Organisation`.

To favor the precision, we filter out the entities that do not follow the ones observed in the labeled data. We use the so-called pruning stage, which relies on a properly trained KNN classifier [1] with the features set listed in 3.3. To train this classifier, we annotate each value coming from the results as false if they do not appear in the gold standard (training set) and as yes if they appear.

4.2 Pipeline_2

A properly trained Stanford NER [3] is used as a named entity recognizer. The type is statistically predicted according to the observed labels in the training data.

The linking step follows the same strategy than the one described for the Pipeline_1. Hence, each named entity is looked up in the index, and a set of matching links are retrieved which are again sorted according to the rank function $r(l)$. Similarly, we use

[7] `http://www.wsj.com`

the so-called pruning stage (using a KNN classifier trained with the features set listed in 3.3) to increase the precision.

4.3 Pipeline_3

This pipeline presents a hybrid approach which implements both an annotation mechanism leaded by a properly trained supervised learning entity recognizer, Stanford NER [3], and a re-enforced mechanism of NNP/NNPS detection, Stanford POS tagger [10] trained with newswire content.

We do make use of our two previously defined gazetteers for job names and nationalities in order to increase the coverage of the system in correctly extracting `dul:Role` entities.

Sometimes, at least two extractors (Stanford NER, Stanford POS tagger or the gazetteer) extract overlapped mentions. For example, given the two extracted mentions *States of America* from Stanford NER and *United States* from Stanford POS tagger (both with the settings described in the previous sections), we detect that there is an overlap between both mentions, so we take the union of both boundaries to create a new mention and we remove the two others. We obtain the mention *United States of America* with the type provided by Stanford NER. In the case that one mention is included in another one (nested mentions) we take the longest one. For example, if *United States* and *States* are extracted, only the first one will be kept while the second one will be removed. If the one removed comes from Stanford NER, the original type associated to the removed mention is kept.

The linking step follows again the same strategy than in the previous two pipelines where we use our index to look up entities and to return matching links. The same pruning stage is also used for increasing precision.

5 Experimental Settings and Results

5.1 Statistics of the Oracle

The training dataset provided by the OKE challenge organizers is composed of a set of 95 annotated sentences using the NIF ontology[8]. The average length of the sentences is 124 chars. In total, the dataset contains 337 mentions corresponding to 290 distinct entities that belong to one of the four types: `dul:Place`, `dul:Person`, `dul:Organization` and `dul:Role`. 256 entities (88%) are linked within DBpedia, while 33 (12%) are not. The breakdown of those annotations per type is provided in the Table 2.

5.2 Experimental Settings

We applied a 4-fold cross validation of the released training set. In each fold of the cross validation, a train and a test sets are generated and respectively used for building the supervised learning models and for benchmarking the output of the model with the expected results of the test set.

[8] `http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#`

type	nb mentions	nb entities	nb mentions disambiguated (%)	nb entities disambiguated (%)
dul:Place	62	61	58 (93%)	57 (93%)
dul:Person	126	87	110 (87%)	71 (81%)
dul:Organisation	98	95	88 (90%)	85 (89%)
dul:Role	51	47	47 (92%)	43 (91%)
Total	337	290	303 (90%)	256 (88%)

Table 2. Statistics of the oracle

5.3 Results on the Training Set

We have tested the three pipelines against the OKE training set provided. We only consider strict match for the extraction (exact boundaries), recognition (exact type) and linking (exact disambiguation uri) tasks. We use the neleval scorer[9] to compute our performance, given the measures detailed in Table 3.

task	measure
Extraction	`strong_match_mention`
Recognition	`strong_typed_mention_match`
Linking	`strong_link_match`

Table 3. Measures used in the evaluation for each task

The results are divided in two parts: Table 4 are the results obtained for each of our three pipelines without running the pruning step. Tables 5 are instead the results obtained when running the pruning step for each pipeline.

Task	*Pipeline_1*			*Pipeline_2*			*Pipeline_3*		
	P	R	F_1	P	R	F_1	P	R	F_1
Extraction	49.48	65	56.18	**93.43**	85.95	**89.53**	83.55	**93.5**	88.2
Recognition	30.6	40.25	34.75	**93.05**	85.63	**89.15**	81.65	**91.38**	86.23
Linking	35.83	50.53	41.95	**65.98**	43.13	**52.13**	53.7	**46.63**	49.9

Table 4. Breakdown figures per task on the OKE challenge training set for the three pipelines without the pruning. Higher values per row for each metric are in bold.

We can see that the pruning step increases significantly the precision but at the cost of decreasing the recall. It overall performs poorly as it removes too many mentions to get good results at the linking stage. Nevertheless, it provides correct results at the recognition stage. This idea has been inspired by the WAT system [9]. However, the features we choose probably differ from the ones used in WAT resulting in this serious performance drop. We stay positive on the fact that a pruning step can typically help

[9] `https://github.com/wikilinks/neleval`

Task	*Pipeline_1*			*Pipeline_2*			*Pipeline_3*		
	P	R	F_1	P	R	F_1	P	R	F_1
Extraction	44.18	12.6	19.58	**94.53**	**49.63**	**64.75**	88.78	41.93	56.7
Recognition	25.18	7.1	11.08	**93.93**	**49.3**	**64.33**	87.28	41.23	55.75
Linking	32.93	9.98	15.3	**69.98**	**19.18**	**29.95**	55.98	21.13	30.6

Table 5. Breakdown figures per task on the OKE challenge training set for the three pipelines with the pruning. Higher values per row for each metric are in bold.

increasing the precision when a real high recall at the recognition level is obtained. In terms of recall at the recognition level, the *Pipeline_3* without pruning provides the best results. This means that our hybrid approach (mix between NLP, NER and gazetteer approaches) is the most appropriate one to get a high recall at the recognition level enabling to apply a pruning strategy to improve the precision. We observe that there is still a margin of progress to correct the performance drop between the recognition stage and the final results at the linking stage.

5.4 Comparison With Other Tools on the Training Set

We have developed a process to evaluate the performance of three other tools on the same dataset, namely AIDA[10], TagMe[11] and DBpedia Spotlight[12]. The results are presented in the Table 6. The recognition level is not evaluated since both TagMe and AIDA always provide a fine-grained type in a specific ontology while we use the DBpedia one. We used the public API of those systems while applying the default settings for each one.

Task	*AIDA*			TagMe			DBpedia Spotlight		
	P	R	F_1	P	R	F_1	P	R	F_1
Extraction	**69.4**	49.65	57.78	49.15	**85.6**	**62.43**	35.55	52.18	42.25
Linking	**54.23**	43.1	47.98	42.43	**82.43**	**55.98**	22.95	37.35	28.4

Table 6. Breakdown figures per task on the OKE challenge training set for AIDA, Tagme and DBpedia Spotlight. Higher values per row for each metric are in bold.

TagME clearly outperforms all systems at the linking level. Nevertheless, the results show that our hybrid approach provides the best results at the extraction stage, motivating the need for researching better linking strategy.

5.5 Results on the Test Set

The official figures of the challenge are published at `https://github.com/anuzzolese/oke-challenge#results`. The figures provided on the official web site do not

[10] `https://gate.d5.mpi-inf.mpg.de/webaida/`
[11] `http://tagme.di.unipi.it/`
[12] `http://dbpedia-spotlight.github.io/demo/`

provide a breakdown view that we propose in the Table 7. We have slightly modified the test set in order to correct visible errors such as mix of several links and phrases for the same entity, for example:
oke:He rdfs:label "4,"@en, " h"@en, " Z"@en, "he"@en, "He"@en .
oke:He owl:sameAs dbpedia:Albert_Fert, dbpedia:Shinya_Yamanaka .

We have then computed the breakdown figures shown in Table 7 using the neleval scorer, as we did for the training set. The pipeline used is the *Pipeline_3* without the pruning stage.

Task	P	R	F_1
Extraction	48.3	39	43.2
Recognition	48.5	44.8	46.6
Linking	69.8	65.1	67.4

Table 7. Breakdown figures per task on the OKE challenge test set for the pipeline 3 without the pruning.

6 Conclusion and Future Work

In this paper, we have described three different pipelines of a hybrid system we have developed to address the OKE challenge. We show that a successful approach relies on effectively using a hybrid approach, which exploits both linguistic features and semantic features as one can extract and index from a large Knowledge Base such as DBpedia. As future work, we plan to focus on improving the linking task by doing better graph based algorithms with a more accurate ranking function and to further develop our pruning strategy by reviewing the list of feature used to show the full potential of our hybrid approach. We plan as well to improve the way we build and make use of gazetteers in order to further increase the recall at the extraction stage.

Acknowledgments

This work was partially supported by the EIT Digital 3cixty project and by French National Research Agency (ANR) within the WAVE Project, under grant number ANR-12-CORD-0027.

References

1. D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
2. A. E. Cano, G. Rizzo, A. Varga, M. Rowe, Stankovic Milan, and A.-S. Dadzie. Making Sense of Microposts (#Microposts2014) Named Entity Extraction & Linking Challenge. In *4^{th} International Workshop on Making Sense of Microposts*, Seoul, South Korea, 2014.

3. J. R. Finkel, T. Grenager, and C. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *43rd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 363–370, Stroudsburg, PA, USA, 2005.
4. J. Hoffart, Y. Altun, and G. Weikum. Discovering Emerging Entities with Ambiguous Names. In *23rd International Conference on World Wide Web (WWW)*, pages 385–396, Seoul, Korea, 2014.
5. J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust Disambiguation of Named Entities in Text. In *8th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 782–792, Stroudsburg, PA, USA, 2011.
6. H. Ji, J. Nothman, and B. Hachey. Overview of TAC-KBP2014 Entity Discovery and Linking Tasks. In *Text Analysis Conference (TAC)*, Gaithersburg, USA, 2014.
7. P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia Spotlight: Shedding Light on the Web of Documents. In *7th International Conference on Semantic Systems (I-Semantics)*, pages 1–8, 2011.
8. A. Moro, A. Raganato, and R. Navigli. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *TACL*, 2:231–244, 2014.
9. F. Piccinno and P. Ferragina. From TagME to WAT: a new entity annotator. In *1st ACM International Workshop on Entity Recognition & Disambiguation (ERD)*, pages 55–62, Gold Coast, Australia, 2014.
10. K. Toutanova and M. Christopher D. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 252–259, Edmond, Canada, 2003.
11. R. Usbeck, M. Röder, A.-C. Ngonga Ngomo, C. Baron, A. Both, M. Brümmer, D. Ceccarelli, M. Cornolti, D. Cherix, B. Eickmann, P. Ferragina, C. Lemke, A. Moro, R. Navigli, F. Piccinno, G. Rizzo, H. Sack, R. Speck, R. Troncy, J. Waitelonis, and L. Wesemann. GERBIL – General Entity Annotation Benchmark Framework. In *24th World Wide Web Conference (WWW)*, Florence, Italy, 2015.