Nordlys: A Toolkit for Entity-Oriented and Semantic Search

Faegheh Hasibi
Norwegian University of Science and Technology
faegheh.hasibi@ntnu.no

Krisztian Balog
University of Stavanger
krisztian.balog@uis.no

Darío Garigliotti
University of Stavanger
dario.garigliotti@uis.no

Shuo Zhang
University of Stavanger
shuo.zhang@uis.no

ABSTRACT

We introduce Nordlys, a toolkit for entity-oriented and semantic search. It provides functionality for entity cataloging, entity retrieval, entity linking, and target type identification. Nordlys may be used as a Python library or as a RESTful API, and also comes with a web-based user interface. The toolkit is open source and is available at http://nordlys.cc.

CCS CONCEPTS

•**Information systems** →**Information retrieval;** *Retrieval tasks and goals; Retrieval models and ranking;*

KEYWORDS

Semantic search; entity retrieval; entity linking; result presentation

ACM Reference format:
Faegheh Hasibi, Krisztian Balog, Darío Garigliotti, and Shuo Zhang. 2017. Nordlys: A Toolkit for Entity-Oriented and Semantic Search. In *Proceedings of SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan, ,* 4 pages.
DOI: http://dx.doi.org/10.1145/3077136.3084149

1 INTRODUCTION

Over the past decade, entities (such as people, organizations, or products) have become first-class citizens in web search, as well as in other domains. Entities are natural units for organizing information and a key enabling component for *semantic search*. Semantic search is not a single method, but rather an umbrella term that encompasses various techniques that aim to understand queries and answer them in a meaningful way. Building blocks of semantic search include, among others, entity retrieval [9, 16, 27, 32], entity linking [5, 7, 15, 19], query understanding [10, 14], and result presentation [6, 18].

There are two main reasons that motivated us to develop this toolkit. First, there exist a range of tools and demonstrators that address one specific task. Examples include GERBIL [30], TAGME [11], STICS [21], and Broccoli [4]. Second, repeatability and reproducibility of results is a fundamental requirement of scientific progress [1, 17, 23]. Having an open source and verifiable implementation of methods can foster research and development. In the area of entity-oriented search, despite recent advances, there is a lack of publicly available implementations of standard methods and techniques. With this work, we aim to fill that gap.

In this paper, we present Nordlys, which is a toolkit for entity-oriented and semantic search. One of its distinguishing features is that it implements a number of traditional and state-of-the-art methods for a range of tasks: entity retrieval, entity linking in queries, identifying target types of queries, and entity cataloging. Another important characteristic is that it accommodates various usage needs on different levels:

- It is made available as an open source Python library that can be integrated into larger applications or can be used as a command line tool for research and experimentation. The code is organized in a three-tier architecture, cleanly separating the various layers of functionality.
- It provides a RESTful API, through which Nordlys can be used as a service, much like a black box.
- The functionality is also available through a graphical web user interface. This interface can be used, e.g., to perform user studies on result presentation (similar to [6]).

In summary, Nordlys represents a major step towards reproducible and extensible semantic search research. It is meant to be a continuous effort, with additional methods included over time, as opposed to a one-time release of code. The toolkit is available at http://nordlys.cc.

2 FUNCTIONALITY

In this section we detail the functionality that is implemented in the Nordlys toolkit.

2.1 Entity Catalog

The main enabling component of semantic search, from a data perspective, is a *knowledge base*, which contains a collection of entities, their attributes, and relations to other entities. It also contains information about entity types, which are often organized in a hierarchical structure (called *type taxonomy*). The *entity catalog* serves as a data access layer to the knowledge base. It is used to obtain information about entities, such as their IDs, names, attributes, and relationships. Additionally, it provides basic statistics about entities and properties; these statistics may be utilized, among others, for result presentation (e.g., identifying prominent properties when generating entity cards) [20]. The entity catalog may be seen as

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/3077136.3084149

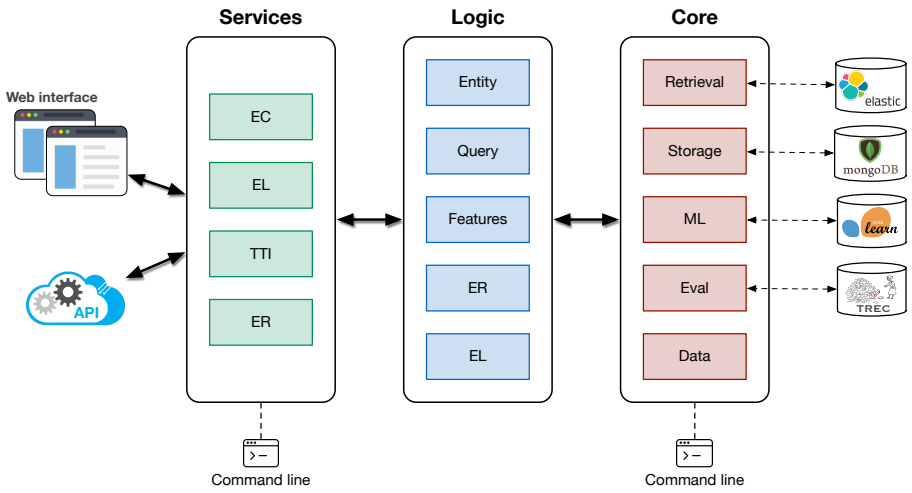

Figure 1: Nordlys architecture.

a data abstraction layer that makes the higher-level functionality largely independent of the particular data source. Nordlys is shipped with code and support for the DBpedia knowledge base, but data may be loaded into the entity catalog from any other knowledge base. Specifically, the entity catalog offers the following functionality:

- **Entity lookup.** Presents all entity properties from the knowledge base in the form of key-value pairs.
- **ID mapping.** Provides a one-to-one mapping of entity IDs across various knowledge bases via same-as links (DBpedia and Freebase).
- **Surface form lookup.** Presents all entities that match a given surface form ("alias"). A large surface form dictionary is obtained from the FACC collection, which contains Freebase annotations for the ClueWeb09 and ClueWeb12 datasets [12].
- **Basic statistics.** A set of statistics are computed over contents of the knowledge base, including RDF types, predicates, and properties.

2.2 Entity Retrieval

Entity retrieval is a core building block of semantic search, where a ranked list of entities are presented in response to an entity-bearing query. Formally, it is defined as follows.

Definition 1 (**Entity retrieval**): *Given a query q, entity retrieval is the task of returning a ranked list of entities $(e_1, ..., e_k)$ from an underlaying knowledge base KB, ordered according to their relevance to the query q.*

The toolkit implements both standard baselines and more recent approaches:

- **Elastic.** The BM25 model, as implemented in Elasticsearch, which is the most efficient retrieval model of our toolkit.
- **LM.** Language Modeling [29] approach (with Dirichlet prior and Jelinek-Mercer smoothing), which employs a single field representation of entities.
- **MLM.** The Mixture of Language Models [28], which represents entities as structured (fielded) documents, using a linear combination of language models built for each field. For DBpedia, we use a five-field representation [20].
- **PRMS.** The Probabilistic Model for Semistructured Data [22], which uses collection statistics to compute field weights in for MLM model (thereby making in parameter-free).
- **ELR.** The Entity Linking incorporated Retrieval model [16] is a state-of-the-art approach, which extends text-based retrieval models by considering entities mentioned in the query.

The online repository includes retrieval performance measurements on the DBpedia-entity test collection [3, 20], which is a standard benchmark for entity retrieval.

2.3 Entity Linking in Queries

Identifying named entities in queries and linking them to the corresponding entry in the knowledge base is known as the task of *entity linking in queries* (ELQ) [15]. Various demonstrations have been released for general purpose entity linking [8, 11, 26, 26, 30]. The most popular among them is TAGME [11], which has been designed for short texts such as tweets, search snippets. TAGME, however, does not handle the ambiguity of search queries and an entity mention is only linked to a single entity. The entity linking functionality in Nordlys aims to bridge this gap, by implementing both baselines and state-of-the-art approaches for entity linking in queries. The task is formally defined as follows.

Definition 2 (**Entity linking in queries**): *Given a query q, entity linking in queries returns one or multiple interpretations of the query, $A_1, ..., A_n$. Each interpretation consists of a set of entity linking decisions, i.e., mention-entity pairs: $A_i = \{(m_1, e_1), ..., (m_k, e_k)\}$, where mention m_j is a query substring that refers to entity e_j in the knowledge base.*

The following methods are implemented in Nordlys:

- **CMNS**. The baseline method that performs entity linking based on the overall popularity of entities as link targets, i.e., the *commonness* feature [24]. This method has been introduced in [25], and has been widely used in later research, e.g., [5, 15, 31].

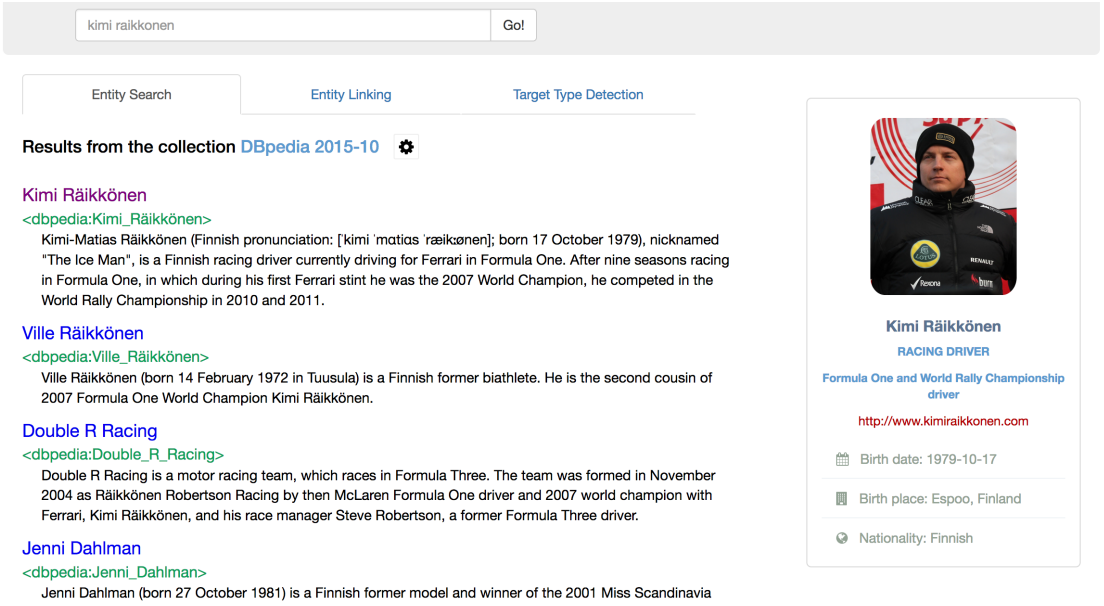

Figure 2: The Nordlys web user interface.

- **MLM-greedy**. An efficient generative retrieval model proposed in [15], that combines the commonness score with the textual similarity between the query and the entity [28].
- **LTR-greedy**. The recommended method (with respect to both efficiency and effectiveness) by Hasibi et al. [19], which employs a learning-to-rank model with various textual and semantic similarity features.

2.4 Target Type Identification

Target type detection is one specific form of query understanding, where the aim is to assign target types (or categories) to queries from some type taxonomy [2, 13]. Formally:

Definition 3 (**Target Type Identification**): *Given a query q, the aim of target type identification task is to return a ranked list of entity types $(t_1, ..., t_k)$, ordered with respect to their likelihood of being the target type of query q.*

We implement three diiferent approaches from the literature [2, 13]:

- **Entity-centric** This method first ranks entities based on their relevance to the query, then looks at what types the top-k ranked entities have. The final score for a given type t is the aggregation of the relevance scores of entities with that type [2].
- **Type-centric** This approach builds, for each type, a direct term-based representation (pseudo type description document), by aggregating descriptions of entities of that type. Then, those type representations can be ranked much like documents [2].
- **Learning-to-rank** A supervised learning approach, which considers a rich set of features such as distributional similarity, type label similarities, and taxonomy-driven features [13].

These methods has been evaluated using a publicly available test collection; the test collection and results are presented in [13].

3 THE NORDLYS TOOLKIT

In this section, we present the overall architecture and the various ways in which our toolkit may be used. We ship our code with a small sample taken from DBpedia, and include scripts for processing and indexing DBpedia and Freebase. We also make data-dumps available for download (entity catalog and search indices).

3.1 Architecture

Nordlys is based on a multitier architecture with three layers: *core* (data tier), *logic*, and *services* (presentation tier); see Figure 1.

Core. The *core* layer provides basic functionality, including retrieval (Elasticsearch), storage (MongoDB key-value store), machine learning (scikit-learn), and evaluation (trec_eval). In parentheses are the third-party tools we currently use. It is possible to connect additional external tools (or replace our default choices) by implementing standard interfaces of the respective core modules. Additionally, a separate data module is provided with functionality for loading and preprocessing standard data sets (DBpedia, Freebase, ClueWeb, etc.). The core layer represents a versatile general-purpose modern IR library, which may also be accessed using command line tools.

Logic. This layer contains the main "business logic," which is organized around five main modules: (1) *entity* provides access to the entity catalog (including knowledge bases and entity surface form dictionaries); (2) *query* provides the representation of search queries along with various preprocessing methods; (3) *features* is a collection of entity-related features, which may be used across different search tasks; (4) *entity retrieval* (ER) contains various entity ranking methods, cf. §2.2; (5) *entity linking* (EL) implements entity linking functionality, cf. §2.3. The logic layer may not be accessed directly.

http://api.nordlys.cc/er?key=xx&q=total+recall&model=lm


```
{"query": "total recall",
 "total_hits": 1000",
 "results": {
  "0": {
   "entity": "<dbpedia:Total_Recall_(1990_film)>",
   "score": -10.042525028471253
  },
  "1": {
   "entity": "<dbpedia:Total_Recall_(2012_film)>",
   "score": -10.295316626850521
  },
  ...
```


Figure 3: Example call to the Nordlys API entity retrieval service.

Services. The *services* layer provides end-user access to the toolkit's functionality, through the command line, RESTful API, and web interface. Four main services are available: entity retrieval (ER), entity linking (EL), target type identification (TTI), and entity catalog (EC); these we have already explained in §2.

3.2 Usage Modes

Nordlys may be used as a Python library, RESTful API, command line tool, or a web interface.

Python Library. We base our toolkit on Python 3.5+, specifically, on the Anaconda distribution. The code is complemented with an automatically generated documentation as well as an extensive how-to. Upon downloading our toolkit, installing the prerequisites, and loading the necessary data, it should be possible to run Nordlys on any machine.

Command line. The main functionality may be accessed through a set of command line tools. This can be useful, e.g., for processing larger amounts of data and eliminating the network overhead.

RESTful API. We provide a public API endpoint (subject to registering for an API key). Figure 3 shows an excerpt from the response received from the API for an entity retrieval request. For convenience, the parameters of the methods are set to reasonable defaults; these values may be changed when calling the services.

Web Interface. Figure 2 shows an excerpt from our web user interface. The implementation is based on Flask and Bootstrap. The default settings of the web interface are similar to that of the API endpoint, any may be changed on the interface.

4 CONCLUSIONS AND FUTURE WORK

We have introduced Nordlys, a toolkit that implements a range of methods for entity-oriented and semantic search. It is available as a Python library, as a command line tool, a RESTful API, and as a graphical web user interface. Nordlys is meant to be a continuous effort; we plan to include additional methods, e.g., entity summarization. We are also working on optimizations, to improve the efficiency of our methods.

REFERENCES

[1] Jaime Arguello, Matt Crane, Fernando Diaz, Jimmy Lin, and Andrew Trotman. 2016. Report on the SIGIR 2015 Workshop on Reproducibility, Inexplicability, and Generalizability of Results (RIGOR). *SIGIR Forum* 49, 2 (2016), 107–116.

[2] Krisztian Balog and Robert Neumayer. 2012. Hierarchical Target Type Identification for Entity-Oriented Queries. In *Proc. of CIKM '12.* 2391–2394.

[3] Krisztian Balog and Robert Neumayer. 2013. A Test Collection for Entity Search in DBpedia. In *Proc. of SIGIR '13.* 737–740.

[4] Hannah Bast, Florian Baurle, Bjorn Buchhold, and Elmar Haumann. 2014. Semantic Full-text Search with Broccoli. In *Proc. of SIGIR '14.* 1265–1266.

[5] Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and Space-Efficient Entity Linking in Queries. In *Proc. of WSDM '15.* 179–188.

[6] Horatiu Bota, Ke Zhou, and Joemon M Jose. 2016. Playing Your Cards Right: The Effect of Entity Cards on Search Behaviour and Workload. In *Proc. of CHIIR '16.* 131–140.

[7] David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-june Paul (Paul) Hsu, and Kuansan Wang. 2014. ERD' 14 : Entity Recognition and Disambiguation Challenge. *SIGIR Forum* 48 (2014), 63–77.

[8] Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2013. Dexter: An Open Source Framework for Entity Linking. In *Proc. of ESAIR '13.* 17–20.

[9] Jing Chen, Chenyan Xiong, and Jamie Callan. 2016. An Empirical Study of Learning to Rank for Entity Search. In *Proc. of SIGIR '16.* 737–740.

[10] W. Bruce Croft, Michael Bendersky, Hang Li, and Gu Xu. 2011. Query Representation and Understanding Workshop. *SIGIR Forum* 44, 2 (2011), 48–53.

[11] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proc. of CIKM '10.* 1625–1628.

[12] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1. (2013).

[13] Dario Garigliotti, Faegheh Hasibi, and Krisztian Balog. 2017. Target Type Identification for Entity-Bearing Queries. In *Proc. of SIGIR '17.*

[14] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named Entity Recognition in Query. In *Proc. of SIGIR.* 267–274.

[15] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2015. Entity Linking in Queries: Tasks and Evaluation. In *Proc. of ICTIR '15.* 171–180.

[16] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. Exploiting Entity Linking in Queries for Entity Retrieval. In *Proc. of ICTIR '16.* 171–180.

[17] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. On the Reproducibility of the TAGME Entity Linking System. In *Proc. of ECIR '16.* 436–449.

[18] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2017. Dynamic Factual Summaries for Entity Cards. In *Proc. of SIGIR '17.*

[19] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2017. Entity Linking in Queries: Efficiency vs. Effectiveness. In *Proc. of ECIR '17.* 40–53.

[20] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-Entity v2: A Test Collection for Entity Search. In *Proc. of SIGIR '17.*

[21] Johannes Hoffart, Dragan Milchevski, and Gerhard Weikum. 2014. STICS: Searching with Strings, Things, and Cats. In *Proc. of SIGIR '14.* 1247–1248.

[22] Jinyoung Kim, Xiaobing Xue, and W Bruce Croft. 2009. A Probabilistic Retrieval Model for Semistructured Data. In *Proc. of ECIR '09.* 228–239.

[23] Jimmy J. Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig MacDonald, and Sebastiano Vigna. 2016. Toward Reproducible Baselines: The Open-Source IR Reproducibility Challenge. In *Proc. of ECIR '16.* 408–420.

[24] Olena Medelyan, Ian H Witten, and David Milne. 2008. Topic Indexing with Wikipedia. In *Proc. of the Wikipedia and AI workshop at the AAAI-08 conference.*

[25] Edgar Meij, Wouter Weerkamp, and Maarten De Rijke. 2012. Adding Semantics to Microblog Posts. In *Proc. of WSDM '12.* 563.

[26] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proc. of I-Semantics '11.* 1–8.

[27] Fedor Nikolaev, Alexander Kotov, and Nikita Zhiltsov. 2016. Parameterized Fielded Term Dependence Models for Ad-hoc Entity Retrieval from Knowledge Graph. In *Proc. of SIGIR '16.* 435–444.

[28] Paul Ogilvie and Jamie Callan. 2003. Combining Document Representations for Known-Item Search. In *Proc. of SIGIR '03.* 143–150.

[29] Jay M Ponte and W Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *Proc. of SIGIR '98.* 275–281.

[30] Ricardo Usbeck, Michael Röder, and et al. 2015. GERBIL: General Entity Annotator Benchmarking Framework. In *Proc. of WWW '15.* 1133–1143.

[31] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2016. Bag-of-Entities Representation for Ranking. In *Proc. of ICTIR '16.* 181–184.

[32] Nikita Zhiltsov, Alexander Kotov, and Fedor Nikolaev. 2015. Fielded Sequential Dependence Model for Ad-Hoc Entity Retrieval in the Web of Data. In *Proc. of SIGIR '15.* 253–262.