Enhancing the Open-Domain Classification of Named Entity Using Linked Open Data

Yuan Ni, Lei Zhang, Zhaoming Qiu, and Chen Wang

IBM Research, China
{niyuan,lzhangl,qiuzhaom,chenwang}@cn.ibm.com

Abstract. Many applications make use of named entity classification. Machine learning is the preferred technique adopted for many named entity classification methods where the choice of features is critical to final performance. Existing approaches explore only the features derived from the characteristic of the named entity itself or its linguistic context. With the development of the Semantic Web, a large number of data sources are published and connected across the Web as Linked Open Data (LOD). LOD provides rich a priori knowledge about entity type information, knowledge that can be a valuable asset when used in connection with named entity classification. In this paper, we explore the use of LOD to enhance named entity classification. Our method extracts information from LOD and builds a type knowledge base which is used to score a (named entity string, type) pair. This score is then injected as one or more features into the existing classifier in order to improve its performance. We conducted a thorough experimental study and report the results, which confirm the effectiveness of our proposed method.

Keywords: Named Entity Classification, Linked Open Data.

1 Introduction

Automatically classifying named entities from text is a crucial step in several applications. For example, in the Question Answering system [17], one important step is to determine whether a candidate answer is of the correct type given the question, and in the Information Extraction system [9], the first step is to identify the named entities and their types from the text. As devices linked to the Internet proliferate, the amount of information available on the Web rapidly grows. At the same time, there is a trend to scaling applications to the Web. For example, the MULDER Question Answering system [19] is designed to answer open-domain questions from the Web, and the TextRunner system is designed to extract open information from the Web [6]. To satisfy the requirements of such Web-scale applications, named entity classification is evolving from a simple three-category classification system to one in which a large number of classes are specified by an ontology or by a taxonomy [8], clearly a more challenging task.

P.F. Patel-Schneider et al. (Eds.): ISWC 2010, Part I, LNCS 6496, pp. 566–581, 2010.

Most proposed approaches for named entity classification adopt machine-learning techniques. The choice of features is critical to obtaining a good classification of named entities. Traditional classification approaches focus on the word-level characters or the context information of the named entities [20] without exploiting in any way the valuable classification information available for the large number of entities provided by Linked Open Data (LOD) on the Web. As of this writing, the Linked Data project [1] includes more than 100 datasets which together contain about 4.2 billion triples, and the datasets cover such various domains as Wikipedia, IMDb, and Geonames. Given a named entity, it is highly likely that some type assertions can be found in LOD. Thus, knowledge from LOD can provide good features for named entity classification. Features used in existing methods can be considered as context-dependent linguistic features, and the machine-learning technique makes use of statistic information based on these features to obtain a posteriori knowledge, while the features from LOD are considered as context-independent features that explore a priori knowledge. Our proposed method is to integrate the a priori knowledge with the a posteriori knowledge to improve named entity classification.

In this work we examine how to make use of the type information from LOD to improve named entity classification. It is not a trivial task due to the following challenges. First, because LOD may contain noisy information and it may be incomplete with respect to the information required to determine named entities and their potential types, we need to improve the precision and completeness of the type information. Second, for the extracted and cleaned type information for named entities, we need a mechanism to store it and to guarantee the efficiency of the method used to process the large, unwieldy LOD for use in real applications. Third, LOD from various sources have various taxonomies, and named entity classification also has its own taxonomies; therefore, we need a mechanism to solve the type similarity problem among these multiple taxonomies. The final challenge is to provide a scoring strategy given the multiple possible type information for a named entity.

Organization. The remainder of this paper is organized as follows. Section 2 gives an overview of our proposed method; Section 3 presents the method we use to prepare the linked data and store the potential type information; the scoring method is discussed in Section 4; Section 5 presents our experimental evaluation of the feature; related work is discussed in Section 6, and the conclusion and future work are discussed in Section 7.

2 Overview

The key idea of our approach is to leverage the knowledge base from the LOD to generate a score to measure the probability that the given named entity could be classified as the target type. The scores can then be used by existing machine-learning methods as additional features to improve named entity classification. Let us use $< f_1, \ldots, f_n >$ to denote the feature vector for a named entity in the existing approach. Our method appends a set of features

derived from LOD to the end of that vector to obtain a new feature vector $< f_1, \ldots, f_n, F_1, \ldots, F_m >$. The new generated feature vectors are used by the classifiers for named entity classification. The rationale behind this method is to combine context-independent, a priori knowledge from LOD with linguistic contextual information in a single feature vector and let the machine-learning algorithm determine how to best combine them statistically.

Fig 1 shows the architecture of the component to compute the LOD feature. In order to efficiently compute the additional feature, our system is divided into online and offline components. For the offline components, the *LOD Preprocessor* is used to extract the type information from the various LOD sources and precompute the type knowledge base in a format that best suits our algorithms. For the online components, the *Type Retrieval* is used to retrieve all type information of a given named entity string from the type knowledge base. The LOD scorer takes charge of computing the probability scores for the retrieved types of the named entity and the target type. An intermediate taxonomy is used to calculate the similarity between each type that is provided by LOD and the target type. Finally, the obtained score, together with other feature scores, are given to the classifier.

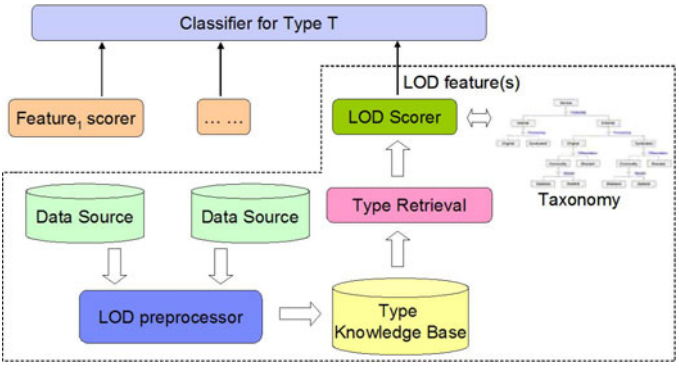


Fig. 1. The Architecture of the Component for Computing LOD feature

3 Type Knowledge Base Generation

The type knowledge base is needed for two reasons. First, the LOD may contain noise and may be incomplete; thus, we need to remove the noisy information and make the type information more complete. Second, to ensure the efficiency of the online scoring, the type information should be pre-computed and indexed in a format that supports fast retrieval. This section describes the method used to generate the type knowledge base. The goal is to enumerate all possible (name string, type) pairs from the LOD.

LOD uses a Uniform Resource Identifier (URI) to identify each instance, and type assertions are provided for each URI. For example, for an instance having the URI dbpedia:John_F._Kennedy[1], one triple from DBpedia indicates that the instance dbpedia:John_F._Kennedy belongs to type *President*:

(dbpedia:John_F._Kennedy, rdf:type, President).

For the instance dbpedia:John_F._Kennedy, we have another triple from DBpedia to indicate one of its names:

(dbpedia:John_F._Kennedy, rdfs:label, "John F. Kennedy").

Based on these two triples, we obtain the (name string, type) pair (John F. Kennedy, President) to specify that the name string "John F. Kennedy" belongs to category "President". From DBpedia, we have another triple which indicates another possible name for the instance dbpedia:John_F._Kennedy:

(dbpedia:John_F._Kennedy, dbpedia:birthName, "John Fitzgerald Kennedy").

From this we obtain another (name string, type) pair, (John Fitzgerald Kennedy, President).

As the type knowledge base requires the type information for each named entity, one problem we needed to solve is how to generate all possible name variants for each instance. Traditional approaches leverage the string similarity to determine whether some name variants correspond to one instance. However, the name variants of one instance may not always be similar. For example, ping-pong and table-tennis are the name variants for the same instance. In our method, we propose to use various name properties and certain relationships in LOD to enumerate all possible names variants for an instance.

3.1 Leverage the Name Properties

Thanks to the broad coverage of LOD, most name variants for an instance that may be mentioned in some text are likely to be specified by some name properties. We analyze the properties used in LOD sources and identify the ones that may describe the name information. For example, in terms of the definition from RDF schema, the property rdfs:label provides a human-readable description of a resource, making it a good candidate for name properties. We observe that there are many name properties in LOD, e.g., DBpedia has 106 properties about such names as dbpedia:name and dbpedia:fullname. To get maximal coverage on all possible names of an entity, we tried to use most of these properties. However, experimental results showed that they lead to many errors due to noisy LOD. For example, from DBpedia we have the following triples:

(dbpedia:Chrysler_New_Yorker, dbpedia:name, 1982)
(dbpedia:Chrysler_New_Yorker, rdf:type, Automobile)

We can then obtain the pair (1982, Automobile) which is not correct. Based on these experiments, we make use of only the properties that exactly describe the names, such as rdfs:label and foaf:name.

[1] The dbpedia: stands for the prefix for the URI from DBpedia.

3.2 Leverage the Relationships

In LOD, some relationships may connect two data instances where one data instance could be considered as providing another name variant of the other instance. We have identified three relationships and make use of them to enrich the name variants of the instances.

Redirects Relationship. The *redirects* relationship is used in DBpedia and links one URI, which has no description, to another URI that has a description. The purpose of creating and maintaining the *redirects* relationship is because the former URI has the relationships, including alternative name, less- and more-specific forms of names, abbreviations, etc. [3] with the later URI. If URI_1 redirects to URI_2, then the name of URI_1 can be considered as a name variant of the instance of URI_2. Therefore, for each (name string, type) pair, i.e., (name_2, type), that is derived from the type assertions for URI_2, we generate another pair (name_1, type).

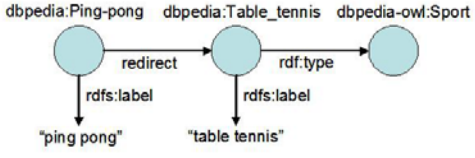


Fig. 2. The Example RDF Data for Redirect Relationship

Example 3.1. Let us use an example to illustrate this. Suppose we have a set of RDF triples, and their graph representation is shown in Fig 2. There is a redirect relationship from the URI dbpedia:Ping-pong to dbpedia:Table_tennis, meaning that ping-pong is a name variant of dbpedia:Table_tennis. From the description of dbpedia:Table_tennis, we obtain a pair (table tennis, Sport), and we then generate another pair (ping-pong, Sport) due to the redirects relationship. □

owl:sameAs Relationship. According to the OWL specification, the owl:sameAs indicates that two URI references actually refer to the same thing. Therefore, if URI_1 owl:sameAs URI_2, we combine them as one instance. The instance has the name variants from both URI_1 and URI_2 and has the types from both URI_1 and URI_2.

Disambiguates Relationship. The *disambiguates* relationship is used in DBpedia. A disambiguation URI has many *disambiguates* relationships with other different URIs which could, in principle, have the same name as that of the disambiguation URI. For example, we have (dbpedia:Joker disambiguates dbpedia:Joker_butterfly) and (dbpedia:Joker disambiguates dbpedia:The_ Joker's_ Wild). It means that dbpedia:Joker_butterfly and dbpedia:The_Joker's_Wild can have the same name Joker. Joker is then a name variant of dbpedia:Joker_butterfly and is also a name variant of dbpedia:The_Joker's_Wild. For all type assertions

about dbpedia:Joker_butterfly and dbpedia:The_Joker's_Wild, we generate corresponding (name string, type) pairs for the name Joker. For example, we know that dbpedia:Joker_butterfly belongs to the type *Insect*, then we generate a pair (Joker, Insect).

The enrichment using these relationships may not be 100% reliable because there may exist incorrect relationships due to the current quality level of LOD. However, we have conducted experiments that verify that the above enrichment helps improve the scoring accuracy.

3.3 Structure of the Type Knowledge Base

Given the (name string, type) pairs extracted from LOD, we need a mechanism to store and index them in order to guarantee the efficient retrieval for online scoring. The inverted list is used to store such information where the name string is the key. Different data instances may have the same name, but the scoring mechanism needs to distinguish the types for different instances (which will be introduced in detail in Section 4). Thus, the type information for a single name string is separated in terms of the data instance, i.e., the URI to which it corresponds. Fig. 3 shows the structure of the inverted list for storing the type information for name strings where the element $(t^1_{j1}, \ldots, t^1_{jm(j)})$ for entry N_1 stores all possible types of the j^{th} instance S^1_j that has the name string N_1.

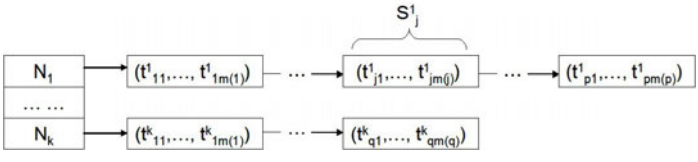


Fig. 3. The Inverted List for Type Knowledge Base

We could have used services like Sindice and Sigma[2] instead of building the inverted index on our own, but considering the complicated preprocessing we need to perform and the networking latency to use these services, we decided to build our own indexes.

In our work, we have generated a type knowledge base that includes LOD from DBpedia, IMDb, and GeoNames. The statistical information is shown in Table 1.

4 Scoring Method

Given a named entity string and a target type, the scoring is performed by computing a metric that measures the probability that the named entity can be classified as the target type using the type information from the type knowledge

[2] Sindice : `http://sindice.com`, Sigma:`http://sig.ma`

Table 1. Statistic Information for the Type Knowledge Base

Dataset	#instances (millions)	#name properties	#name variants (millions)	#type assertions (millions)
DBpeida	3.2	4	4.7	7.9
IMDb	24.5	8	12.0	24.4
GeoNames	6.7	3	8.4	6.5

base. There are three main challenges to doing this: (1) given a named entity string, how to find the matched names in the type knowledge base and get all possible type information for them; (2) because the types from LOD and the target type may be from different taxonomies, we need a strategy to precisely compute the similarity between these types; (3) because one named entity may correspond to multiple instances with multiple types, we need a mechanism to determine a final score. In the following sections, we introduce the details of the techniques used to meet these challenges.

4.1 Retrieving Types for the Named Entity String from Type Knowledge Base

To retrieve the possible type information, one simple method is to use the given named entity string as the key to find the corresponding types from the inverted lists. However, for the same entity, the given name may not be exactly the same as the names indexed in the type knowledge base. There are two reasons that can cause a name mismatch.

The entity itself has various names. For example, for President John F. Kennedy, one may use the full name John Fitzgerald Kennedy. As mentioned in Section 3, during the generation of the type knowledge base, we make use of the properties of names of instances to generate all possible names of an instance. Additionally, we make use of three types of relationships in LOD to enrich the possible names. With the help of all name properties of an instance and the relationships, our type knowledge base is likely to have a broad coverage of all possible names of an instance. Then, given a named entity string, a simple index lookup is enough to find its type information.

The names are presented in different format. For example, the indexed name is *tomato* while the given name is *tomatoes*. To solve this problem, we conduct the normalization on both the indexed names and the given names using the following rules: (1) perform word stemming on the names; (2) convert the names to lowercase; (3) remove any articles from names.

4.2 Matching the Target Type with the Retrieved Type

The types provided by LOD are considered as from open-domain. According to data publishers' requirements, new types can be added to describe new instances. Meanwhile, the types generated by data publishers are more flexible; for instance,

a type can be represented by a phrase, such as the category *jewish american film directors* from DBpedia. Even more, each data source in LOD may have its own type system. On the other side, the target type would also be considered as from open-domain because of the requirement of scaling information extraction and question answering to the Web. It is very difficult to match various types from open-domain taxonomies. We propose an intermediate ontology (denoted as **O**) to compute the similarity. First, the target type and the retrieved type are linguistically matched to some nodes in **O**, and we then compute the semantic distance between the two matched nodes in **O** and use this distance measurement as the similarity score.

Intermediate Ontology. One simple method for the intermediate ontology is to leverage an existing, well-defined general taxonomy. WordNet [11] is a well-defined general taxonomy widely used by the natural-language processing community. However, WordNet lacks a formal constraint on classes; for example, WordNet does not provide information about disjoint classes which could help us determine that a named entity does not belong to a type. Additionally, WordNet contains word senses that are too diverse, and a very rarely used word sense may incur a negative effect on the similarity computation. The AI community has also built general-purpose ontologies, such as Cyc [2], with formal logic theories. However, the ontology is very complex and lacks linguistic features. Considering the drawbacks of existing taxonomies, in our work we built a simple intermediate ontology. The ontology is designed in terms of the following principles: (1) the ontology covers the most frequently used concepts; (2) the ontology captures the disjoint axioms between classes such that we can obtain a score to measure how the named entity does not conform to the target type. For example, *people* and *organization* are disjointed classes. Our ontology is relatively small so if some type cannot be matched, we revert back to using WordNet.

Calculating the Similarity Score. The created ontology **O** is used as an intermediate ontology to link the target type and the retrieved type. The first step is to find the corresponding types of the target type and the retrieved type in **O**. If T denotes the target type/retrieved type, and the corresponding type in **O** is denoted as T', then T' should stand for the same concept as T. If a node in **O** exactly matches the type T, then the node is considered the best match for T. However, because of the flexibility of types from the open-domain, especially types from linked data, some types can be a phrase with adjective qualifiers that are not covered in **O**. To match these types for which no exactly matched nodes exist in **O**, we perform a normalization on the type phrase to get the headword. By analyzing type phrases from the linked data, we observed that the qualifiers are mainly presented in three ways: (1) an adjective is used to qualify a noun, for example, "Australian anthropologists"; (2) a qualifier phrase beginning with *of xxx* is used, such as "people of the French and Indian war"; (3) a qualifier phrase with *from xxx* is used, such as "people from St. Louis, Missouri". Given the type T, we remove the qualifiers for the above three cases to get the headword of

T, denoted as T_{root}. Finally, the node in **O** that matches T_{root} exactly is the corresponding type of T in **O**.

Suppose, T'_{target} and $T'_{retrieve}$ are the corresponding types of the target type and the retrieved type, respectively. The similarity score (denoted as s) is computed as follows:

- if T'_{target} and $T'_{retrieve}$ are the same node, then $s = 1$.
- if T'_{target} and $T'_{retrieve}$ are disjointed in terms of the ontology, then $s = -1$.
- if T'_{target} and $T'_{retrieve}$ are on the same path and there exists n steps between them, then $s = 1/n$.
- if T'_{target} and $T'_{retrieve}$ are in different paths and n_1 and n_2 are the number of steps to their lowest common ancestors, then $s = 1/(n_1 + n_2)$.

It is possible that T_{head} cannot be exactly matched by some node in the ontology **O** because our created ontology cannot cover all possible kinds of types. For these cases, we make use of the online resource WordNet to calculate the similarity score. Given a word, WordNet provides an API to get matched nodes. We then use the method proposed in [5] to calculate the semantic similarity between nodes in WordNet.

4.3 Determining the Final Score

Given a named entity $N(i)$, it may correspond to multiple data instances S_1^i, ..., S_j^i, ...,$S_n^i(i)$, and for each data instance S_j^i, it may belong to multiple types T_{j1}^i, ..., T_{jk}^i, ..., $T_{jm(j)}^i$. For each type T_{jk}^i, we calculate a score with respect to the target type T using the mechanism discussed in the previous section. For the named entity string $N(i)$, we then obtain multiple scores, which are divided into subsets according to the instances they describe. In this paper, we propose a two-step strategy to determine the final score. The first step is to compute the score for each data instance S_i given the scores for $\{T_{j1}^i, \ldots, T_{jk}^i, \ldots, T_{jm(j)}^i\}$; after that, we compute the score for the named entity $N(i)$ given the scores for $\{S_1^i, \ldots, S_j^i, \ldots, S_n^i(i)\}$. The advantage of the two-step strategy is as follows. By considering the characteristic of a single instance, we can avoid some noisy scores, making the score for a single instance more precise. The precise score for each instance is indispensable to obtaining a precise final score.

Determining the score for an instance S_i. Given a certain instance S_i, it can be stated that there should not be any conflicts within all of the type information for that instance. Therefore, for all scores of an instance, it is unlikely that some are positive (i.e., conform to the target type to some degree) and some are negative (i.e., conflict with the target type to some degree). However, due to the fuzzy match in the type-matching step and possible noise in the linked data, conflicts may occur. We propose the use of a vote strategy to solve this problem. For an instance S_i, if most of its types get positive scores, then the largest score is picked as the score for S_i; otherwise, if most of its types get negative scores, then the smallest score is picked as the score for S_i.

Determining the score for $N(i)$. Given multiple data instances for a named entity, if we know its URI, then the score for the data instance with the matched URI is used as the final score. This situation may occur in some Web-based question-answering applications in which Wikipedia is used as the corpora [4]. When the title of the Wikipedia page is selected as a candidate answer, the URI of the page is considered as the URI of the named entity and corresponds directly to a DBpedia URI.

In cases where there is no URI for the named entity and we cannot know which instance is indicated for this named entity, then we can use the following strategies to determine the final score. (1) *The aggressive strategy*: Considering that the named entity could be any one of the indexed data instances, one aggressive heuristic is that if the maximum score is larger than 0, then we pick the largest score; else if the maximum score is smaller than 0, then we pick the smallest score; otherwise the final score is 0. The aggressive strategy tends to give an exact score, either exactly matched or exactly unmatched. The score will be distinguishable. (2) *The average strategy*: We assume that the named entity has the same probability to match each indexed data instance, then the average of the total scores is used as the final score. An experimental study is provided in Section 5 to compare the above two strategies.

4.4 Applying the Score in Machine Learning

This section introduces how to use the score (which measures the probability of whether the named entity belongs to the target type) as a feature in the machine-learning step.

We can simply add one feature, which is called TyCorLOD (Type Coercion using LOD) in machine learning, and the generated score is used as the feature score. The score range for the TyCorLOD feature would then be [-1,1]. The higher score indicates that the named entity is more likely to belong to the target type. However, the problem with this method is that we cannot give different weights on the positive effect and negative effect. Therefore, we developed another method. In the machine-learning step, we split the score into two features (we call them TyCorLOD and AnTyCorLOD). The TyCorLOD feature indicates the likelihood that the named entity **conforms** to the target type, while the AnTyCorLOD feature indicates the likelihood that the named entity **conflicts** with the target type. These two feature scores are generated in terms of the final score S using the following strategy: If $S >= 0$, then we give the score S for TyCorLOD and 0 for AnTyCorLOD; otherwise (i.e., $S < 0$), we give the score 0 for TyCorLOD and $|S|$ for AnTyCorLOD. The comparison of the above two options is discussed in Section 5.

5 Experimental Study

To verify our proposed feature using LOD for named entity classification, we conducted extensive experiments to demonstrate the effectiveness of our proposed method.

5.1 Experimental Setup

Datasets. We have tested our proposed method on two datasets. The first dataset (denoted as $Data_Q$) is extracted from an IBM question-answering system for open-domain factoid questions. We randomly selected 400 questions. For each question we manually labeled the type of entity that was expected as an answer to the question (i.e., the target type) and we also extracted the top 10 candidate answers that were generated by the system. We asked one test person to determine whether the candidate answer belongs to the target type. For all candidate answers that belonged to the target types, we generated a list of (candidate answer, type) pairs, called *ground-truth* (denoted as $Data_Q^{true}$); for all candidate answers that did not belong to the target types, we generated a list of (candidate answer, type) pairs, called *ground-wrong* (denoted as $Data_Q^{wrong}$). We obtained 1,967 pairs for ground-truth and 3,053 pairs for ground-wrong. There are 114 distinguishing target types, which reflects the fact that the data was from open-domain. The second dataset (denoted as $Data_P$) is the *People Ontology*, which is a benchmark in [16], [15]. It was extracted from WordNet and contained 1,657 distinct person instances arranged in a multilevel taxonomy having 10 fine-grained categories, such as chemist or actor. Each instance and its category could also be considered as a pair in ground-truth. From this dataset, we obtained 1,618 pairs for ground-truth. The obtained dataset is denoted as $Data_P^{true}$.

Evaluation Metric. We conducted two types of evaluations. First, we measured how our feature and scoring method performed on ground-truth/ground-wrong. The three metrics used here, i.e., accuracy, false-rate, and unknown-rate, are described in Table 2, where N stands for total number of pairs.

Table 2. Descriptions for Used Metrics

Metric Name	Description	Measurement	Remarks
accuracy	#(correctly scored pairs)/N	correctness percentage of the scoring method	for ground-truth (resp. ground-wrong) dataset, the positive scores (resp. negative scores) are considered as correct
unknown-rate	#(pairs with score 0)/N	the coverage of the linked data	if the named entity is not indexed in the type knowledge base, we give the score 0
false-rate	#(incorrectly scored pairs)/N	incorrectness percentage of the scoring method	for ground-truth (resp. ground-wrong) dataset, the negative scores (resp. positive scores) are considered as incorrect

Second, we wanted to illustrate how our proposed feature helps to improve the performance of named entity classification. To do this, we compared the precision/recall of the classification with and without our feature.

5.2 Scoring Accuracy

This section demonstrates the scoring accuracy of LOD feature on the ground-true/wrong dataset. As introduced in Section 4.3, we have two strategies to

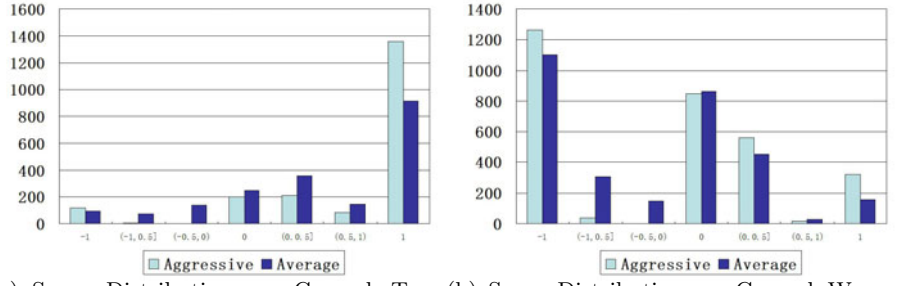
determine the final score, i.e., the aggressive strategy and the average strategy. We compared the performance of these two strategies on the ground-true/wrong of $Data_Q$ and $Data_P$ using the metrics introduced in Table 2. The results for the *aggressive* strategy and for the *average* strategy are shown in Table 3.

Table 3. The Scoring Performance on the Ground-Truth/Wrong Dataset

	Aggressive			Average		
Data Set	accuracy	unknown-rate	false-rate	accuracy	unknown-rate	false-rate
$Data_Q^{true}$	83.4%	10.2%	6.35%	71.9%	12.6%	15.5%
$Data_Q^{wrong}$	50.6%	25.9%	23.5%	53.9%	26.2%	19.9%
$Data_P^{true}$	91.5%	7.85%	0.65%	88.3%	9.88%	1.83%

We first observed that the aggressive strategy resulted in high accuracy, i.e., 83.4% for $Data_Q^{true}$ and 91.5% for $Data_P^{true}$ on ground-true data. This indicates that the linked data provides high-quality type information with good coverage and that our scoring method measures the type similarity quite well. Second, it shows that the accuracy for the ground-wrong data from $Data_Q$ is a little lower. For the higher unknown-rate, although in a question-answering project the correct response should be a fact, the extracted candidate answers may not all be fact entities, for example, "Germany history" or "the best movie." Therefore, these candidates are not covered by the linked data. Actually, named entity classification does not target for this kind of entity. For the higher false-rate, there are two main reasons: (1) our strategy for final score gives higher preference on the positive score. One name may correspond to multiple instances in our type knowledge base, and we do not know which instance the named entity corresponds to. Giving a relaxed score is safer for our linked data feature. With the additional context information of the named entity, information that can be used as additional features in named entity classification, the false-rate could be reduced; (2) the type knowledge base contains only more general types than the target type. For example, the named entity has a type *person*, and the target type is *actor*. The named entity may not be an *actor*, but as the type *person* is not disjoint with the type *actor*, we could not give a negative score for this named entity with respect to the type *actor*. This may incur the false-positive cases.

To compare the aggressive strategy with the average strategy, we observed that the aggressive strategy outperforms the average strategy for the ground-true data while the average strategy outperforms the aggressive strategy for the ground-wrong data. Because the aggressive strategy gives a higher priority on the positive score, the ground-true pairs benefit more from this strategy. For ground-wrong data, as mentioned in the previous paragraph, the false-positive instances may occur due to the more general types in the type knowledge base. Using the aggressive strategy, this false-positive instance will affect the final score; while using the average strategy, the false-positive instance may be compensated by other correctly scored instances. That is why the average strategy is better than



(a) Score Distribution on Ground True $Data_Q$

(b) Score Distribution on Ground Wrong $Data_Q$

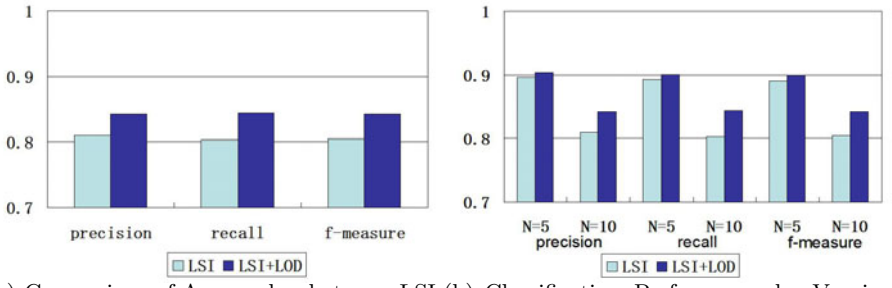
Fig. 4. Score Distribution on the Dataset

the aggressive strategy for ground-wrong data. When combining the ground-true and ground-wrong data together for $Data_Q$, the aggressive strategy is better in general.

Fig. 4(a) and (b) illustrate the distribution of the scores between the range [-1,1] for the ground-true and ground-wrong datasets from $Data_Q$. We can observe that for correct scores (i.e., positive scores for ground-true and negative scores for ground-wrong), most are exactly correct (i.e., score 1 for ground-true and score -1 for ground-wrong). This indicates the accuracy of our scoring. To compare the aggressive strategy and the average strategy, it shows that the aggressive strategy gives more exact scores and the average aggressive tends to be more balanced. The reason is that the average strategy computes the average of all instance scores, so the exact score may be reduced by other scores. Considering both the accuracy and the distribution, we suggest using the aggressive strategy.

5.3 Impact on the Machine Learning for Classification

This section reports the results of adding our proposed feature into the feature space of an existing named entity classification method using latent semantic kernels (LSI) [15]. We use LSI to denote the existing approach and LSI+LOD to denote the approach with our feature. Given each instance from the people ontology, the multi-context is derived by collecting 100 English snippets by querying Google^{TM}. The proximity matrices are derived from 200,000 Wikipedia articles. and the features are then generated using jLSI code [14]. With respect to our LOD feature, for each target class i, one feature is added to measure the probability of whether the instance belongs to the class i. Therefore, for 10 classes, we need to add 10 features. We use the KNN (k=1) method to do the classification. Fig. 5(a) compares the precision/recall/f-measure between the approaches LSI and LSI+LOD. It is shown that LSI+LOD outperforms the LSI on both precision and recall, and then f-measure. Specifically, the precision is improved from 81% to 84.3%, the recall is improved from 80.3% to 84.3%, and the f-measure is improved from 80.5% to 84.3%.



(a) Comparison of Approaches between LSI and LSI+LOD $Data_P$

(b) Classification Performance by Varying the Number of Classes $Data_P$

Fig. 5. Effectiveness of Linked Data Feature

Fig. 5(b) illustrates the performance on different numbers of target classes. The bars with $N = 5$ are the results for the 5-class dataset where we select 5 classes from the 10 classes and extract the corresponding instances to generate the dataset. The bars with $N = 10$ are the results for the 10-class dataset. It shows that the improvement of LSI+LOD over LSI is larger on the dataset with $N = 10$ than the dataset with $N = 5$. Specifically, for $N = 5$, the f-measure is improved by 1.1% using LSI+LOD, and for $N = 10$, the f-measure is improved by 4.7% using LSI+LOD. This indicates that as the number of target classes grows larger, the improvement using our LOD feature becomes greater. The reason is that as the number of classes becomes larger, the classification accuracy using traditional features, such as word characteristic and word context, becomes lower. However, as the linked data knowledge base provides more fine-grained type information, the scoring is still accurate for fine-grained classes. Therefore, the improvement of the method using the LOD features becomes greater.

We also conducted the experiments on a dataset from an IBM question-and-answering system to compare the performance of using feature TyCorLOD only or using both TyCorLOD and AnTyCorLOD, as discussed in Section 4.4. The results verify that the strategy to use both TyCorLOD and AnTyCorLOD outperforms the strategy of using only TyCorLOD. Due to space limitations, we omit the detailed results here.

6 Related Work

Named entity classification has been studied by many researchers [13], [10], [15], [12], [16]. The early named entity classification task considered a limited number of classes. For example, the MUC named entity task [18] distinguishes three classes, i.e., PERSON, LOCATION and ORGANIZATION, and the CoNLL-2003 adds one more class, i.e., MISC. The dominant technique for addressing this task is supervised learning where the labeled training data for the set of classes is provided [7]. Recently, a more fine-grained categorization of named entities has been studied. Fleischman and Hovy [12] examined different features

and learning algorithms to automatically subcategorize person names into eight fine-grained classes. Cimiano and Völker [8] have leveraged the context of named entities and used unsupervised learning to categorize named entities with respect to an ontology. In short, machine-learning approaches are widely adopted by the proposed approaches and the features used for the learning method can be classified into three categories: (1) the word-level feature [7], such as the word case or digit pattern; (2) handcrafted resources, such as gazetteers or lists [13], [12]; (3) the context of the named entities [8], [15].

The feature proposed in this paper is different from these existing approaches. It exploits a resourceful knowledge base, i.e., linked open data. This knowledge base is different from precomplied lists for classifying certain categories, as used in [13]. The linked data is published by various data providers and has a broad coverage. Because information in the linked data is still growing, more type information will be available in the future. The linked data feature is orthogonal with existing features and can be combined with them in order to improve the performance of named entity classification, including both the three-class task and fine-grained classification.

7 Conclusion and Future Work

In this paper, we proposed to explore the extensive type information provided by LOD to generate additional features, and these new features, together with existing features, can be used in machine-learning techniques for named entity classification. Specifically, in the first step, we proposed a mechanism to generate a type knowledge base that precisely and completely captures the type information for all possible named entity strings. We then proposed scoring strategies to generate feature scores based on the precomputed type information. Our experimental results verified the effectiveness of the proposed method and indicated that the improvement margin becomes larger as the number of target classes grows. In the future, we plan to investigate more data sources for LOD in order to provide better coverage.

Acknowledgements

We thank Christopher Welty and Aditya Kalyanpur for their valuable suggestions regarding this paper.

References

1. Linked data, `http://linkeddata.org/`
2. Opencyc, `http://www.cyc.com/opencyc`
3. Redirects, `http://en.wikipedia.org/wiki/Redirects_on_wikipedia`
4. Ahn, D., Jijkoun, V., Mishne, G., Muller, K., de Rijke, M., Schlobach, S.: Using wikipedia at the trec qa track. In: Proceedings of the 13rd Text REtrieval Conference, TREC 13 (2004)

5. Banerjee, S., Pedersen, T.: Extended gloss overlaps as a measure of semantic relatedness. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (2003)
6. Banko, M., Cafarella, M.J., Soderland, S., Boardhead, M., Etzioni, O.: Open information extraction from the web. Communications of the ACM (2008)
7. Bikel, D.M., Miller, S., Schwartz, R., Weischedel, R.: Nymble: High-performance learning name-finder. In: Proceedings of the 5th Conference on Applied Natural Language Processing (1997)
8. Cimiano, P., Volker, J.: Towards large-scale, open-domain and ontology-based named entity classification. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP (2005)
9. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Yates, D.S.W.S.A.: Web-scale information extraction in knowitall. In: Proceedings of the 13th International Conference on World Wide Web, WWW (2004)
10. Evans, R.: A framework for named entity recognition in the open domain. In: Proceedings of the Recent Advances in Natural Language Processing, RANLP (2003)
11. Fellbaum, C. (ed.): Wordnet: An electronic lexical database. MIT Press, Cambridge (1998)
12. Fleischman, M., Hovy, E.: Fine-grained classification of named entities. In: Proceedings of the 19th International Conference on Computational Linguistics, Coling (2002)
13. Ganti, V., Konig, A.C., Vernica, R.: Entity categorization over large document collections. In: Proceedings of the 14th ACM SIGKDD International Conference On Knowledge Discovery & Data Mining (2008)
14. Giuliano, C.: jLSI a for latent semantic indexing (2007) Software available at, `http://tcc.itc.it/research/textec/tools-resources/jLSI.html`
15. Giuliano, C.: Fine-grained classification of named entities exploiting latent semantic kernels. In: Proceedings of the 13rd Conference onCcomputational Natural Language Learning, CoNLL (2009)
16. Giuliano, C., Gliozzo, A.: Instance-based ontology population exploiting named-entity substitution. In: Proceedings of the 22nd International Conference on Computational Linguistics, Coling (2008)
17. Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Rus, V., Morarescu, P.: Falcon: Boosting knowledge for answer engines. In: Proceedings of 9th Text REtrieval Conference, TREC 9 (2000)
18. Hirschman, L., Chinchor, N.: Muc-7 named entity task definition. In: Proceedings of the 7th Message Understanding Conference, MUC-7 (1997)
19. Kwok, C.C.T., Etzioni, O., Weld, D.S.: Scaling question answering to the web. In: Proceedings of the 10th World Wide Web Conference, WWW (2001)
20. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. In: Linguisticae Investigationes (2007)