Discovering Names in Linked Data Datasets

Bianca Pereira[1], João C. P. da Silva[2], and Adriana S. Vivacqua[1,2]

[1]Programa de Pós-Graduação em Informática,
[2] Departamento de Ciência da Computação
Instituto de Matemática, Universidade Federal do Rio de Janeiro, Brazil
bianca.pereira@ppgi.ufrj.br, {jcps,avivacqua}@dcc.ufrj.br

Abstract. The Named Entity Recognition Task is one of the most common steps used in natural language applications. Linked Data datasets have been presented as promising background knowledge for Named Entity Recognition algorithms due to the amount of data available and the high variety of knowledge domains they cover. However, the discovery of names in Linked Data datasets is still a costly task if we consider the amount of available datasets and the heterogeneity of vocabulary used to describe them. In this work, we evaluate the usage of `rdfs:label` as a property referring to entities' name and we describe a set of heuristics created to discover properties identifying names for named entities in Linked Data datasets.

Keywords: Named Entity, Named Entity Recognition, Linked Data

1 Introduction

Named Entity Recognition (NER) in natural language texts is one of the most common tasks in Natural Language Processing. Since the sixth Message Understanding Conference (MUC) with the emergence of the term "named entity" and the formalization of the NER task, the techniques for recognizing names in texts have greatly evolved. Additionally, better knowledge bases not only for recognition of names but also for its disambiguation have been developed.

A named entity (NE) is an entity that can be identified by a proper name [2]. Originally NEs were instances of person, organization or location classes and also dates and numeric values. Nowadays there are many other classes that identify NEs [3] [4].

Techniques for NER range from dictionary-based approaches to rule or machine learning ones [10]. Over time, different knowledge bases have been used as background knowledge for the NER task: from manually created lists to datasets using knowledge available on the Web [4]. Recently, with the emergence of databases in Linked Data format, Linked Data datasets have been presenting as promising sources for NEs.

The Linked Open Data cloud (LOD cloud) provides knowledge in diverse human knowledge domains, including not only the most common types of entities mentioned previously as NE types, but also entities in the field of music, video, biology, among many others.

Several recent studies and tools have appeared linking NE mentions in free text to Linked Data resources [11]. The first step in this task is the discovery of which classes identify NEs and which properties refer to their names in the LD dataset. Only after this step, that is usually performed manually, the comparison between a name of a resource from the dataset with the name mentioned in the text can be made.

The heterogeneity in metadata used to describe LD datasets is one of the difficulties in using datasets from LOD cloud for NER. As a consequence of this heterogeneity, the identification of names in a LD dataset is a hard task, which starts with the identification of properties that refer to names, a costly task in and of itself. Due to this, works using LD datasets for NER and entity linking still use only a limited number of datasets available on the LOD cloud.

Our goal in this paper is to propose heuristics that help to determine which properties contain names of NEs as their values, henceforth called PIN (Property that Identifies Names), in generic LD datasets. Our results can be used to enable current tools to work with different datasets without requiring a manual analysis to understand all the metadata used to describe resources in a LD dataset.

The rest of this paper is organized as follows: in section 2 we present related work on using Linked Data for NER. In section 3 we explain the NER task and which features of Linked Data datasets can be used to perform this task. Following that, we evaluate the feasibility of using the `rdfs:label` property as a sole source of names in section 4 and present our algorithms for PIN identification in section 5. In section 6 we evaluate our heuristics and present our conclusions in section 7.

2 Related Works

There is a large number of tools (mostly commercial) using LD datasets for NER and linking. Despite the large number of LD datasets available on the LOD cloud they work only with a small set of them.

DBpedia Spotlight [9] is a tool which its main goal is to recognize names from a text and link them to resources from DBPedia[1]. DBpedia Spotlight uses a set of possible names also called surface forms created from the `rdfs:label` property as well as written variations of names taken from Wikipedia links. It is highly optimized for DBPedia and achieves high precision.

The work of Hoffart et al. [6] performs the same task as DBPedia Spotlight but it uses YAGO [12] as its source for NEs and the `yago:means` property as a source for names.

Large KB Gazetter[1] is a plug-in for GATE Platform[2] that enables using a generic LD dataset as a dictionary. It aims to allow any SPARQL query to be used as a source for NE names.

All previous work require knowledge about every vocabulary used to describe the LD datasets in order to use them as a source for NEs. We propose to

[1] http://nmwiki.ontotext.com/lkb_gazetteer/
[2] http://gate.ac.uk

use heuristics to allow them to identify NEs and their names from generic LD datasets without requiring manual analysis.

3 Using Linked Data for Named Entity Recognition

NER algorithms which are dictionary-based require some effort to create the dictionary used as background knowledge. Instead of manually creating these dictionaries, websites such as Wikipedia have been used as external knowledge for NER[7]. These new knowledge bases require different algorithms to structure their knowledge and extract entity names.

The main advantage of using LD datasets for NER tasks is that data is already structured. Algorithms that use Wikipedia as a gazetteer require preprocessing to extract all possible names of entities contained in its various pages. In another hand, LD datasets allow the creation of SPARQL queries for data retrieval, making the whole process much simpler. Another feature of LD datasets is the description of data using vocabularies or ontologies. This description enables the determination of an entity's type (person, place, etc.) through a simple query.

LD datasets are structured using RDF [5] resources to describe entities from the real world. Each resource is described through properties and relationships with other resources. Both property and relationship are specified by vocabularies or ontologies that indicate to a human what each one of them means. Furthermore, RDF Schema (RDFS) [8] presents a set of properties commonly used to describe resources in LD datasets. In our work, the `rdfs:label` is a relevant property because it describes a human-readable name for RDF resources often a NE name.

A starting point in searching for NE names in a LD dataset would be to use the contents of the `rdfs:label` property as DBPedia Spotlight does. In the following section, we present an analysis of the usage of `rdfs:label` as a unique source for names in LD datasets.

4 Using `rdfs:label` as a name

The most intuitive approach for the identification of names from NEs in a LD dataset is using the `rdfs:label` property. To verify the applicability of this approach, we conducted an analysis of a small set of datasets from the LOD cloud. Our goal was to see if this approach was sufficient for the task of acquiring names for NEs in generic LD datasets.

The first step was to select LD datasets that contain resources describing NEs that explicitly specify their name using properties. We selected a set of domain-specific datasets: Linked Movie Database [5], Geo Linked Data[8], Linked Brainz[3] and Jamendo (DBTune)[4]. The first dataset contains data from films

[3] http://linkedbrainz.c4dmpresents.org/
[4] http://dbtune.org/jamendo/

with information such as actors, characters and performances. Geo Linked Data describes spatial data, such as places and points of interest. The last two datasets are about music but Jamendo focuses on indepent musical groups and singers.

In the Linked Movie Database the `rdfs:label` property is present in almost all classes of entities described by the dataset. Among the classes there are those representing NEs and those representing other types of entity. We noticed that these other entities are, in fact, relationships between more than two entities.

For this first dataset, if we always use the `rdfs:label` property as a source of names we would extract some incorrect names. Further, given that the `rdfs:label` property is used to provide a human-readable label, and not necessarily the name of the entity, the NEs present in the dataset usually had the entity class as part of the value of the `rdfs:label` property. For instance, the entity identified by the URI http://data.linkedmdb.org/resource/film_character/253 is of *Film Character* class and contains the text "Kate (Film Character)" in its `rdfs:label` property. On the other hand, there is a set of properties that identify the names of various NEs in the dataset: `actor_name`, `director_name`, `cinematographer_name`, `editor_name`, among others. In the example mentioned above, the name of the entity is represented by `film_character_name` property whose value is "Kate".

The second dataset is Geo Linked Data. This dataset consists of ten named graphs, where seven of them are datasets and the others contain some metadata. Among these seven, we excluded two, which referred to statistical indexes and one that referred to years, which is not our focus at this point. Of the four remaining datasets we could verify that all names from NEs are exclusively described by the `rdfs:label` property. In addition, this property does not appear in entities that are not NEs. Even though it is possible to extract all the names using only the `rdfs:label` property, a large part of the entities have values in a format not commonly used. For example, an entity of *Aeropuerto* (*Airport*) class has the string "Sevilla, Aeropuerto de" as the `rdfs:label` property value, rather than "Aeropuerto de Sevilla".

The third dataset selected was the Linked Brainz, a dataset created from information available on the MusicBrainz website. Linked Brainz describes entities in the music domain such as singers, music groups and their work. It has ten classes that represent NEs but only seven use `rdfs:label` to describe the name of the entities. All NEs, even those using `rdfs:label`, use other properties not only to describe the most common name of the entity, but also to describe alternative names. The properties used are: `skos:altLabel` and `skos:notation`, described by the SKOS vocabulary, `foaf:name` defined by the FOAF vocabulary, `dc:title` described by the Dublin Core vocabulary, `vo:sortLabel` from the OpenVocab[5] vocabulary, and another `geo:name` property described by the Basic Geo (WGS84 lat/long)[6] vocabulary . Given that not all NE classes use the `rdfs:label` property, using only this property would exclude useful information.

[5] http://open.vocab.org/docs
[6] http://www.w3.org/2003/01/geo/

The last dataset in our analysis was Jamendo, from the DBTune.org website. This dataset was generated from the information of the Jamendo website and contains information about independent music groups and artists, and their work. This dataset does not use `rdfs:label` to describe their entities. All names are described by two properties: `dc:title` from the Dublin Core vocabulary and `foaf:name` from FOAF vocabulary.

We could verify that a range of properties may contain the names for NEs in a LD dataset. As this list is not fixed because it depends on the vocabularies used by each dataset, it is not possible to create an algorithm that considers the full list of every possible property that identify names of NEs. Thus we need to be able to identify automatically which properties contain names of NEs for each dataset.

5 Discovering properties that refer to names

In this section, we present a set of heuristics to identify PIN (Property that Identifies Names) in LD datasets. Each algorithm receives a LD dataset as input and returns a set of PIN for each class in the dataset. If a specific class does not represent NEs, the algorithm must not return a PIN for this class, otherwise, it should return one or more PIN. Each heuristic was created based on the assumption that names are represented by proper names. To identify if a given string is a proper name we are considering that every string with at least 50% of its words capitalized is a proper name.

The same basic algorithm is used differing only in the heuristic (score function and requisites) used.

Each algorithm recovers every class in a LD dataset and every property p that has a literal as its value for each class c found. After that, for each class c and each property p used to describe instances of this class the algorithm calculates a score based on the occurrence of proper names as values of p. Each heuristic identifies as a PIN the best scored property according to their respective requisite. As our goal is identify PIN that differ from one dataset to another we will give priority for other properties than `rdfs:label`.

Four heuristics were developed, and are described in the following subsections: Naive, Parametrized Naive, Multivalue and Multivalue with Threshold. The Naive and Parametrized Naive heuristics consider only the best scored property for each class (return a single result) and the Multivalue and Multivalue with Threshold heuristics return every property that score higher than a given value.

5.1 Naive Heuristic

The Naive heuristic is the simplest and returns the property that has the highest occurrence (higher score) of proper names as its value for each class.

The score(p,c) function is given by the sum of each occurrence of a proper name as a value of the property p in entities from class c (e_c) :

$$score_n(p,c) = \sum_{e_c} \left\{ \begin{array}{l} 1,\ if\ p_{\,value} = proper\ name \\ 0,\ otherwise \end{array} \right\} \quad (1)$$

If every score for properties is equal to zero, class c has no PIN.

5.2 Parametrized Naive Heuristic

The Naive heuristic does not impose any kind of restriction for a proper name to be recognized as a name. Therefore, this heuristic can return not only names but also acronyms and possibly descriptions that have a high frequency of capitalized words. Acronyms are usually short strings about 2 to 4 characters in upper case while descriptions tend to be paragraphs or a set of paragraphs formed by a large number of characters.

The Parametrized Naive heuristic aims to avoid occurrence of description texts and acronyms as value for PIN. The heuristic uses two constraints *min* and *max* to restrict the length of the string accepted as a name.

The score function (Formula 2) only counts occurrences in which p value is a proper name with length greater than or equal to *min* and lower than or equal to *max*.

$$score_{pn}(p,c) = \sum_{e_c} \left\{ \begin{array}{l} 1,\ if\ p_{\,value} = proper\ name, length(p_{\,value}) \in [min, max] \\ 0,\ otherwise \end{array} \right\} \tag{2}$$

If the highest score is equal to 0 for a given class then there are not any PIN associated with it.

5.3 Multivalue Heuristic

Given that many entities can be reffered to by a set of names instead of a unique name, we propose a heuristic to identify these alternate names as well. We can identify the most used name as a preferred name, and other names as alternate names or acronyms referring to the same entity. The previous heuristics return only one property as a PIN while this heuristic retrieves every possible PIN including properties referring to acronyms. This heuristic is the same as the Parametrized Naïve when considering the min value equal to zero and adding a parameter with the number of returned PIN per class with a value equal to one.

In the Multivalue heuristic we intend to accept acronyms as valid values but not descriptions. In this way, *max* is also used in the score function (Formula 3).

$$score_{Multi}(p,c) = \sum_{e_c} \left\{ \begin{array}{l} 1,\ if\ p_{\,value} = proper\ name, length(p_{\,value}) \leq max \\ 0,\ otherwise \end{array} \right\} \tag{3}$$

The requisite to decide if a property p can be chosen as a PIN for a given class is if its score is higher than zero.

5.4 Multivalue with Threshold Heuristic

This last heuristic is characterized by recognizing more than one property as a PIN for each class and identifying only the best scored properties rather than every property with a score greater than zero.

The score for Multivalue with Threshold is calculated based on the relative frequency of occurrence of property p referring to names for entities in class c. In other words, a property p will be considered a PIN only if it also appears describing a percentage of entities higher than a given threshold for a given class. The score function can be seen in Formula 4.

$$score_{Threshold}(p, c) = score_{Multi}(p, c)/|e_c| \quad (4)$$

A given entity in a real world can have many alternative names. Due to this, the Linked Data resource representing this entity may use the same property many times to describe these diverse names. In order to give the same weight for each property in the dataset this heuristic only counts one occurrence of each property for each instance of a class.

The threshold will be used to select PIN. Every property with a score higher than the threshold value will be considered as a PIN for class c.

6 Experiments

In our evaluation process we used the same datasets aforementioned excepting Geo Linked Data and Linked Brainz. The Geo Linked Data was not used because it only uses `rdfs:label` as a PIN. The Linked Brainz has about two billion triples what requires a machine with a high processing power and memory available to enable a good processing time.

The evaluated datasets were: Jamendo (DBTune.org) and Linked Movie Database[5].

6.1 Gold Standard

To enable the evaluation of our heuristics we have created a gold standard. It was manually developed and consists of a list of classes that have NEs as its instances and a list of PIN associated to those classes. We assume at this point that if a class represents NEs then each one of its instances is a NE.

The steps to create the gold standard are as follows:

- Identify all classes describing resources in the dataset.
- For each class identify all properties whose value is a literal.
- Analyze the meaning of each class and property
- Select classes that define NEs as its instances
- Select a set of PIN for each class that define NEs.

For the first two steps we used SPARQL queries to list all classes and their respective properties. Having all classes and their respective properties we analyzed their meaning. In other words, we have searched for the ontology description and if it does not exist or it is inconclusive we manually analyzed few instances of each class. These ontology descriptions are mainly searched based on their namespace and the respective LD dataset's project website.

Based on the meaning of each class and property we could identify those that describe NEs. Each NE is identified by one or more names. We say that a class represents NEs if it has one or more PIN associated with its instances. We assumed that names are proper names and are infrequently shared by many instances from the same class.

Each PIN was identified as referring to a preferential name, an alternative name or an acronym. Preferential name is the most frequent name for a given NE. Properties associated with preferential names appears only once in a Linked Data resource that describes a NE but those associated with alternative names and acronyms may appear zero or more times. Acronyms are identified by having many capitalized characters in a single word while alternative names do not have this feature.

There are some dataset features that should be pointed out. In every dataset there are classes that do not refer to NEs and therefore do not have PIN. Analyzing the meaning of classes we notice that a class does not always describe NEs or even entities. Some examples are: class *Playlist* in Jamendo and class *Performance* in Linked Movie Database. The instances of *Performance* class are not NEs but ternary relationships involving actors, films and characters. These instances have properties responsible to link them with other entities but their values are string representations of related entities names and not URIs, as is recommended for relationships in Linked Data.

Another feature founded is that we can not use only the ontology description to identify if a class describes NEs because sometimes the ontology description is not available as in the case of Linked Movie Database.

We also have to make some observations about properties classified as containing preferential or alternative names. In Linked Movie Database there are some properties that share the same values such as `dc:title` and `rdfs:label` for the class *Film* then in this case both were identified as referring to preferential names.

6.2 Evaluation

The goal for the heuristics developed is, primarily, the identification of PIN associated with preferential names. If a heuristic identifies alternative names or acronyms we understand this as a correct answer however it is not the best answer. In the case of Multivalue and Multivalue with Threshold we intend to retrieve every PIN from the LD dataset. In any case we understand the identification of PIN for classes that do not identify NEs as an error.

Our experiments were processed in two steps. Each one evaluates all heuristics using a different dataset. For each dataset we made a local installation using the RDF files provided by CKAN website in order to provide results that do not change during our experiments.

Jamendo Jamendo is a small dataset with 11 classes being 3 of them describing NEs, each one containing 1 PIN.

The Naive Heuristic found all three properties correctly plus another one: `mo:text` from *Lyrics* class. This represents song letters which are reproduced as a value for property `mo:text`. The incorrect classification of this property is due to the fact that we are considering every string as possible names regardless whether a string has many or few letters.

In the Parametrized Naive Heuristic, we considered that the value of the candidate properties should have minimum length of 4 and maximum length of 100 characters. Although the number of occurrence for `mo:text` has dropped, the same classes and properties were obtained.

There was an increase of false positive candidate PIN with the Multivalue Heuristic because it discovered a new property for each of the classes *Record* and *MusicArtist* since it selects not only the best scored properties but any property of a class with non-zero score. In the Multivalue with threshold heuristic, we established that the maximum length of a string is 100, and use values 0.4, 0.6, 0.8 and 0.95 as threshold. With 0.4, 0.6 and 0.8 as threshold, the classes *Record* and *MusicArtist* and their properties were correctly identified. Increasing the threshold to 0.95, no property was identified. The overall results of application of each heuristic can be seen in Table 1.

Table 1. Results from PIN identification heuristics for Jamendo Dataset

Heuristic	**PIN found**	**False positives**
Naive	3 (100%)	1
Parametrized Naive	3 (100%)	1
Multivalue	3 (100%)	3
Multivalue with Threshold (0.4 , 0.6, 0.8)	2 (66.67%)	0
Multivalue with Threshold (0.95)	0 (0%)	0

Linked Movie Database The Linked Movie Database[5] has a large number of classes (53 classes being 34 describing NEs). It has its own ontology that, unfortunately, does not have description for its classes and properties available on CKAN or dataset website.

This dataset has a particular feature. There are some entities that belongs to two different classes. For example, the resource identified by the URI `http://data.linkedmdb.org/resource/actor/1` has two `rdf:type` values: *Actor* and *Person*. Due to this the *Person* class does not have a unique PIN associated with preferential names and each one of these PIN does not appear together.

The Naive heuristic identified 31 out of 41 PINs. There was a draw between the hits number for `rdfs:label` and the correct PIN in many classes but as `rdfs:label` has lower priority according to our algorithm the correct properties were identified as PIN. Only two PIN had more hits than `rdfs:label`: `movie:film_character_name` from *Character* class and `movie:film_company_name` from *Film Company* class.

There were also a large number of false positives due to two features of the dataset. The first is that the Linked Movie Database has classes we did not recognize as identifying NEs such as *Film Focus* or *Film Distribution Medium*. Regardless whether these classes do not identify NEs they have properties describing their names such as "Theatre" or "CD" for instances of *Film Distribution Medium*. The second feature is literals as values for properties referring to relationships. There are some classes such as *Performance* that identify ternary relations between instances of *Actor*, *Character* and *Film* but properties relating an instance of *Performance* with instances of the other classes have strings as their values instead of URIs. In this case, these relationships were returned as PIN due to a wrong description used in the dataset.

The Parametrized Naive heuristic identified the correct PIN (`movie:country_name`) instead of properties referring to acronyms(`movie:country_iso_alpha3`) due to parameters min = 4 and max = 100 as expected. This heuristic had more false positives due to the recognition of `rdfs:label` rather than the correct PIN for some classes. It happens because the `rdfs:label` value is composed by the value of the correct name plus the class name so some values for the correct PIN were discarded by the min parameter but the `rdfs:label` for the respective entity was not discarded because its value has more characters. If `rdfs:label` has at least one hit more than the correct PIN, this heuristic will recognized it as a PIN.

The Multivalue heuristic identified all PIN from the dataset. It could also identified every PIN for the *Person* class because it does not restrict the number of hits to recognize a property as a PIN. Although the 100% of recall this heuristic retrieved a large number of false positive. This heuristic had the same problem identified in the Naive Heuristic but the previous get only one property as a PIN.

At last, the Multivalue with Threshold Heuristic did not identified any PIN for the *Person* class as we expected. It still have a high number of false positives due to features aforementioned. The overall results can be seen in Table 2.

Table 2. Results of PIN identification heuristics for Linked Movie Database

Heuristic	**PIN found**	**False positives**
Naive	31 (75.61%)	21
Parametrized Naive	19 (46.34%)	33
Multivalue	41 (100%)	85
Multivalue with Threshold (0.4)	35 (85.36%)	68
Multivalue with Threshold (0.6)	35 (85.36%)	65
Multivalue with Threshold (0.8)	34 (82.93%)	61
Multivalue with Threshold (0.95)	32 (78.05%)	54

6.3 Analysis

Each heuristic has different characteristics and the best fit will depend on the dataset features.

The Naive and Parametrized Naive Heuristics do not prioritize recall. They return only one PIN for each NE class and these PIN refer only to preferable names for NEs. Thus they presented a better precision because they usually identify the right PIN but they do not return every possible PIN from the dataset. In applications that need to recognize names in a LD dataset without many errors these two heuristics are preferable. Moreover, the Multivalue and Multivalue with Threshold have a better recall. The Multivalue Heuristic returns every possible PIN from the dataset recognizing every PIN in our experiments but also returning a high number of false positives. The Multivalue with Threshold Heuristic allows mantaining the recall but with more precision. As we increase the value of the Threshold we have less false positives with a good recall. These last two heuristics are preferable in applications that only need PIN to reduce the search space for names. The results for each heuristic can be seen in Table 3.

In addition our heuristics could also identify which classes have NE as their instances. Each class that have at least one PIN recognized can be seen as NE class.

Despite the high number of false positives, our heuristics have obtained a reasonable result in this preliminary study. The next step is evaluating our heuristics using a bigger set of LD datasets in order to acquire more insights about common Linked Data features and how the heuristics perform with new features. The heuristics may also be important to provide an overview of which properties are actually used to describe names for NEs in LD datasets and to reduce the search space for names of NEs described in the dataset. Therefore, they help using generic datasets in LD as knowledge bases for NER and linking tasks.

	Jamendo			Linked Movie Database		
Heuristics	Precision	Recall	F-Score	Precision	Recall	F-Score
Naive	0.75	1	0.8571	0.5962	0.7561	0.6667
Parametrized Naive	0.75	1	0.8571	0.3654	0.4634	0.4085
Multivalue	0.5	1	0.6667	0.3254	1	0.4910
Multivalue (Threshold = 0.4)	1	1	1	0.3398	0.8536	0.4861
Multivalue (Threshold = 0.6)	1	1	1	0.35	0.8536	0.4964
Multivalue (Threshold = 0.8)	1	1	1	0.3579	0.8293	0.5
Multivalue (Threshold = 0.95)	0	0	0	0.3721	0.7805	0.5039

Table 3. Overall Results for the application of every heuristic for PIN identification

7 Conclusion

In this paper we started to address the problem of finding names for NE in generic Linked Data datasets. Due to the heterogeneity in the description of

these datasets, the identification of properties that have names as their values is not trivial. We analyzed the feasibility of using `rdfs:label` as a unique source for NE names and then presented a set of heuristics for identification of PIN, those properties whose values may be names for NEs.

We conducted a preliminary study using our heuristics with two datasets from the LOD cloud. Both datasets have a significant number of triples, classes and properties. We created a gold standard to evaluate our heuristics. Based on the results of the evaluation, we discovered that our heuristics can be used to identify PIN for these LD datasets, but given that the heuristics' accuracy were not 100%, we suggest that they undergo a process of manual review before they are used in applications that require 100% accuracy.

8 Acknowledgments

This work was supported by a grant from CAPES, Brazil.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. The Semantic Web pp. 722–735 (2007)
2. Chinchor, N.: Overview of muc-7/met-2 (1998)
3. Cohen, W., Sarawagi, S.: Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In: Proceedings of the tenth ACM SIGKDD. pp. 89–98. ACM (2004)
4. Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., Yates, A.: Unsupervised named-entity extraction from the web: An experimental study. Artificial Intelligence 165(1), 91–134 (2005)
5. Hassanzadeh, O., Consens, M.: Linked movie data base. In: Proceedings of the 2nd Workshop on Linked Data on the Web (LDOW2009) (2009)
6. Hoffart, J., Yosef, M., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., Weikum, G.: Robust disambiguation of named entities in text pp. 782–792 (2011)
7. Kazama, J., Torisawa, K.: Exploiting wikipedia as external knowledge for named entity recognition. In: Proceedings of the EMNLP-CoNLL 2007. pp. 698–707 (2007)
8. Lopez-Pellicer, F., Silva, M., Chaves, M., Javier Zarazaga-Soria, F., Muro-Medrano, P.: Geo linked data. In: Database and Expert Systems Applications. pp. 495–502. Springer (2010)
9. Mendes, P., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems. pp. 1–8. ACM (2011)
10. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes 30(1), 3–26 (2007)
11. Rizzo, G., Troncy, R.: Nerd: A framework for unifying named entity recognition and disambiguation extraction tools. EACL 2012 p. 73 (2012)
12. Suchanek, F., Kasneci, G., Weikum, G.: Yago: A large ontology from wikipedia and wordnet. Web Semantics: Science, Services and Agents on the World Wide Web 6(3), 203–217 (2008)