Eaglet – a Named Entity Recognition and Entity Linking Gold Standard Checking Tool

Kunal Jha[1], Michael Röder[1], and Axel-Cyrille Ngonga Ngomo[1,2]

[1] AKSW Research Group
University of Leipzig
Augustusplatz 10, 04103 Leipzig, Germany
kunal.jha@uni-bonn.de,
roeder@informatik.uni-leipzig.de
[2] Data Science Group
University of Paderborn
Pohlweg 51, 33098 Paderborn, Germany
ngonga@upb.de

Abstract. The desideratum to bridge the unstructured and structured data on the web has lead to the advancement of a considerable number of annotation tools and the evaluation of these Named Entity Recognition and Entity Linking systems is incontrovertibly one of the primary tasks. However, these evaluations are mostly based on manually created gold standards. As much these gold standards have an upper hand of being created by a human, it also has room for major proportion of over-sightedness. We will demonstrate EAGLET[3], a tool that supports the semi-automatic checking of a gold standard based on a set of uniform annotation rules.

Keywords: Entity Recognition, Entity Linking, Benchmarks

1 Introduction

The number of information extraction systems has grown significantly over the past few years. In particular, NER (Named Entity Recognition) frameworks aim to locate named entities in natural language documents while Entity Linking (EL) applications link the recognised entities to a given knowledge base (KB). NER and EL tools are commonly evaluated using manually created gold standards (e.g., [4]), which are partly embedded in benchmarking frameworks (e.g., [7, 1]). While these gold standards have clearly spurred the development of ever better NER and EL systems, they have certain drawbacks [2]. The creation of a NER/EL gold standard is a difficult task because human annotators commonly have different interpretations of this task as shown by Ratinov et al. [6]. EAGLET provides a very generic, adaptable set of rules derived from existing benchmarks by Jha et al. [2] which allows us to check these gold standards in a semi-automatic way. EAGLET has been evaluated on 13 English gold standards and detected 38,453 errors. An evaluation of 10 tools on a subset of these datasets shows a performance difference of up to 10% micro F-measure on average [2].

[3] Available at https://github.com/AKSW/Eaglet

2 Architecture Overview

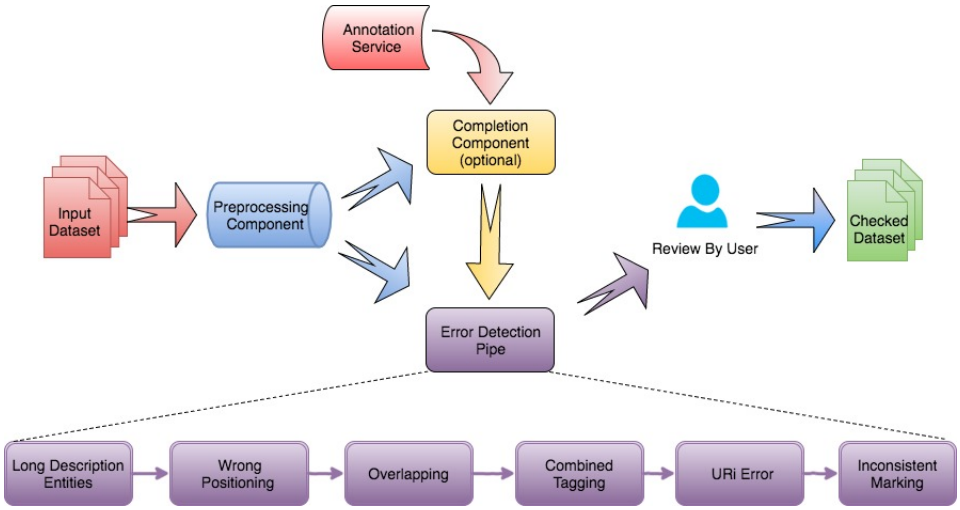

Fig. 1. EAGLET's Architecture. The modules marked in purple depict the pipeline structure of EAGLET.

EAGLET works based on the architecture presented in Figure 1. Following this architecture, we will attempt to explain the implementation. EAGLET provides a systematic classification of errors, which are in violation of the rules, with the ability to detect and correct a significant portion of these errors with minimum human interference. EAGLET categorises the violation of the rule set which are hard-coded into the system. The rest of this section discusses the implementation of each component shown in the architecture.

2.1 Input Dataset

The input of EAGLET is an annotated gold standard dataset which is primarily a set of documents where each document is an ordered set of words $d = \{w_1, ..., w_n\}$ along with the meta information of each annotation.

2.2 Preprocessing Component

The text of each document is passed through the Stanford NLP core library [3] to tokenise the text and lemmatise the tokens. After that, original text is coupled with the meta-info added by the NLP module. The annotation list is extended to the Corrected Annotation List (CAL) which contains some extra relevant EAGLET introduced meta-information as it is passed through the pipeline.

2.3 Error Detection Pipe

The processed dataset is passed into the pipe. Every module of the pipe represents a single error type[4] and is implemented independently from the other modules. This enables the possibility to adapt the pipe by deactivating or introducing modules. An adaptation is easy since (1) the tool is available as open source project and can be adapted by the user, (2) the modules share the same API and (3) the modules act independently.Annotations, identified as faulty by each module, are marked with an error type and a suggested solution. The flow of the pipeline is as follows.

1. *Long Description Detection Component*: The module checks for the sequences of words which may describe the entity they are linked to but do only contain an indirect description of the entity instead of directly naming it. The module identifies such a description by searching for a relative clause inside an annotation.
2. *Wrong Positioning Detection Component*: This module searches for faulty annotations that do not fit to the positions of the words, e.g., an annotation that does not start with the first character. The last character of an annotation is checked in a similar way, i.e., it should be the last character of a word.
3. *Overlapping Entity Detection Component*: The module checks for annotations involving the presence of two or more annotations that share a common sub-string.
4. *Combined Tagging Detection Component*: This module searches for consecutive annotations that are separated by a white space character. Such entities are marked and a larger, combined annotation is generated and added to the CAL. This module helps in tackling a non-trivial tier of errors wherein consecutive word sequences are marked as separate entities while the word sequences, if combined, can be annotated to a more specific entity. The tool later suggests the user to review the suggested marking of this new entity and assign the new entity a URI.
5. *URI Error Detection Component*: The module verifies the URIs of all entities regarding their format. If a URI points to a set of predefined KB the module tries to dereference the URI to check whether a) the entity exists and b) the URI does not point to a disambiguation page. For example, if the given KB is the Wikipedia[5] or entities can be directly mapped to Wikipedia entities the module uses the Wikipedia API to determine whether the URI is outdated and derives the new URI.
6. *Inconsistent Marking Component*: This module collects all annotations in the corpus that have not been marked as faulty by one of the other modules. The lemmatized surface form of every annotation is used to search for all non-marked occurrences of the entity throughout the dataset. Since these newly added annotations might be incorrect, e.g., because a URI that is linked to a word in one document does not need to fit to the same word in a different document, they are marked separately by the pipeline and should be checked by the user in the review module.

2.4 Completion Component

This component has been introduced as an optional module in EAGLET as the definition and requirement of the dataset completion may vary depending on the use case of the

[4] The detailed description of each error type can be found in EAGLET research paper [2].
[5] `http://wikipedia.org`

dataset. The component uses publicly available annotation services to derive a list of entity annotations on the dataset and compares it to the CAL list under consideration in the pipe, thereby generating a list of additional entities. These additional annotations support the work of a user that wants to make sure that the dataset is complete. However, since state-of-the-art annotation systems are not perfect [7], this module is based on a majority vote, i.e., the majority of the annotation systems have to contain an annotation inside their result list before it is added to the document. For this module, we relied on the open-source project GERBIL that enables the usage of up to 13 different annotation systems [5].

The components described above generated and stored a set of documents, different from the original input dataset, incorporating all the changes suggested in the pipe. This facilitates the reviewing which can be done by the user in different time duration and need not necessarily be completed in one run of EAGLET.

2.5 Review Component

This component involves at least one user reviewing all the corrections made by the pipe before they can be written to generate a final set of documents. We implemented a user interface which allows a user to check the results in an efficient, interactive way. The user interface (see Figures 2 and 3) of our tool allows every user to check each of the documents in the gold standards manually. Users can accept, modify or reject the suggestions of the tool as well as add new entities that have been missed by the auto-completion module. The user also has the liberty to change the URI of an entity. The review component has an auto completion module running in parallel which allows the addition made by a user in one of the documents to be automatically reflected in the other documents that have not been reviewed by the user so far reducing the redundancy in the reviewing process. The module is loaded with a login feature allowing multiple users to work on the same pipe results in order to generate a final, more accurate dataset. Each of these users can review and maintain their own versions of dataset at their own pace.

3 Conclusion

In this paper, we presented EAGLET—a tool to evaluate the gold standards used for NER and EL tasks. We described the different features of the tool which aim at correcting the existing the gold standard in order to achieve a more precise evaluation. We attempt to make this process user friendly and regard this work as a first stepping stone in a larger agenda pertaining to improving the assessment of the performance of natural language processing approaches.

Acknowledgments

This work has been supported by the H2020 project HOBBIT (GA no. 688227) as well as the the EuroStars projects DIESEL (project no. 01QE1512C) and QAMEL (project no. 01QE1549C).

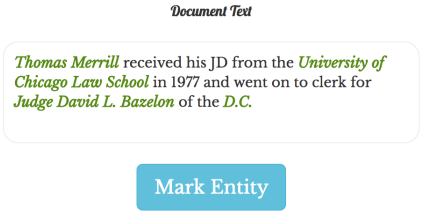

Fig. 2. Review Module: Document Text

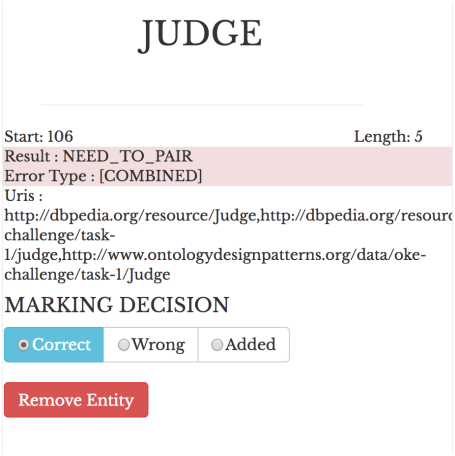

Fig. 3. Marking List

References

1. Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 249–260, New York, NY, USA, 2013. ACM.
2. Kunal Jha, Michael Röder, and Axel-Cyrille Ngonga Ngomo. All That Glitters is not Gold – Rule-Based Curation of Reference Datasets for Named Entity Recognition and Entity Linking. In *The Semantic Web. Latest Advances and New Domains: 14th International Conference, ESWC 2017, Proceedings*. Springer International Publishing, 2017.
3. Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.
4. Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8. ACM, 2011.
5. Röder Michael, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. Techreport for GERBIL 1.2.2 - V1. Technical report, Leipzig University, 2016.
6. 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. ACL, 2011.
7. 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.