Learning Relatedness Measures for Entity Linking

Diego Ceccarelli[1,2,3], Claudio Lucchese[1], Salvatore Orlando[1,4],
Raffaele Perego[1], Salvatore Trani[1,2]

[1] National Research Council of Italy [2] University of Pisa
[3] IMT Lucca [4] Ca' Foscari University of Venice

{firstname.lastname}@isti.cnr.it

ABSTRACT

Entity Linking is the task of detecting, in text documents, relevant mentions to entities of a given knowledge base. To this end, entity-linking algorithms use several signals and features extracted from the input text or from the knowledge base. The most important of such features is *entity relatedness*. Indeed, we argue that these algorithms benefit from maximizing the relatedness among the relevant entities selected for annotation, since this minimizes errors in disambiguating entity-linking.

The definition of an effective relatedness function is thus a crucial point in any entity-linking algorithm. In this paper we address the problem of learning high-quality entity relatedness functions. First, we formalize the problem of learning entity relatedness as a learning-to-rank problem. We propose a methodology to create reference datasets on the basis of manually annotated data. Finally, we show that our machine-learned entity relatedness function performs better than other relatedness functions previously proposed, and, more importantly, improves the overall performance of different state-of-the-art entity-linking algorithms.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—|*Information Filtering, Search process*

General Terms

Algorithms, Design, Experimentation.

Keywords

Entity linking, relatedness measures, learning to rank

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505711.

1. INTRODUCTION

Document enriching is today a fundamental technique to improve the quality of several text analysis tasks, including Web search [21, 24]. In this work we specifically address the *Entity Linking Problem*: given a plain text, the entity linking task aims at identifying the small fragments of text (in the following interchangeably called *spots* or *mentions*) referring to any *named entity* that is listed in a given knowledge base, e.g., Wikipedia. The ambiguity of natural language makes it a non trivial task. The same entity can be in fact mentioned with different text fragments, e.g., "President Obama" or "Barack Obama". On the other hand, the same mention may refer to different entities, e.g., "President" may refer to the U.S. president or to Alain Chesnais, the president of the Association for Computing Machinery.

A typical entity linking system performs this task in two steps: *spotting* and *disambiguation*. The *spotting* process identifies a set of candidate spots in the input document, and produces a list of candidate entities for each spot. Then, the *disambiguation* process selects the most relevant spots and the most likely entities among the candidates. The *spotting* step exploits a given catalog of named entities, or some knowledge base, to devise the possible mentions of entities occurring in the input. One common approach to address this issue is resorting to Wikipedia [12, 19]: each Wikipedia article is considered to be a named entity, and the anchor texts associated with Wikipedia links a rich source of possible mentions to the linked entity. The spotter can thus process the input text looking for any fragment of text matching any of the Wikipedia mentions, and therefore potentially referring to an entity. Indeed, the spotter should detect all the mentions and find all the possible entities associated with such mentions. The coverage of the source knowledge base and the accuracy of the spotter have in fact a strong impact on the recall of the entity linking system [4]. To isolate the impact of spotting, we implemented a general framework for entity linking that uses the same spotting technique to feed several state-of-the-art disambiguation algorithms.

Let us introduce a simple example to describe how the entity linking process works:

> *On July 20, 1969, the Apollo 11 astronauts - Neil Armstrong, Michael Collins, and Edwin "Buzz" Aldrin Jr. - realized President Kennedy's dream.*

The text "President Kennedy" can be easily spotted and linked to *John F. Kennedy*, since in Wikipedia there are 98 anchors exactly matching such fragment of text and linking to the U.S. president page. In addition, the text "Apollo 11"

may refer to two distinct candidates: the famous spaceflight mission, or the 1996 film directed by Norberto Barba. Similarly, the text "Michael Collins" may refer to either the well known astronaut, or to the Irish leader and president of the Irish provisional government in 1922. Indeed, mentions to the latter (408) are much more frequent than those to the former (141)[1].

The above spots and the relative candidate entities are further processed during the *disambiguation* step. The goal of disambiguation is twofold. First, only relevant spots have to be filtered. For instance, the word "the" may refer to the entity associated with the definite article, but this linking might be relevant only for documents discussing the English grammar. Second, the best candidate entity for each spot has to be selected. This is usually done by considering the context of close mentions and by maximizing some measure of *relatedness* among the linked entities [6, 8, 13, 19, 22]. In our example, the astronaut "Michael Collins" and the "Apollo 11" spaceflight mission entities are preferred since they are clearly strongly related to each other and to the other entities found in the document, i.e., Buzz Aldrin and John F. Kennedy.

The effectiveness of the *entity relatedness* function adopted is thus a key-point for the accuracy of any entity-linking algorithm. In this work we propose a machine learning approach to devise a high-quality entity relatedness function. The main contributions of this paper are:

- a formalization of the problem of devising entity relatedness functions as a learning-to-rank problem;
- a novel technique to build benchmark datasets for learning and testing entity relatedness functions;
- an extensive experimentation showing that our automatically learned function outperforms state-of-the-art relatedness functions. More importantly, our approach can improve the performance of a whole class of entity-linking algorithms;
- an open source publicly available framework for addressing the entity linking problem and evaluating new algorithms in a fair test environment.

The paper is organized as follows. In Section 2 we formalize the problem of learning automatically a entity relatedness function. In Section 3 we discuss related works, and how entity relatedness functions are used in these works. In Section 4 we evaluate some machine learned entity relatedness functions, and in Section 5 we evaluate their impact on entity linking algorithms. Finally, Section 6 draws some final conclusions.

2. ENTITY RELATEDNESS DISCOVERY

Given a set of known entities $\mathcal{E}$ from a knowledge base $\mathcal{KB}$, and an unstructured text document D, entity linking aims at identifying all the relevant mentions in D to the entities of $\mathcal{E}$. The entity linking process involves two steps that we are going to detail in the following.

Spotting and Candidate Selection. Spotting aims at identifying spots, i.e., contiguous sequences of n terms (n-*grams*) occurring in D that might mention some entity $e \in \mathcal{E}$. A common method to identify the spots $S_D = \{s_1, s_2, \ldots\}$ is to exploit a *controlled vocabulary* of spots $\mathcal{L}$, and to search the input document for the n-grams that exactly match an entry of this vocabulary.

When Wikipedia is used as $\mathcal{KB}$, each Wikipedia article identifies an entity, and the vocabulary $\mathcal{L}$ can be easily built by considering the article *titles* along with the *anchor texts* of all internal Wikipedia hyperlinks.

Each spot $s_i \in S_D$ is then associated with a set of candidate entities $C(s_i) \subseteq \mathcal{E}$. This is done by considering all the entities of $\mathcal{E}$ that are referred to in $\mathcal{KB}$ by using spot s_i as an anchor text. Unfortunately the same spot s_i can occur in different places of $\mathcal{KB}$ (and even of D!) and refer to distinct entities. Finally, we denote by $\epsilon(s_i) \in C(s_i)$ the entity that is actually mentioned by s_i in D.

Figure 1 illustrates the three spots $\{s_1, s_2, s_3\}$ detected in our text example. For each s_i, the outgoing dashed directed edges identify the set of candidate entities $C(s_i)$, where $\epsilon(s_i) \in C(s_i)$, i.e., the entity that is actually mentioned by s_i, is represented as a rectangle.

To limit the set of spots and candidate entities to the most meaningful ones, *link probability* and *commonness* properties can be usefully exploited [17]. The link probability for a spot s_i is defined as the number of times s_i occurs as a mention in $\mathcal{KB}$, divided by its total number of occurrences. This permits to discard spots that are rarely used as a mention to a relevant entity. For example the spot "July 20", introduced in the example of Section 1, occurs hundreds of times in Wikipedia, and, even if it is the title of an article, only in a few cases it used as anchor text.

The *commonness* of a candidate $c \in C(s_i)$ for spot s_i is instead defined as the fraction between the number of occurrences of s_i in $\mathcal{KB}$ actually referring to c, and the total number of occurrences of s_i in $\mathcal{KB}$ as a mention to an entity. For example, the spot "Michael Collins" may refer to more than 20 different entities, but the Irish revolutionary leader (421 mentions, commonness 0.5), the film about his life (126 mentions, commonness 0.15) and the astronaut (132 mentions, commonness 0.15) are largely the most common.

Setting a threshold on minimum linking probability and minimum commonness has been proven to be a simple and effective strategy to limit the number of spots and associated candidates, without harming the recall of the entity linking process [19].

Disambiguation and Linking. Since in many cases we have several candidates for a single spot s_i (i.e., $|C(s_i)| > 1$), the spot has to be disambiguated by choosing the right entity $\epsilon(s_i)$ among the candidates $C(s_i)$. For each spot, a disambiguation algorithm outputs the selected entity and a confidence score. This confidence score can be used to select the most likely matching entities, and to trade precision with recall.

In order to choose the best entity for a spot, disambiguation may exploit different signals and features. These include commonness and linking probability, and many others features considering the text surrounding the spot, and the other spots of the document. The most important of such features is *entity relatedness*, usually defined as a real function $\rho : \mathcal{E} \times \mathcal{E} \to [0, 1]$, where 0 and 1 are the minimum and maximum relatedness measure, respectively. To guarantee the accuracy of entity linking, the entities selected by the disambiguation process to be linked to the detected spots have in fact to be strongly related to each other.

[1]Throughout this paper, we used the 04/03/2013 dump, available at `http://dumps.wikimedia.org/enwiki/20130403/enwiki-20130403-pages-articles.xml.bz2`

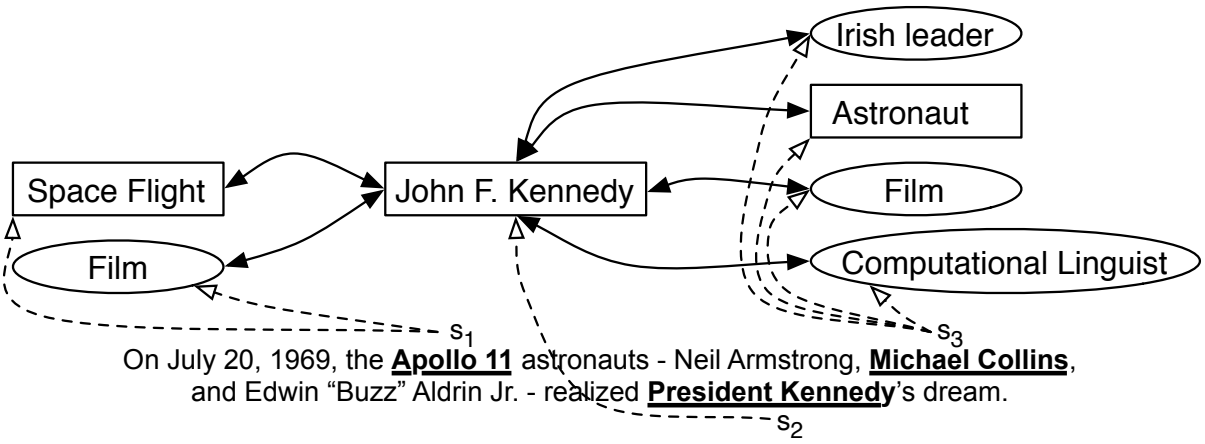

Figure 1: Entity relationships for spotting and disambiguation. Three spots extracted from the above example are underlined: s_1 = "*Apollo* 11", s_2 = "*President Kennedy*", and s_3 = "*Michael Collins*". The graph shows relatedness edges connecting candidates entities. For simplicity of representation the candidates for spot s_3 are omitted. Rectangles are used to indicate correctly disambiguated entities, while ellipses refer to other candidate entities.

Figure 1 shows the relatedness graph referred to our example. The selection of the best entities is often implemented on top of this relatedness graph, where edges are weighted by some entity relatedness function. Therefore, the role of such entity relatedness function is crucial for the accuracy of the disambiguation process.

Even if the definition of a entity relatedness function is not a trivial task, several works agree on the effectiveness of the Wikipedia-based relatedness function proposed by Milne and Witten [19, 18]. The relatedness between two entities a and b is in this case computed by exploiting the graph structure of Wikipedia:

$$\rho^{\mathsf{MW}}(a,b) = 1 - \frac{\log(\max(|in(a)|,|in(b)|)) - \log(|in(a) \cap in(b)|)}{\log(|W|) - \log(\min(|in(a)|,|in(b)|))}$$

where W is the set of all Wikipedia entities, while $in(a)$ and $in(b)$ are the sets of Wikipedia articles linking to a and b, respectively. When $|in(a) \cap in(b)| = 0$, we have $\rho^{\mathsf{MW}}(a,b) = 0$. In addition, ρ^{MW} is maximum (equal to 1) when $in(a) \cap in(b) = in(a) = in(b)$, and thus all the articles that cite a also cite b, and vice versa.

The ρ^{MW} function, promoting entities that are co-cited by the same Wikipedia articles, is considered the *state-of-the-art* relatedness measure, adopted also in [8, 12, 14]. On the other hand, there is no guarantee that ρ^{MW} would produce a proper scoring of the candidate entities.

EXAMPLE 2.1. *Given the entities a="Andronicus of Rhodes", b="Chondrichthyes", and c="Aristotle" occurring in a document, we have that $\rho^{\mathsf{MW}}(a,b) = 0.54$ and $\rho^{\mathsf{MW}}(a,c) = 0.56$[2]. The connection between entities a and c is very strong since Andronicus of Rhodes is credited with the production of the first reliable edition of Aristotle's works. The (unexpected) high relatedness score between entities a and b is instead due to a single co-citing Wikipedia article (which is c) that reports about Aristotle's studies of a group of fishes he named selachians, a.k.a. chondrichthyes. Therefore, in this case a single co-citation is enough to produce an unexpected high value $\rho^{\mathsf{MW}}(a,b)$, which is similar to the expected large value of $\rho^{\mathsf{MW}}(a,c)$.*

Another interesting observation is that ρ^{MW} is symmetric: Andronicus of Rhodes is relevant to Aristotle, to the same degree Aristotle is relevant to Andronicus of Rhodes.

Our claim is that a good entity relatedness function ρ can improve the performance of a large class of entity linking algorithms. In Section 3 we will discuss related works and show the crucial role of entity relatedness functions in many proposals. Here, we propose a set of properties that an optimal entity relatedness measure should satisfy, and we formalize the problem of discovering a good entity relatedness function into a learning-to-rank problem.

Relatedness as a Ranking Function. Suppose that an entity linking algorithm identifies only two spots s_h and s_i for a document D, and for these spots it generates the two sets of candidate entities $C(s_h)$ and $C(s_i)$ respectively. Most disambiguation algorithms assume that if one of the candidate entities in $C(s_h)$ is highly related to another entity in $C(s_i)$, then it is very likely that they are the entities $\epsilon(s_h)$ and $\epsilon(s_i)$ actually mentioned by the two spots.

We claim that a good entity relatedness function ρ should promote the relatedness of correct entities: given entity $\epsilon(s_h)$, its relatedness with $\epsilon(s_i)$ should be larger then that with any other candidate in $C(s_i)$. This should hold for every spot $s_i \neq s_h$.

PROPOSITION 2.1. *Given D, $S_D = \{s_1, s_2, \ldots\}$, and, for each spot s_i, $C(s_i)$ and $\epsilon(s_i)$, a relatedness function ρ improves entity-linking accuracy if the following constraint holds:*

$$\forall s_h \in S_D, \quad \forall s_i \in S_D \setminus \{s_h\}, \quad \forall c \in C(s_i) \setminus \{\epsilon(s_i)\} : \\ \rho(\epsilon(s_h), \epsilon(s_i)) \; > \; \rho(\epsilon(s_h), c). \qquad (1)$$

Indeed, the constraint in Eq. 1 nicely fits into a learning-to-rank based formulation [16]. The relatedness function ρ

[2]The values to compute the two ρ^{MW} measures are: $|in(a)| = 24$, $|in(b)| = 261$, $|in(c)| = 3502$, $|in(a) \cap in(b)| = 1$, $|in(a) \cap in(c)| = 17$, and $|W| = 4,255,306$.

can be in fact modeled as a ranking function, with entity $\epsilon(s_h)$ used as a query. According to the above Proposition, function ρ should score all the entities actually mentioned in the document, i.e. $\epsilon(s_i)$ for all $s_i \neq s_h$, higher than any other false-positive candidate, i.e. $c \in C(s_i) \setminus \{\epsilon(s_i)\}$ for all $s_i \neq s_h$.

Given document D and spots $S_D = \{s_1, s_2, \ldots, s_h, \ldots\}$, we denote by $\mathcal{R}_D^h = \cup_{i \neq h} C(s_i)$ the set of candidates to be ranked for query $\epsilon(s_h)$, and by $\mathcal{E}_D^h = \cup_{i \neq h} \epsilon(s_i)$ the set of relevant entities for the query, where $\mathcal{E}_D^h \subseteq \mathcal{R}_D^h$. From an information retrieval perspective, items in $\mathcal{R}_D^h$ are relevant for query $\epsilon(s_h)$ if and only if they belongs to $\mathcal{E}_D^h$. Note that we are considering the spots of D altogether, and therefore there are potentially many related entities to the query $\epsilon(s_h)$ that should be ranked high. Let us denote with π_ρ^h the score descending ordering of $\mathcal{R}_D^h$ induced by our ranking relatedness function ρ for query $\epsilon(s_h)$. According to Proposition 2.1, a scored list π_ρ^h is effective when entities in $\mathcal{E}_D^h$ are in the top positions of the list. We can thus measure the effectiveness of our ranking relatedness function by using common information retrieval quality metrics such as *NDCG* [15]. In our context we define $DCG(\pi_\rho^h)$ as:

$$DCG(\pi_\rho^h) = \sum_{j=1}^{|\pi_\rho^h|} \frac{[\![\pi_\rho^h[j] \in \mathcal{E}_D^h]\!]}{\log(j+1)}$$

where $\pi_\rho^h[j]$ denotes the j-th item of the scored list, and $[\![x]\!]$ equals 1 if x is true and 0 otherwise. *NDCG* is defined as the usual normalized version of *DCG*.

We can now introduce the *Entity Relatedness Discovery* problem.

> PROBLEM 2.1 (ENTITY RELATEDNESS DISCOVERY).
>
> *Let $\mathcal{D}$ be a collection of entity-linked documents, where for each document $D \in \mathcal{D}$ and every relevant spot s_i of S_D we know $\epsilon(s_i)$. Given the entity $\epsilon(s_h)$ and the set $\mathcal{R}_D^h$, a ranking relatedness function ρ induces an ordering π_ρ^h of $\mathcal{R}_D^h$.*
> *The* Entity Relatedness Discovery Problem *requires to find the function ρ that maximizes the ranking quality:*
>
> $$\frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \frac{1}{|\mathcal{S}_\mathcal{D}|} \sum_{s_h \in S_D} \mathsf{NDCG}(\pi_\rho^h)$$

In our experiments, we chose to optimize *NDCG* to find a good entity relatedness function, but we used several other ranking quality functions to assess the goodness of results.

Unlike previous approaches, we do not suggest a specific novel entity relatedness function. Rather, we define a learning-to-rank framework to discover the optimal entity relatedness function.

3. RELATED WORKS

In the following we discuss how the notion of *entity relatedness* is exploited by state-of-the-art entity linking algorithms. Emphasis is given to the solutions proposed in [19] and [12] and [8] which are the most relevant proposals in the field, and they are all adopting ρ^{MW} as entity relatedness function. We show that the entity relatedness function defined in Proposition 2.1 can replace ρ^{MW} since it fits better the framework and the objectives of the above algorithms.

WikiMiner [19]. Given a document D, let us consider its spots S_D and for each spot s_i the associated set of candidates $C(s_i)$. Let us suppose that a subset of the spots are associated with only a single entity. We denote with $U \subseteq \mathcal{E}$ the *context*: the set of unambiguous entities linked to spots in S_D, i.e., $U = \bigcup_{|C(s_i)|=1} \epsilon(s_i)$. The WikiMiner algorithm exploits the entities in U as safe reference points to help the disambiguation of the other ambiguous spots for which $|C(s_i)| > 1$ holds. The idea is to select for every ambiguous spot of S_D the entity which is, on average, the most related with the "safe" entities in U. The relatedness function adopted is ρ^{MW}. It is worth noting that not all the entities in U have the same impact: an entity $u \in U$ is in fact considered of high quality if it is strongly related to the other entities in U, and if the *link probability* of the corresponding spot is high. These two criteria allow a weight w_u, $0 \leq w_u \leq 1$ to be assigned to each entity u in U. Note that the main aim of this weight is to reduce the impact of low-quality entities occurring in U. When applied to our simple example, the low resulting weight would demote the importance of the safe entity "July 20".

Every candidate c in $C(s_i)$ is scored according to the following function:

$$score(c \mid U) = \frac{1}{\sum_u w_u} \sum_{u \in U} w_u \cdot \rho^{\mathsf{MW}}(u, c). \qquad (2)$$

It is easy to show that the accuracy of the disambiguation would improve if we adopted, instead of ρ^{MW}, a relatedness function ρ that satisfies our Proposition 2.1.

Given an entity $u \in U$, we can rewrite Eq. 1 and derive as follows:

$$\begin{aligned} \rho(u, \epsilon(s_i)) &> \rho(u, c) \quad \Rightarrow \\ \frac{1}{\sum_u w_u} \sum_{u \in U} w_u \cdot \rho(u, \epsilon(s_i)) &> \frac{1}{\sum_u w_u} \sum_{u \in U} w_u \cdot \rho(u, c) \quad \Rightarrow \\ score(\epsilon(s_i) \mid U) &> score(c \mid U) \end{aligned}$$

Therefore, a relatedness function satisfying Proposition 2.1 would always correctly rank entity $\epsilon(s_i)$ higher than any other candidate for the corresponding spot s_i even when integrated in the WikiMiner framework.

Interestingly, the authors of [19] use machine learning to combine the above relatedness score with other two features: *commonness* and *context quality* (measured as $\sum w_u$). They experiment with a training set built from 500 Wikipedia articles. However, machine learning is not exploited to improve the relatedness function as in our proposal.

Referent Graph [12]. *Referent Graph*, a graph-based method still exploiting the relatedness function ρ^{MW}, is proposed in [12]. Let $RG(V, E)$ be a weighted directed graph where the nodes includes all the spots s_i of S_D and candidates $C(s_i)$. RG has a directed edge from s_i to every $c \in C(s_i)$, and reciprocal edges connecting every pair of candidate entities a and b, $a \in C(s_i), b \in C(s_h), i \neq h$. Spot-candidate edges (s_i, c) are weighted according to the cosine similarity between the Wikipedia article corresponding to entity c and a local context window of 50 words around the spot s_i. The candidate-candidate edges (a, b) are weighted by using ρ^{MW}. Finally, weights are normalized

so that weights on outgoing edges from a given node always sum up to 1. The graph shown in Figure 1 is a toy referent graph, where relatedness edges connecting candidates entities for the spot s_1 with candidates entities for the spot s_3 are omitted for clarity.

The score of a candidate entity for a given spot is given by the steady state distribution of a random walk with restarts [20] in RG, where candidate nodes have restart probability 0, and spot nodes have a restart probability proportional to their inverse document frequency score in the Wikipedia corpus. Also in this case, assigning a different restart probability to spot nodes, and weighting as above explained spot-candidate edges is aimed to limit the impact of non relevant or incorrectly matched mentions.

The rationale of the random walk approach is to evaluate the relationships among the whole set of candidates simultaneously, in contrast to previous methods where the scores of candidate entities are assigned independently of each other. Also in this algorithm the choice of the entity relatedness function ρ has a strong impact on the performance since it drives the random walk process. A set of entities being very related to each other is likely to produce a reinforcement loop, and eventually include the most probable states of the random walk.

Even if we do not provide a formal proof as for WikiMiner, it is clear that a good relatedness function should promote the reciprocal relatedness among the right entities in the graph, thus helping the random walk to converge to the correct ranking of candidates.

A similar approach is used in [26], where a slightly differently weighted referent graph is pruned progressively by removing iteratively the node with the lowest weighted degree (sum of the weights of incoming edges). Even in this case, the weights of candidate-candidate edges are computed with the ρ^{MW} relatedness function. The paper do not compare performances with those of [19], [12], or other algorithms, and it is thus difficult to estimate the impact of this proposal.

TAGME [8]. TAGME is an annotation framework focussing on efficiency that exploits two main features: *commonness* and the ρ^{MW} relatedness. First, candidate entities for a spot s_i are ranked according to their average relatedness with other candidate entities for spots $s_j \neq s_i$, weighted by their commonness. Then, from the top 30% candidates of the resulting ranked list, the entity with the largest commonness is finally selected.

Also this algorithm would benefit by a relatedness function satisfying Proposition 2.1, since it would help to boost the score of the actual entities mentioned in the document. However, the benefit is limited, since the relatedness function impacts more on the pruning irrelevant candidates, while the final choice of the best entity is mainly driven by the commonness feature.

Other approaches. Relatedness function ρ^{MW} is partially inspired by the so-called Normalized Google Distance (NGD) [5], which borrows from Kolmogorov complexity and information distance concepts. While NDG is tailored to measure similarity between words or phrases, ρ^{MW} measure is specifically tailored to entities represented in a graph structure such as the one of Wikipedia. In [10] another Wikipedia-based relatedness measure, named ESA, is proposed. A word is represented in a high dimensional space by considering for each Wikipedia article the relevances of the word in the article, and by summing such score vectors for longer text fragments. Also [13] investigates new text-based relatedness measures that try to go beyond link-based similarities. The study conducted in [18] shows however that ESA has a performance similar to that of ρ_{Milne}, with the latter being much cheaper to be computed since it does not require to index the whole Wikipedia textual content. In [22], the authors improve only slightly the solution proposed in [6], but they do not provide any comparison with [19, 12].

The authors of [23] propose a machine learning approach to rank entity-based facets related to a given Web search query. Since the paper focuses on a special set of entities, such as monument and celebrities, the presented technique exploits information coming from image search queries and Flickr image tags. The goal of [23] is not to discover the degree of relatedness between entities, but rather to suggest entities that are most likely to generate a large click through.

Finally, several works [7, 8, 19] exploit machine learning techniques for entity linking, but in this paper we use learning to rank for improving the *relatedness function*, which is important for improving quality of entity linking task as well in other tasks, such as entity ranking [1] or entity suggestion [2].

4. ENTITY RELATEDNESS EVALUATION

In the following we describe the methodology adopted to build a reference dataset for the learning process, the feature used to describe entities, and finally the performance of two automatically learned relatedness functions.

4.1 Building a benchmark dataset

In order to evaluate the impact of different relatedness functions, we built a benchmark dataset for Problem 2.1. This dataset, used to train and test our relatedness function, contains a set of tuples in the form $\langle \epsilon, \mathcal{R}_\epsilon, \mathcal{E}_\epsilon \rangle$, where ϵ is an entity occurring in a document D, $\mathcal{R}_\epsilon$ is a set of *candidate entities* possibly occurring in D, and $\mathcal{E}_\epsilon \subseteq \mathcal{R}_\epsilon$ are the *relevant entities* occurring in D besides ϵ.

In order to build these tuples, we need both positive and negative examples, i.e., positive ones from $\mathcal{E}_\epsilon$ and negative ones from $\mathcal{R}_\epsilon \setminus \mathcal{E}_\epsilon$. In most entity-annotated datasets, each document is annotated by one or more human assessors, who manually performed some kind of spotting and entity disambiguation tasks. Therefore, for each document D we only have positive examples, i.e. the set $\mathcal{A}_D$ of entities actually occurring in D. In addition, we do not know the spot in D that actually mentions each entity in $\mathcal{A}_D$.

Hence, to generate our dataset for training our relatedness function, we have to devise a sort of reverse annotation process, aimed at discovering the spots associated with the known entities, and the potential candidates of such spots. In this way, we identify also the negative examples to build the tuples $\langle \epsilon, \mathcal{R}_\epsilon, \mathcal{E}_\epsilon \rangle$. In more detail, we generate our benchmark dataset as described below:

1. we set up a knowledge base $\mathcal{KB}$ of entities based on Wikipedia. This contains entities, their mentions, i.e, anchor text of incoming links and page title, and the hyper-link structure; we created a vocabulary of entity mentions $\mathcal{L}$ containing only spots with link probability larger than 2%. Finally, for each spot we disregarded entities with commonness smaller than 3%;

2. we generate all *n-grams* of every given document D, with $n \leq 6$, and we match them against $\mathcal{L}$ to devise the spots S_D;
3. for each spot s_i in S_D, we retrieve the candidate entities $C(s_i)$ as the set of entities linked in Wikipedia by the same n-gram;
4. we finally consider the set of relevant entities $\mathcal{A}_D$ of D, as annotated by the human assessors. Since we do not know the real association between each spot s_i and the human annotated entities in $\mathcal{A}_D$, for each spot s_i we look for the actual entity $\epsilon(s_i)$ in the set $C(s_i) \cap \mathcal{A}_D$. If $C(s_i) \cap \mathcal{A}_D = \emptyset$, we assume that $\epsilon(s_i)$ is not known, and thus discard s_i. If $|C(s_i) \cap \mathcal{A}_D| > 1$[3], we also throw away s_i, since we are not able to disambiguate. Finally, if $|C(s_i) \cap \mathcal{A}_D| = 1$, then $\epsilon(s_i) \in C(s_i) \cap \mathcal{A}_D$ is the actual entity to link to s_i.

At the end of the process, for each document D we have: a set of spots S_D and, for each spot s_i, a set of candidate entities $C(s_i)$ and also the mentioned entity $\epsilon(s_i)$.

Thus, for every spot s_h of every document D, we can generate a tuple $\langle \epsilon, \mathcal{R}_\epsilon, \mathcal{E}_\epsilon \rangle$ for the benchmark dataset that contains: (i) the actually mentioned entity $\epsilon(s_i)$, (ii) the set of candidate entities for every other spot in the document, and (iii) the set of correctly linked, and thus related, entities in the document. By assuming that close spots are more likely to be related, we did not consider in this tuple generation step those spots occurring at a distance larger than $\omega = 150$ characters from the current spot associated with entity ϵ.

In our experiments we used a subset of the CoNLL 2003 entity recognition [14] task dataset, which includes annotated news stories of the Reuters Corpus V1. The dataset contains 1494 documents with an average length of 187 terms. Each document contains on average 11.7 entities.

We processed the corpus as explained above, and we thus built a dataset for evaluating the relatedness containing over 1.6 million tuples. We split the tuples in training, validation, and test set, respectively containing $977,514$, $369,798$ and $302,529$ records. Please observe that we take care of producing each dataset from a disjoint subset of documents in the collection, so that the tuples in the training and test sets were actually generated from a different subset of documents.

4.2 Features

A pair of entities a and b, for which the relatedness $\rho(a, b)$ has to be estimated, is represented by a set of 27 features shown in Table 1. The choice of such features is driven by the following considerations. First, we want to maximize their applicability by using publicly available data, and by using measures that can be easily applied to other entity knowledge bases, e.g., FreeBase.[4] For this reason we do not use click-through, access log, or query log based data, which are very difficult to obtain. We use instead several features related to the link structure of our knowledge base, such as the number of in-links $in(e)$ and out-links $out(e)$ of an entity e.

Second, there are applications of the entity relatedness function where the concept of spot is not applicable. Consider, for instance, the case of related entity recommendation where the query is a entity that is not associated with any spot. Therefore, we do not include features such as *link probability* and *commonness*.

Finally, we do not include text-based similarity measures, such as cosine similarity between Wikipedia articles pages, because this kind of approaches have been proven to perform similarly to the ρ^{MW} measure, but are much more computationally expensive [18].

Note that, by using the proposed machine learning approach, the feature set we adopt can be easily enriched with any additional feature, or by analyzing any other different knowledge base.

We categorize the features listed in Table 1 in three categories: *singleton*, *asymmetric* and *symmetric*.

Singleton features regard a single entity. They include only frequency and entropy, computed on the basis of the frequency of Wikipedia links to the entity article page. These features are computed for both entities of a given pair (a, b), resulting in four scores.

We claim that a relatedness function should not be symmetric. Consider for example the entities *Neil Armstrong* and *United States of America*: it seems reasonable that the relatedness of *United States of America* given *Neil Armstrong* is greater than the relatedness of *Neil Armstrong* given the *United States of America*. For this reason we included five asymmetric features, which are computed in both directions of the pair, resulting in ten scores.

Last, we considered 13 symmetric features, such as ρ^{MW}. Some of these features derive from asymmetric ones, and others are variations computed by considering outgoing links of an entity instead of incoming ones.

All the above features are computed on the basis of the same Wikipedia dump mentioned in the Section 1. Therefore, features are not extracted on the training or test dataset.

4.3 Quality of entity relatedness

To solve the Entity Relatedness Discovery problem, we used an existing tool for learning ranking functions, named RankLib.[5] This includes the implementation of several effective algorithms. We report the results of the two most effective: Gradient-Boosted Regression Trees [9] and LambdaMart [25]. We denote the models built with those algorithm ρ^{GDBT} and $\rho^{\lambda\mathsf{MART}}$.

Note that the two models differ significantly in the objective function being optimized. The $\rho^{\lambda\mathsf{MART}}$ model was built by a list-wise algorithm and minimizing $\mathsf{NDCG}@10$. This is indeed in perfect agreement with our definition of entity relatedness problem, and with the benchmark created. On the other hand, the ρ^{GDBT} model optimizes the error in predicting the class label (i.e., relevant vs. not relevant) of a given instance. Therefore, the prediction can be used to produce a ranking, but the model does not optimize the ranking directly.

In Table 3 we report the performance of the two relatedness functions ρ^{GDBT} and $\rho^{\lambda\mathsf{MART}}$, and compare it against ρ^{MW}. The improvement of using a machine learned function that exploits 27 features is apparent with every ranking quality measure adopted. If we consider $\mathsf{NDCG}@10$, $\rho^{\lambda\mathsf{MART}}$ improves over ρ^{MW} by a factor of 25%. The two learned functions have very similar performance, with no significant difference. Recalling to the Example 2.1, $\rho^{\mathsf{GDBT}}(a,b) = 0.0015$ while $\rho^{\mathsf{GDBT}}(a,c) = 0.66$: the value of ρ^{GDBT} (a,b) (*Andronicus of Rhodes* and *Chondrichthyes*) is low as we expected.

[3] In our datasets, this happens in only 2% of the cases.

[4] `http://www.freebase.com/`

[5] `http://people.cs.umass.edu/~vdang/ranklib.html`

Singleton Features	
P(a)	probability of a mention to entity a: $P(a) = \|in(a)\|/\|W\|$.
H(a)	entropy of a: $H(a) = -P(a)\log(P(a)) - (1-P(a))\log(1-P(a))$.
Asymmetric Features	
P(a|b)	conditional probability of the entity a given b: $P(a\|b) = \|in(a) \cap in(b)\| \ / \ \|in(b)\|$.
$\mathsf{Link}(a \to b)$	equals 1 if a links to b, and 0 otherwise.
$P(a \to b)$	probability that a links to b: equals $1/\|out(a)\|$ if a links to b, and 0 otherwise.
$\mathsf{Friend}(a, b)$	equals 1 if a links to b, and $\|out(a) \cap in(b)\|/\|out(a)\|$ otherwise.
$KL(a\|\|b)$	Kullback-Leibler divergence: $KL(a\|\|b) = \log \frac{P(a)}{P(b)} P(a) + \log \frac{1-P(a)}{1-P(b)}(1 - P(a))$.
Symmetric Features	
$\rho^{\mathsf{MW}}(a, b)$	co-citatation based similarity [19].
$J(a, b)$	Jaccard similarity: $J(a, b) = \frac{in(a) \cap in(b)}{in(a) \cup in(b)}$.
$P(a, b)$	joint probability of entities a and b: $P(a, b) = P(a\|b) \cdot P(b) = P(b\|a) \cdot P(a)$.
$\mathsf{Link}(a \leftrightarrow b)$	equals 1 if a links to b and vice versa, 0 otherwise.
$\mathsf{AvgFr}(a, b)$	average friendship: $(\mathsf{Friend}(a, b) + \mathsf{Friend}(b, a))/2$.
$\rho^{\mathsf{MW}}_{\mathsf{out}}(a, b)$	ρ^{MW} considering outgoing links.
$\rho^{\mathsf{MW}}_{\mathsf{in\text{-}out}}(a, b)$	ρ^{MW} considering the union of the incoming and outgoing links.
$J_{\mathsf{out}}(a, b)$	Jaccard similarity considering the outgoing links.
$J_{\mathsf{in\text{-}out}}(a, b)$	Jaccard similarity considering the union of the incoming and outgoing links.
$\chi^2(a, b)$	χ^2 statistic: $\chi^2(a,b) = (\|in(b) \cap in(a)\| \cdot (\|W\| - \|in(b) \cup in(a)\|) + -\|in(b) \setminus in(a)\| \cdot \|in(a) \setminus in(b)\|)^2 \cdot \cdot \frac{\|W\|}{\|in(a)\| \cdot \|in(b)\| (\|W\| - \|in(a)\|)(\|W\| - \|in(b)\|)}$
$\chi^2_{\mathsf{out}}(a, b)$	χ^2 statistic considering the outgoing links.
$\chi^2_{\mathsf{in\text{-}out}}(a, b)$	χ^2 statistic considering the union of the incoming and outgoing links.
$\mathsf{PMI}(a, b)$	point-wise mutual information: $\log \frac{P(b\|a)}{P(b)} = \log \frac{P(a\|b)}{P(a)} = \log \frac{\|in(b) \cap in(a)\| \|W\|}{\|in(b)\| \|in(a)\|}$

Table 1: Features for entity relatedness learning.

In order to gain some insight on the learned functions, and on the role of the different features, we run a study based on a naïve feature selection algorithm [11]. This algorithm ranks features by leveraging their similarity and the score of single-features models. It promotes effective features and demotes features similar to any other already selected one. Our objective here is not to find the best performing subset of features, but rather to investigate the importance of ρ^{MW} compared with other features not considered by state-of-the-art algorithm.

We measured the performance of the models built by means of LambdaMart algorithm when exploiting a single feature. In Table 2 we reported for each feature the score it can achieve. Recall that the relatedness function is required to learn a score of a candidate entity w.r.t. to a correct entity, which in the table are denoted with c and e respectively. Therefore, $P(c|e)$ is the conditional probability of finding the candidate entity c given our actually mentioned entity e, while $P(e|c)$ is the converse.

Features	Rank	NDCG@5	NDCG@10	P@5	P@10	MRR
P(c|e)	**1**	**0.68**	**0.72**	**0.47**	**0.33**	**0.80**
J(e, c)	**2**	0.62	0.66	0.44	0.31	0.75
Friend(e,c)	24	0.59	0.64	0.42	0.31	0.71
$\rho^{\mathsf{MW}}(e, c)$	19	0.59	0.63	0.42	0.31	0.72
$J_{\mathsf{in-out}}(e, c)$	26	0.60	0.63	0.42	0.30	0.74
$\mathsf{AvgFr}(e, c)$	**3**	0.57	0.62	0.40	0.30	0.69
P(e,c)	27	0.56	0.60	0.39	0.28	0.70
$\rho^{\mathsf{MW}}_{\mathsf{in\text{-}out}}(a, b)$	9	0.56	0.60	0.40	0.29	0.71
$J_{\mathsf{in-out}}(e, c)$	**4**	0.54	0.58	0.39	0.28	0.67
$\rho^{\mathsf{MW}}_{\mathsf{out}}(a, b)$	17	0.52	0.55	0.37	0.27	0.65
$\chi^2(e, c)$	25	0.51	0.55	0.37	0.27	0.64
$P(e\|c)$	22	0.48	0.54	0.36	0.28	0.60
H(c)	**5**	0.48	0.51	0.30	0.20	0.68
$\chi^2_{\mathsf{out}}(e, c)$	16	0.47	0.50	0.34	0.24	0.61
$\mathsf{AvgFr}(c, e)$	21	0.44	0.49	0.33	0.25	0.56
P(c)	13	0.47	0.49	0.29	0.19	0.66
$\mathsf{PMI}(e, c)$	23	0.42	0.48	0.32	0.25	0.53
$\chi^2_{\mathsf{in-out}}(e, c)$	11	0.44	0.46	0.33	0.23	0.58
$P(e \to c)$	18	0.37	0.38	0.24	0.15	0.55
$\mathsf{Link}(e \to c)$	20	0.37	0.38	0.24	0.15	0.55
$P(c \to e)$	12	0.35	0.36	0.22	0.14	0.52
$\mathsf{Link}(c \to e)$	15	0.31	0.33	0.21	0.14	0.46
$KL(c\|\|e)$	10	0.32	0.32	0.19	0.12	0.51
$\mathsf{Link}(c \leftrightarrow e)$	14	0.28	0.29	0.17	0.11	0.45
$KL(e\|\|c)$	8	0.26	0.28	0.17	0.11	0.44
P(e)	6	0.08	0.11	0.06	0.06	0.17
H(e)	7	0.08	0.11	0.06	0.06	0.17

Table 2: Entity ranking performance with a single feature. Features are sorted by *NDCG*@10.

Features	NDCG@5	NDCG@10	P@1	P@5	P@10	MRR
ρ^{MW}	0.59	0.63	0.62	0.42	0.31	0.72
$\rho^{\lambda\mathsf{MART}}$	**0.75**	**0.79**	**0.80**	**0.51**	**0.36**	**0.87**
ρ^{GDBT}	**0.75**	0.78	**0.80**	**0.51**	0.35	0.86

Table 3: Entity ranking performance of learned relatedness functions.

Results are very similar for every quality measure. Let's consider *NDCG* @10. The function ρ^{MW} is the fourth most effective feature with a score slightly below that of Jaccard and Friend functions. The most effective feature is $P(c|e)$, that is the conditional probability of the finding a mention to entity c given a Wikipedia page that mentions the entity e. Note that this quite intuitive feature behaves largely better than ρ^{MW} with a score of .72, but it is however far from the score achievable with the full set of features. Also, note that statistic $P(c|e)$ comes from a collection being completely different from the test set, since it was computed on the Wikipedia corpus and not on the train collection. A third interesting property is the asymmetry of this feature.

The second column of Table 2 reports the rank assigned by feature selection algorithm. While $P(c|e)$ is ranked first being the most effective features, ρ^{MW} is ranked only 19-th. This is due to the heuristic strategy of the algorithm, which demotes features if they are similar to previously selected ones.

Figure 2 shows the result of a multidimensional scaling mapping of the 27 features into a 2-dimensional space, thus

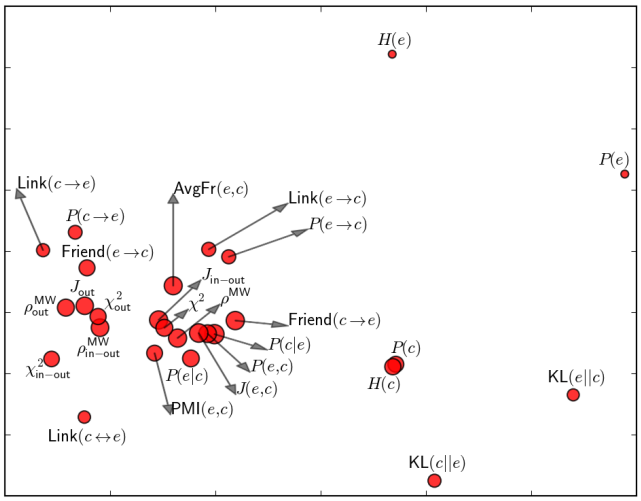

Figure 2: Multidimensional mapping of feature similarity computed using Kendall's τ coefficient. The size of each circle is proportional to the single-feature model score.

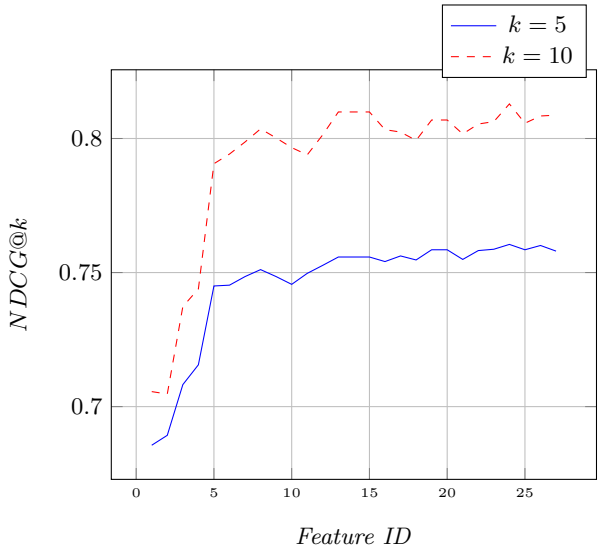

Figure 3: Incremental performance of $\rho^{\lambda\text{MART}}$.

approximately preserving feature similarity. We measured the similarity between a feature pair according to the Kendall's τ coefficient. We can identify two interesting clusters. The first contains ρ^{MW} together with $J_{\text{in-out}}$ and χ^2, and, indeed, the first two have identical performance. The second cluster includes the two best performing features $P(c|e)$, $P(e,c)$ and also Jaccard similarity. Even if the features in those clusters are similar w.r.t. the Kendall's τ coefficient, the score of the corresponding single feature model is very different, in particular for the best scoring $P(c|e)$. This suggest that the Kendall's τ coefficient may not be the best indicator in this context, and the feature selection may not be trivial.

Finally, in Figure 3 we measured the relative improvement provided by each feature. Features are sorted according to the ranking given by the feature selection algorithm mentioned above, and we measured the performance of the model by adding features incrementally. The model achieves almost optimal performance with the first 5 features. Optimal performance are achieved after 9 features are introduced in to the model. This shows that not all the features are necessary, and that a wisely chosen subset of features can provide optimal performance, or help in trading accuracy with efficiency. Several existing feature selection techniques can be used to this end. However, this is outside the scope of this work.

5. IMPACT ON ENTITY LINKING

We run a set of experiments to show how the automatically learned relatedness function can be profitably exploited by a class of entity disambiguation algorithms. We plugged the learned function into several annotation methods, which can be considered the state-of-the-art ones:

WikiMiner. The method proposed by Milne and Witten that exploits the relatedness function to identify a subset of not ambiguous entities called *context*. Given an ambiguous spot, the relatedness function is employed again to select the entity that is more coherent with the context;

Referent Graph. This method takes into account all the possible entities associated with the set of detected spots. The disambiguation is performed by modeling the entities as nodes of a complete graph, where the weight of each edge is the relatedness between the connected nodes;

TAGME. This annotator computes the weighted average relatedness between an entity and all the other possible entities associated with the spots. It disambiguates the entity by selecting the most common entities in the subset of the possible meanings with the highest average relatedness with the others.

With the exception of WikiMiner, the source code of the frameworks proposed is not publicly available. Furthermore the code released is not easy to extend for implementing other annotators. Annotation depends on several subtasks, i.e., *(i)* process Wikipedia (parse the dump, generate the possible spots, filter stop-words, etc.); *(ii)* perform the spotting (relying on a dictionary or using a name entity recognition framework, like the Stanford Named Entity Recognizer[6]); *(iii)* disambiguate the ambiguous spots, and *(iv)* rank entity candidates.

It is worth to observe that a good performance obtained in the first tasks may heavily impact on the performance of the whole system, as well as using a different dump of Wikipedia (i.e., old dumps contain less entities, but also have less ambiguity for each spot), or a different commonness or link probability thresholds. For these reasons, we strongly believe that for this kind of research it is important to share a unique framework where these tasks are well separated and easy to isolate in order to study their performance. This would also allow us to experiment hybrid solutions combining subtask solutions of different methods (e.g., the TAGME spotter with the WikiMiner disambiguation algorithm).

We developed **Dexter** [3], an entity annotator framework, containing several utilities to manage the Wikipedia dump,

[6] `http://www-nlp.stanford.edu/software/CRF-NER.shtml`

	Referent Graph			TAGME			WikiMiner		
	ρ^{MW}	$\rho^{\lambda\mathsf{MART}}$	ρ^{GDBT}	ρ^{MW}	$\rho^{\lambda\mathsf{MART}}$	ρ^{GDBT}	ρ^{MW}	$\rho^{\lambda\mathsf{MART}}$	ρ^{GDBT}
$P@1$	0.59	$0.68_{+15\%}$	$0.74_{+25\%}$	0.78	0.81 $_{+4\%}$	0.80 $_{+3\%}$	0.78	$\mathbf{0.86}_{+10\%}$	0.83 $_{+6\%}$
$P@5$	0.51	$0.62_{+22\%}$	$0.61_{+20\%}$	0.65	0.66 $_{+2\%}$	0.66 $_{+2\%}$	0.64	0.68 $_{+6\%}$	**0.69** $_{+8\%}$
$P@10$	0.44	$0.50_{+14\%}$	$0.51_{+16\%}$	0.50	0.50 $_{+0\%}$	0.51 $_{+2\%}$	0.50	0.51 $_{+2\%}$	**0.53** $_{+6\%}$
$iP_{r=0.10}$	0.76	$0.84_{+11\%}$	$0.87_{+14\%}$	0.87	0.89 $_{+2\%}$	0.89 $_{+2\%}$	0.88	**0.92** $_{+5\%}$	0.91 $_{+3\%}$
$iP_{r=0.50}$	0.55	$0.69_{+25\%}$	$0.70_{+27\%}$	0.67	0.68 $_{+1\%}$	0.69 $_{+3\%}$	0.66	$0.73_{+11\%}$	$\mathbf{0.77}_{+17\%}$
$NDCG$	0.64	0.70 $_{+9\%}$	$0.72_{+13\%}$	0.68	0.69 $_{+1\%}$	0.69 $_{+1\%}$	0.66	0.72 $_{+9\%}$	$\mathbf{0.75}_{+14\%}$
MRR	0.73	$0.81_{+11\%}$	$0.84_{+15\%}$	0.87	0.89 $_{+2\%}$	0.89 $_{+2\%}$	0.87	**0.92** $_{+6\%}$	0.90 $_{+3\%}$
$NDCG@5$	0.55	$0.67_{+22\%}$	$0.68_{+24\%}$	0.72	0.74 $_{+3\%}$	0.73 $_{+1\%}$	0.71	0.76 $_{+7\%}$	**0.77** $_{+8\%}$
$NDCG@10$	0.57	$0.68_{+19\%}$	$0.70_{+23\%}$	0.70	0.70 $_{+0\%}$	0.71 $_{+1\%}$	0.69	0.73 $_{+6\%}$	**0.75** $_{+9\%}$
$Recall$	0.76	**0.77** $_{+1\%}$	**0.77** $_{+1\%}$	0.68	0.69 $_{+1\%}$	0.69 $_{+1\%}$	0.64	0.70 $_{+9\%}$	$0.75_{+17\%}$
$Rprec$	0.46	$0.58_{+26\%}$	$0.60_{+30\%}$	0.56	0.58 $_{+4\%}$	0.58 $_{+4\%}$	0.56	0.60 $_{+7\%}$	$\mathbf{0.64}_{+14\%}$

Table 4: Entity Linking performance

a spotter based on the anchors and titles extracted from the dump, and data structures for retrieving all the features used by the annotators. Unlike WikiMiner, our framework does not rely on an external database to store the labels. In addition, during the execution it can maintain the model either on the disk or in main memory to improve performance. The framework runs also on normal hardware, since we exploit efficient data structures in order to maintain compressed data in main memory. The dataset we used, the source code and more detailed informations can be found at this address: `dexter.isti.cnr.it`.

We implemented WikiMiner, Referent Graph and TAGME in our framework, in order to verify if our relatedness function is able to improve the annotator performance. During the implementation, we slightly modified WikiMiner and TAGME: in WikiMiner we decided to rank the entities using a linear combination of commonness, link probability, and average relatedness with the context (the authors employed a classifier trained with several features that were heavy to retrieve); in TAGME we relied on our spotter that returns all the possible spots detected in the text, while in the original version the authors employ a specific policy for deleting spots in case of overlaps (we remove overlapping annotations at the end of the process, relying on the final ranking of the entities). We set the commonness threshold to 0.03 and we discard spots with link probability lower than 0.02.

Note that we are not interested in the absolute entity-linking performance of WikiMiner, TAGME, and Referent Graph, but rather on how the relatedness function impacts on the disambiguation process. For this reason, we implemented all the three algorithms within the same framework, and thus providing them with the output of the same spotter. For the same reason, the results of the Web services implementation of WikiMiner and TAGME are not reported. Those services use a different dump of Wikipedia, which is processed in a different way (e.g., tokenization, etc.), and they exploit a slightly different spotting algorithm, and this makes such results non significant within the scope of this work. However, it is important to report that we observed that our implementation always improves over the WikiMiner online service, and that it behaves only slightly worse then TAGME after the top 5 results, probably due to a different processing of Wikipedia.

We compared the results obtained by embedding different implementations of ρ: ρ^{MW}, $\rho^{\lambda\mathsf{MART}}$, and ρ^{GDBT}. Note that by embedding ρ^{MW} we are replicating the original algorithms that we consider as baselines to evaluate our proposed relatedness function.

The quality of the resulting algorithms is evaluated with the usual Precision@k ($k = 1, 5, 10$), Recall, and NDCG measures. We also report the interpolated precision at a certain recall cutoff r, iP_r with $r = 0.1$ and $r = 0.5$, the Mean Reciprocal Rank MRR and the Precision after R documents have been retrieved, where R is the total number of relevant entities for the document ($RPrec$).

We remind that in this evaluation we want to evaluate the number of correctly annotated entities *for a given document*; the evaluation is not spot-based, but we are rather considering the entity linking process as a whole, and its goodness on the full document.

The test dataset adopted is the same as the one of previous experiment, meaning that there is no overlap among the documents used for training the function ρ, and the documents used to evaluate its impact on the entity annotation process.

Table 4 reports the performance of the three annotators: for each annotator we show the performance using the original ρ^{MW} relatedness function, and then the effects of replacing the relatedness function with our learned relatedness $\rho^{\lambda\mathsf{MART}}$ and ρ^{GDBT}. The performance improvement given by the trained functions is significant:

Referent Graph. The proposed functions improve the ranking of results, in particular if we annotate only one entity per document using the ρ^{MW} relatedness only the 59% is correctly annotated, while with ρ^{GDBT} the percentage of correct documents is 74%. The relatedness function also reinforces the correct entities, improving the final ranking on the top entities as showed by the $NDCG$ measure which exhibits from a 14% up to a 25% of performance gain;

WikiMiner. ρ^{GDBT} improves both recall and $NDCG$, with gains superior to 10%. In both ρ^{GDBT} and $\rho^{\lambda\mathsf{MART}}$ the entity annotated with the largest confidence is correct in more than the 80% of the documents, with a improvement of 6% (ρ^{GDBT}) and of 10% ($\rho^{\lambda\mathsf{MART}}$) with respect to ρ^{MW};

TAGME. Recall, $NDCG$, and precision exhibit a positive improvement (from 1% up to 4%). The reader will note that ρ^{GDBT} and $\rho^{\lambda\mathsf{MART}}$ does not improve TAGME in the same measure as the other annotators: this is not surprising because the TAGME annotator is designed

to manage short texts, and relies less on the relatedness and more on the commonness.

In general, the best result quality was obtained using the ρ^{GDBT} function.

6. CONCLUSIONS

In this work, we have proposed a machine learning based approach aimed at discovering the entity relatedness function that can better support the entity linking task. We illustrated some of the properties that such function should preserve, and we presented a simple method to generate a training set form a collection of document human assessed entity linked documents. We casted the problem of discovering a suitable entity relatedness function into a learning to rank formulation. Our proposed approach is thus able to learn how to wisely blend the available features to generate a good entity relatedness function. We demonstrated that by exploiting our framework it is possible to better estimate the relatedness of two entities, and to compare and improve the performance of different state-of-the-art entity linking algorithms.

The proposed framework opens up a wide spectrum of improvement opportunities. In particular, We plan to investigate the tread-off between feature computational cost and benefit, and to embed our machine learned relatedness function into other entity-based tasks.

Acknowledgements

This work was partially supported by the EU projects InGeoCLOUDS (no. 297300), MIDAS (no. 318786), E-CLOUD (no. 325091) and the Regional (Tuscany) project SECURE! (FESR PorCreo 2007-2011).

7. REFERENCES

[1] M. Bron, K. Balog, and M. de Rijke. Ranking related entities: components and analyses. In *Proceedings of CIKM*, 2010.

[2] D. Ceccarelli, S. Gordea, C. Lucchese, F. M. Nardini, and R. Perego. When entities meet query recommender systems: semantic search shortcuts. In *Proceedings of SAC*, 2013.

[3] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani. Dexter: an open source framework for entity linking. In *Proceedings of the Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR)*, 2013.

[4] S. Chakrabarti, S. Kasturi, B. Balakrishnan, G. Ramakrishnan, and R. Saraf. Compressed data structures for annotated web search. In *Proceedings of WWW*, 2012.

[5] R. Cilibrasi and P. Vitanyi. The google similarity distance. *Knowledge and Data Engineering*, 2007.

[6] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL*, 2007.

[7] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *Proceedings of COLING*, 2010.

[8] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of CIKM*, 2010.

[9] J. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Statist*, 2001.

[10] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of IJCAI*, 2007.

[11] X. Geng, T.-Y. Liu, T. Qin, and H. Li. Feature selection for ranking. In *Proceedings of SIGIR 2007*, 2007.

[12] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of SIGIR*, 2011.

[13] J. Hoffart, S. Seufert, D. B. Nguyen, M. Theobald, and G. Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of CIKM*, 2012.

[14] J. Hoffart, M. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *Proceedings of EMNLP*, 2011.

[15] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 2002.

[16] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD*, 2002.

[17] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of CIKM*, 2007.

[18] D. Milne and I. H. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *In Proceedings of AAAI*, 2008.

[19] D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proceedings of CIKM*, 2008.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[21] P. Pantel and A. Fuxman. Jigs and lures: Associating web queries with structured entities. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011.

[22] W. Shen, J. Wang, P. Luo, and M. Wang. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of WWW*, 2012.

[23] R. van Zwol, L. Garcia Pueyo, M. Muralidharan, and B. Sigurbjornsson. Ranking entity facets based on user click feedback. In *Semantic Computing (ICSC)*. IEEE, 2010.

[24] G. Weikum and M. Theobald. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of PODS*, 2010.

[25] Q. Wu, C. Burges, K. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Inf. Retr.*, 2010.

[26] M. Yosef, J. Hoffart, I. Bordino, M. Spaniol, and G. Weikum. Aida: An online tool for accurate disambiguation of named entities in text and tables. *Proceedings of the VLDB Endowment*, 2011.