A Piggyback System for Joint Entity Mention Detection and Linking in Web Queries

Marco Cornolti, Paolo Ferragina
University of Pisa, Italy
{cornolti,ferragina}@di.unipi.it

Massimiliano Ciaramita
Google, Switzerland
massi@google.com

Stefan Rüd, Hinrich Schütze
University of Munich, Germany
inquiries@cis.lmu.org

ABSTRACT

In this paper we study the problem of linking open-domain web-search queries towards entities drawn from the full entity inventory of Wikipedia articles. We introduce SMAPH-2, a second-order approach that, by piggybacking on a web search engine, alleviates the noise and irregularities that characterize the language of queries and puts queries in a larger context in which it is easier to make sense of them. The key algorithmic idea underlying SMAPH-2 is to first *discover a candidate set of entities* and then *link-back those entities to their mentions* occurring in the input query. This allows us to confine the possible concepts pertinent to the query to only the ones really mentioned in it. The link-back is implemented via a collective disambiguation step based upon a supervised ranking model that makes one joint prediction for the annotation of the complete query optimizing directly the F1 measure. We evaluate both known features, such as word embeddings and semantic relatedness among entities, and several novel features such as an approximate distance between mentions and entities (which can handle spelling errors). We demonstrate that SMAPH-2 achieves state-of-the-art performance on the ERD@SIGIR2014 benchmark. We also publish GERDAQ (General Entity Recognition, Disambiguation and Annotation in Queries), a novel, public dataset built specifically for web-query entity linking via a crowdsourcing effort. SMAPH-2 outperforms the benchmarks by comparable margins also on GERDAQ.

Keywords

Entity linking, query annotation, ERD, piggyback

1. INTRODUCTION

As conversational interfaces become more popular in web applications, interaction increasingly resembles natural language dialogue and natural language understanding becomes a key problem. A deeper level of semantic understanding is necessary for improved precision, contextualization, and personalization of information exchange in ubiquitous computing devices via natural language. One of the techniques that has considerable traction to ground language with respect to semantic representations is *entity linking* to large repositories of structured knowledge such as Wikipedia or Knowledge Graphs. The quality of entity detection is a crucial first step towards the development of reliable and robust query processing, representation and understanding algorithms (see e.g.[10, 27, 28, 38, 43]).

Entity linking in queries is quickly emerging as a novel algorithmic challenge [9] that faces two difficult issues: (i) the noisy language of queries, characterized by misspellings, unreliable tokenization, capitalization and word order, and (ii) their brevity, as queries typically consist of just a few terms. Issue (i) has the main effect of degrading the coverage of the string-to-entity mapping, which is a primary component of entity annotators and is typically generated from well-edited texts such as Wikipedia. Issue (ii) affects the availability of context that can be leveraged to assist the disambiguation of the spotted entity-mentions. As a consequence, the "coherence" estimates that are typically implemented by known entity annotators to detect the most pertinent entities (see e.g. [14, 17, 20, 32]) are much less reliable when applied to queries rather than to well-formed documents, such as books, blog posts or news.

Nowadays, when people face an unknown word, phrase or query, they often search for it on search engines. In the best case, they will receive a definition or other direct result; in other cases, they obtain something else that is valuable: a *context* for the query. Partially inspired by this process, we propose to deal with the challenges posed by queries by piggybacking on web search engines. The intuition behind the *piggyback* approach, first introduced in [34], is that search engines can be viewed as the closest available substitute for the world knowledge that is required for solving complex natural language understanding tasks.

Search engines tend to be robust to the issues raised by queries because they have been designed to deal effectively with surface problems like misspellings, tokenization, capitalization and word order; e.g., a query such as `armstrog mon landign` is resolved to `armstrong moon landing`. They also provide additional context, in the form of a sequence of *snippets* showing the query keywords in bold form in their occurring documents. These snippets are ranked by relevance, with respect to the query, by means of sophisticated algorithms that leverage huge indexed document collections (the whole web), link graphs and log analysis.

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2883061

Our starting point is the SMAPH system, winner of the web queries track at the ERD Challenge, hosted by SIGIR 2014 [9], which dealt with the problem of annotating queries with sets of Wikipedia named entities via a piggyback approach. We perform a thorough experimental analysis of SMAPH, which allows us to identify, among others, a limitation related to the abundance of false negatives (leading to low recall). While investigating this problem, we also generalize the task to the more challenging problem of fully annotating queries not only with entities (the so-called C2W task in [8]) but also with their linking mentions (the so-called A2W task in [8]). Furthermore, in this paper we consider not only *named entities* (NEs) but also *general concepts*, which has been recognized as a key feature of modern query-annotation tools [10].

We evaluate our systems on two datasets: (i) GERDAQ, a novel dataset specifically built for the A2W problem, via an extensive crowdsourcing effort, that we have released to encourage and support reproducible scientific research on this problem; and (ii) the dataset employed for the ERD challenge. On GERDAQ, we show that our system, SMAPH-2, achieves effective results on the A2W problem with respect to strong baselines. On ERD, we show that SMAPH-2 is the state-of-the-art by improving upon ERD's winner and all other annotators that were proposed and tested on ERD after the challenge (i.e. since August 2014).

Technically speaking, SMAPH-2 deals with the A2W problem in two main phases. In the first phase we deploy the piggyback approach, combined with a robust text annotator designed for short texts (TagME-WAT [31]), to extract candidate entities from the snippets resulting from an input query. This alleviates the language noise and brevity of queries by offering a longer and cleaner context. In the second step, we adopt a novel *link-back* approach that prunes the set of candidate entities and keeps only those more likely to actually be mentioned in the query. The main idea behind link-back is to try to connect via approximate matching and entity-coherence optimization the predicted candidate entities to portions of the query. We propose two implementations of link-back: a simpler one, SMAPH-S, that deals with each mention-entity pair individually, and a more sophisticated one, SMAPH-2, that jointly evaluates all mention-entity pairs. Both approaches rely on supervised learning, based on both known and novel features. SMAPH-S trains a regressor on single annotations and uses a boolean response variable indicating whether the annotation is correct; SMAPH-2 is trained to rank the full set of annotations for the whole query (called *bindings*), and uses as response variable the F1 measure between the binding and the gold standard for that query.

The paper's main contributions are the following:

- We investigate for the first time the *A2W problem for open-domain web queries*, stated as the problem of identifying all entities and their mentions in a query.
- We release to the public the GERDAQ evaluation dataset for A2W. GERDAQ contains 1000 well-curated queries that have been labeled via a two-phase crowdsourcing process. GERDAQ meets the highest standards in terms of precision and recall even though A2W is a complex annotation task. This is the first dataset for entity-linking in queries for which the construction process is fully documented.
- We design a state-of-the-art query annotator, SMAPH-2, for the A2W problem on web queries. SMAPH-2 uses a learning-to-rank model that jointly predicts the best complete A2W annotation for the input query. The ranking crucially relies on a novel set of features we have designed for A2W that model the probability that a candidate entity is actually mentioned in the query, and thus should be "linked back" to it.
- In contrast to prior work, learning involves direct optimization of F1, the measure of evaluation.
- We present an extensive set of experiments that evaluate SMAPH-2 on two benchmarks, the ERD@SIGIR2014 benchmark and our novel dataset GERDAQ, and show that it achieves state-of-the-art performance on both.

2. RELATED WORK

The focus around entity annotators has increased significantly in the last few years, with respect to several knowledge bases such as DBpedia, Freebase and Yago [38, 28, 24, 14, 20, 32, 15, 29].

There is little prior work on entity annotators for queries, mostly concerned with the detection of NEs [30], possibly associated to *intent* [25] or pre-defined classes [27, 16, 13], POS tagging or tagging with a limited number of classes and other linguistic structures [1, 2], abbreviation disambiguation [41], assigning a coarse-grained purpose to each segment [23]. Some of them operate just on the query text (see e.g., [21]), others use query logs [1], click through information [26], search sessions [11], top-k snippets and web phrase DBs [2, 18], and large manually annotated collections of open-domain queries to extract robust frequency or mutual-information features and contexts [13].

Another interesting line of research that is related to query segmentation has the goal of identifying important phrases and compound concepts, like `new york times`, as indivisible segments of a query. The search engine can exploit such hints to increase result precision since documents that do not contain a segment's words in proximity or even the exact same order can be discarded (see e.g. [33, 22, 39]). More recently, a series of papers by Hagen et al. (see e.g. [18]) proposed simple and effective scoring functions and showed via a large experimental test that Wikipedia-page titles are very effective in query segmentation. In contrast to our work, these papers dealt with query segmentation only.

The paper closest to our study is [3], in which a fast and space-efficient entity-linking method leveraging information from query logs and anchor texts was designed in order to solve a *ranking version* of the A2W problem where entity-mention pairs are scored and ranked. The system was evaluated on the Webscope L24 (Yahoo Search Query Log To Entities) dataset by concentrating on entities only, and not considering mentions (i.e., a sort of *ranked C2W*). This way the authors did not determine the final annotation for a query but, rather, a ranked list of (possibly many) entities which was then evaluated by means of typical ranking metrics. The authors did not evaluate their system on ERD and against the annotators participating in ERD. In our paper, instead, we propose and evaluate a solution to the A2W problem and experiment in our design with some of the features proposed in [3], showing that other features we will introduce in this paper are more effective. We notice also that all of our features are reproducible, e.g., via publicly

available search engine APIs, while some features of [3] are not because they are derived from query logs.

A few joint mention-entity machine learning approaches have been proposed before [36, 37]. Sil and Yates [36] generate candidates with independent NER and entity linking base systems, then re-rank joint candidates with a linear maximum entropy model trained to optimize the L2-regularized conditional log likelihood on a training set of documents. Guo *et al.* [37] perform joint mention detection and entity linking on tweets. They generate candidates on n-grams with a base linking model and represent with a binary vector the entity mentions predicted. Learning is cast as a structured prediction task, via SVM, using the Hamming distance between the predicted and gold vector as the loss function. We instead propose to optimize directly F1, which is the final evaluation metric, in a significantly simpler formulation of the task.

The piggyback approach to language understanding was first proposed in [34], where it was successfully applied to the task of named entity recognition (NER) in web queries.

Our systems use TagME-WAT for candidate generation and the AIDA system as a baseline. We recall briefly their main features and refer the reader to the literature for details:

AIDA searches the input text for mentions using the Stanford NER tagger and adopts the YAGO2 knowledge base as catalog of entities. Disambiguation comes in three variants: PriorOnly (disambiguation to the most-common entity), LocalDisambiguation (independent per-mention disambiguation), CocktailParty (global disambiguatation by maximizing the coherence among annotations with a graph-based approach).

TagME searches the input text for mentions defined by the set of Wikipedia page titles, anchors and redirects. Disambiguation exploits the structure of the Wikipedia graph, according to a *voting scheme* based on a *relatedness measure* that takes into account the amount of common incoming links between two pages. The new and improved version of TagME, named WAT [31], follows the main algorithmic structure of TagME with two notable changes: it uses Jaccard-similarity between two pages' in-links as a measure of relatedness, and PageRank to sort the candidate entities that may annotate a mention.

3. PROBLEM STATEMENT

Let $q = t_1 t_2 \cdots t_n$ be a *query*, namely a sequence of n terms. An *entity* is identified by a Wikipedia page that describes its underlying unambiguous concept. Entities can be *named entities* (e.g., *New York City*, *Ian Murdock*) or *general concepts* (e.g., *City*, *Toyota Corolla*, *Peace*). A *mention* is a contiguous sequence of terms in q, say $t_b t_{b+1} \cdots t_e$, that refers to an entity. An *annotation* is a mention-entity pair and is represented by a triple (m_b, m_e, e), where m_b/m_e are the beginning/end of a mention and e is the entity it refers to. As an example, we can hypothesize two annotations in query $q =$ `armstrong moon landign` (note the typo). The first annotation is $a_1 = (1, 1, Neil\_Armstrong)$; it indicates[1] that the mention $t_1 =$ `armstrong` is linked to the Wikipedia page dedicated to the astronaut Neil Armstrong. The second annotation is $a_2 = (2, 3, Moon\_Landing)$ indicating that the substring $t_2 t_3 =$ `moon landign` refers to the event of landing on the moon.

A *query binding* for a query q is a set of annotations, that is, a set of non-overlapping mentions and the entities they refer to. The following example shows two possible bindings for query $q =$ `armstrong moon landign`:

$$b_1 = \{(1, 1, Neil\_Armstrong), (2, 3, Moon\_Landing)\};$$

$$b_2 = \{(1, 1, Louis\_Armstrong), (2, 2, Moon)\}$$

The first query binding b_1 identifies the first term as a mention of the astronaut "Neil Armstrong", and the last two terms as a mention of the "Moon landing" topic. The second binding b_2 identifies the first term as a mention of the jazz musician "Louis Armstrong", and the second mention as one single term referring to the entity "Moon".

Our goal is to design effective algorithms that find the most pertinent binding for a query q (in the example, b_1 rather than b_2). As discussed above we follow [8] and distinguish between two entity linking tasks, now formulated over queries rather than texts: (i) the C2W Task aims at detecting only the entities referred to in q without identifying their mentions; (ii) the A2W Task aims at detecting the full binding of q consisting of mentions and the entities they refer to. The former task was the one addressed by the participants of the 2014 ERD Challenge, the latter is the one typically addressed in (short and long) texts.

Queries can be inherently ambiguous. For example, `apple price` may refer to the cost of Apple Inc.'s stocks or to the market price of apples. In case of ambiguous queries, we aim at the identification of the single most likely binding for the query according to human common sense.

4. THE GERDAQ DATASET

In this section we describe the process that led to the creation of our query-annotated dataset GERDAQ. The queries of this dataset have been sampled from the KDD-Cup 2005 competition dataset, which consists of 800,000 queries. First we polished the dataset by discarding the queries that looked like web addresses (i.e., those containing `www` or `http`), then we randomly sampled 1000 queries according to a distribution that reflected their word-count length.

Guaranteeing a well-curated query annotation process, consisting in the detection of mentions and their entities, is a difficult process even for humans. Queries are short, possibly misspelled and otherwise malformed; query content could be unknown to a worker; queries are often ambiguous thus inducing different humans to link multiple entities for the same mention. These issues typically lead to noisy, incorrect, sparse, sometimes multiple annotations that had to be properly curated and reconciled in order to generate a robust and reliable dataset over which query annotators can be trained and tested reliably.

In order to achieve this goal we performed the annotation involving many human raters (aka workers) in a crowdsourcing environment, `CrowdFlower`, in two phases, called *recall-oriented* and *precision-oriented*, detailed below.

4.1 Phase 1: Recall-oriented

Given a query, this phase aims at finding a wide range of annotations a query may contain, focusing on coverage rather than on precision.

[1]The string *Neil_Armstrong* is the short version for the Wikipedia URL identifier en.wikipedia.org/wiki/Neil_Armstrong.

Partition	Queries	Queries with at least one annotation	Avg. Annotations per non-empty query	Avg. Annotations per query	Avg. query length (chars)
Training	500	446	2.10	1.86	25
Development	250	221	2.05	1.81	22
Test	250	222	1.95	1.73	23

Table 1: GERDAQ dataset statistics. We use "GERDAQ train", "GERDAQ dev" and "GERDAQ test" to refer to the three splits in the paper.

For each query we asked the workers to spot annotations in queries, namely both the mention and a pertinent entity the mention refers to (see Section 3). Workers were instructed to make sure that the concept they were thinking of for a mention was actually the one described by the Wikipedia page they were going to link. Workers were asked for each query to spot as many mentions of Wikipedia entities as they could and link them to Wikipedia URLs. Mentions were verified to be actual substrings of the query and URLs were verified to be existing Wikipedia articles. Each worker's contributions were tested against 70 randomly interspersed gold-standard queries to identify and exclude unreliable workers.

Each query was processed by at least 10 workers to guarantee a robust annotation process given its difficulty. The job, which completed in a few hours, collected a total of 10,038 trusted judgments. A judgment is a set of annotations spotted by a worker for a query. The workers found a total of 3,197 distinct annotations (3.2 per query). A total of 271 workers took part in the job; they processed 37 queries each on average.

1	2	3	4	5	6	7	8	9	≥ 10
1048	384	291	229	215	190	189	234	218	199

Table 2: Distribution of judgments over the 3,197 distinct annotations. Read the first column as "1,048 annotations were found by a single worker"

Table 2 shows the distribution of how many workers spotted the same annotation: about half of the annotations were found by one or two workers, but many of these rarely found annotations were actually correct and valid (see Table 3).

4.2 Phase 2: Precision-oriented

Since no worker is expert on all subjects, they can produce wrong annotations or miss correct ones. Thus, we designed a second phase for discarding bad annotations from Phase 1. We created a second job on CrowdFlower, asking workers who did not take part in the first job to rate, on a scale from 1 to 10, the likelihood that an annotation found in Phase 1 was correct. Workers were prompted with questions like: *In the query* `armstrong moon`*, how likely does* `armstrong` *refer to the entity "Neil Armstrong"?* Workers were also provided with an abstract of the candidate Wikipedia page (e.g. *Neil Alden Armstrong was an American astronaut...*), to better tell appropriate entities from wrong ones.

Each query was processed by at least 3 workers, for a total of 390 workers who processed 26 queries each on average, and collected a total of 9,612 annotation scores. Table 3 shows the distribution of average scores assigned to an annotation by workers, normalized in $[0, 1]$. We notice that many annotations have a high average score, and thus can be assumed to be highly pertinent to the query. Comparing these figures with Table 2 we notice that most annotations found in the previous phase by just one or two workers are actually pertinent because they get high scores in this second phase.

0	.1	.2	.3	.4	.5	.6	.7	.8	.9	1
53	81	94	134	169	251	325	369	633	712	375

Table 3: Distribution of average annotation scores. Example: 81 annotations received an average score in $[.1, .2)$. 375 annotations were given the maximum score by all workers (last column).

Two of the authors manually analyzed a sample of the scores assigned to annotations. They observed that annotations with scores smaller than 0.58 were all wrong and thus to be discarded; those with a score higher than 0.65 were all correct and thus to be kept; while those in between (a few dozens) were manually double checked for correctness until complete agreement between the two authors was reached. As cited above, queries can be inherently ambiguous, and this reflects into workers assigning a high score to two entities linked by the same mention. This happened for 90 mentions over a total of 2043. In such cases, we keep the entity with highest score, given that workers considered it the most probable meaning[2]. At the end the dataset consists of 2043 distinct annotations (2.0 annotations per query on average), this constitutes the final GERDAQ dataset (see Table 1 for basic statistics).

We randomly split GERDAQ into *training set* (500 queries), *development set* (250 queries), and *test set* (250 queries). We will make the dataset publicly available under a Creative Commons license[3]. We trained our systems on the *training set* and did parameter tuning, feature selection, and manual error analysis on the *development set*, keeping the *test set* only for the very final evaluation.

5. THE SMAPH SYSTEMS

In the present paper we discuss three query annotators that piggyback over search engines.

We first discuss SMAPH-1, which was designed to deal with the C2W problem (Section 5.1), and dig into some of its features and experimental results that were not presented in the paper for the ERD Challenge [9]. Then we present two new systems for solving the A2W Task: SMAPH-S and SMAPH-2. The first one is our first solution to the A2W Task (Section 5.2); it computes annotations of the input query by executing a *local* disambiguation approach. The second one, SMAPH-2, is our best solution to the A2W Task (Section 5.3) whose annotation process pivots on a *collective* disambiguation approach that jointly processes groups

[2]The GERDAQ dataset also features secondary meanings spotted by the workers, that are not considered in this paper, but are available for future work.

[3]Dataset at http://acube.di.unipi.it/datasets/

of mention-entity pairs in search of the best query binding among them.

5.1 SMAPH-1: The ERD Challenge system

SMAPH-1 piggybacks on the results returned by the public `API` of `Bing`, and works in three phases.

Fetching. The query q to be annotated is issued to the `Bing` search engine through its public `API`, enabling the *spelling correction* feature. This way results are not affected by spelling errors possibly present in the query. SMAPH-1 concentrates its analysis over the first 25 snippets returned by `Bing`. In addition to that, it concatenates q with the word `wikipedia`, issues it again to `Bing` and takes the top 10 results. The modified query boosts results from Wikipedia, without constraining the search to Wikipedia articles, which would return high-ranking articles loosely related to the actual query.

Candidate-entity generation. Entities are drawn from three sources. Wikipedia pages occurring in `Bing` results for query q form the set $\mathcal{E}_1$, and those occurring for query q+`wikipedia` form the set $\mathcal{E}_2$. A third, bigger set $\mathcal{E}_3$ of entities is found by annotating the top 25 snippets resulting from the search of q with the text annotator WAT[4] (see Section 3). This annotator has been designed to process short documents (e.g., tweets), and snippets are indeed excerpts of well-formed sentences of a few dozen terms. WAT finds mentions in the snippets and links them to Wikipedia entities, by exploiting the context provided by the snippet. For each snippet, WAT returns a set of annotations. We only keep the ones that overlap with a bold-highlighted substring of the snippet, as those substrings are the way in which query terms appear in web pages.

An analysis of the coverage of the three sources in Section 6.4 will show that this is a high-recall and pretty accurate source of candidate entities, having three clear advantages with respect to segmenting directly the query terms (cf. [18, 3]): (i) we have an implicit and automatic correction of the possible spelling errors in the query terms; (ii) the search engine "translates" query terms into forms as they appear in actual web pages; (iii) the snippet in which query terms occur provides a textual context for those terms that is then used by WAT to link those terms to entities.

As an example let us consider the query $q =$ `armstrong landing mon`, which presents a misspelling in the writing of the term `moon`, a swapping of the terms `moon` and `landing`, and an incomplete, ambiguous reference to *Neil Armstrong*. This query is hard for text annotators like WAT because it most likely will not find valid mentions and, thus, any annotation. However, `Bing` might return a snippet like `Immediately after` **Moon landing**, **Armstrong** `and Aldrin prepared the LM for liftoff.` Terms `mon` and `landing` are corrected, put in the right order. In a way, query terms are "translated" into natural language, since they are shown as they appear in an existing web page written by a human. The term `Armstrong` is present in the same context as the term `Aldrin`, which informs WAT that it refers to *Neil Armstrong* rather than *Louis Armstrong* or *Armstrong County, Pennsylvania*.

[4]During the development of SMAPH-1, we tried several other annotators other than WAT, but they yielded worse performance when annotating snippets.

Pruning. The last phase aims at pruning the entities in $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ that are not mentioned by q. To solve this task SMAPH-1 implements a binary classifier via LIBSVM with an RBF kernel [6]. The classifier was trained on GERDAQ aiming to maximize average-F1 on the development set. The entities of $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ surviving the classification process are returned as the result for q. At that time we used a set of features that took into account the coherence and robustness of the annotation process, the ranking and composition of snippets, the syntactic similarity between q and the snippets' bold portions, the syntactic similarity between q and the title of the candidate entity. Before exploring the features presented in Table 4, we need a few definitions:

- $U(q)$ is the list of URLs returned by `Bing` for query q. First URL is that of the highest-rank result;
- $\mathcal{C}(q)$ is the list of snippets returned by `Bing` for q (a sort of context of query terms). First snippet is that of the highest-rank result;
- $\mathcal{B}(q)$ is the multi-set of bold portions of all snippets returned by `Bing` for q;
- $\mathcal{W}(q)$ is the total number of web pages found by `Bing` for query q;
- $\mathcal{T}(e)$ is the Wikipedia-page title of entity e;
- $\mathcal{T}^*(e)$ is $\mathcal{T}(e)$ excluding the final parenthetical-string, if any. E.g. for $e = ER(TV series)$, $\mathcal{T}^*(e) =$ `ER`;
- $\mathcal{A}(s)$ is the set of annotations (mention-entity pairs) found by our auxiliary annotator WAT in snippet s overlapping with a bold portion of the snippet;
- $\rho(s, m, e)$ is the ρ-score indicating the confidence of the WAT-annotation (m, e) in snippet s [31];
- $lp(m)$ is the *link probability* of mention m [29], computed as the ratio between the number of times m is an anchor in Wikipedia divided by the number of all its occurrences in the Wikipedia pages;
- $comm(m, e)$ is the *commonness* of the annotation (m, e) [29], computed as the number of links in Wikipedia having m as anchor and pointing to the page of e, divided by the number of times anchor m appears in Wikipedia as a link to any page.
- $amb(m)$ stands for *ambiguity* and is the number of distinct Wikipedia pages that a mention m points to in the whole Wikipedia;
- $ED(x, y)$ is the Levenshtein distance between strings x and y, normalized by $\max(|x|_c, |y|_c)$, where $|x|_c$ stands for the number of characters of string x.
- $MinED(a, b)$ is an asymmetric measure of distance of string a towards string b, defined as follows[5]: let a_t and b_t be the set of terms in strings a and b, then

$$MinED(a, b) = \operatorname*{avg}_{t_a \in a_t} \left(\min_{t_b \in b_t} (ED(t_a, t_b)) \right)$$

  In other words, for each term in a, we find the closest term in b; $MinED(a, b)$ is the average distance between them.
- q^w is the query formulated by juxtaposing q and the term `wikipedia`.

[5]Throughout the paper, we use $\text{avg}_{x \in X} f(x)$ to indicate the arithmetic average of the results of function f applied to all elements of X, and $\text{avg}(X)$ to indicate the arithmetic average of the elements of X.

5.2 SMAPH-S: Local entity link-back

As we will see in the experimental section, most errors made by SMAPH-1 are false negative entities that appear as candidates, are explicitly mentioned in the query, but are assigned a low score due to their bad features and, thus, eventually discarded. This reflects into a result with high precision but low recall. To overcome this limitation, we decided to enforce the bond between candidate entities and the query terms that cite them, in order to come up with features that boost the score of entities cited by the query.

With this motivation, we decided to move our focus to the A2W problem, which forces us to think not only about the entities associated to a query, but also about the mentions (terms of the query) that refer to those entities. Our first step has been therefore to design a variant of SMAPH-1, called SMAPH-S, that enforces this by a process we call *linking-back*. The goal of this step is to match the candidate entities of the set $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ to the most appropriate mentions present in the input query. In this *link-back* process, some entities will be discarded because they cannot be linked to any mention in q. The final result is a set of full annotations (i.e., mention-entity pairs) of the input query.

The implementation of the link-back step employs a superset of the features employed by SMAPH-1 – which draw only from *entity* characteristics, see Table 4 –, including a set of new features that capture aspects of the *binding between mentions and entities* (see Table 5). We point out that the features listed in both tables are the result of a feature selection process from a larger set of features involving annotations, bold parts of snippets and entities. The description of features in Table 5 uses, in addition to the definitions introduced in Section 5.1, the following definitions:

- $\mathcal{F}(e, a)$ is the number of times (frequency) that entity e has been linked in Wikipedia pages by anchor a.
- $\mathcal{G}(e)$ is the set of anchors used in Wikipedia to link e.

SMAPH-S implements the link-back step with an SVM linear regressor trained to predict the likelihood that entity e is a pertinent concept for the mention m occurring in q. The regressor $\mathcal{R}$ is trained on GERDAQ train. For each gold-query q, let us denote with $Seg(q)$ the set of all possible segments in q (a segment is an n-gram of any length) and with $\mathcal{E}_q = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ the set of its candidate entities. The training examples for the regressor are all pairs $Seg(q) \times \mathcal{E}_q$, for all queries q in the dataset. A pair is considered a *positive* annotation iff it appears in the gold standard; all other pairs are considered to be *negative* annotations. Needless to say, the training examples are heavily unbalanced towards negative ones. We generate a feature vector for each annotation (m, e) by taking the features in Tables 4 (for entity e) and 5 (for annotation (m, e)). Feature *anchorsAvgED* is key: it is the average edit distance between mention m and all anchors that point to e in Wikipedia, which are possible ways to reference e. Edit distances are weighted wrt the number of times e is pointed to by an anchor (the square root mitigates the effect of high-frequency anchors). The more times e has been referenced by anchors similar to m, the lower *anchorsAvgED* will be.

In order to annotate a query q, SMAPH-S greedily selects its annotations in three steps: (i) it constructs the set $Seg(q) \times \mathcal{E}_q$, as done for the training; (ii) sorts all candidate annotations (m, e) in that set by decreasing score $\mathcal{R}(m, e)$; and (iii) scans the sorted candidate annotations and keeps (m_i, e_i) only if m_i does not overlap with any previously selected annotation, stopping whenever $\mathcal{R}(m_i, e_i)$ is lower than a threshold. The threshold is chosen in order to maximize the average-F1 on GERDAQ dev.

Drawn From All Sources		
ID	**Name**	**Definition**
1	$webTotal$	$\mathcal{W}(q)$
2	$isNE$	1 if e is a named entity, 0 otherwise. Based on Freebase as detailed in [5]

Drawn From Sources $\mathcal{E}_1$ and $\mathcal{E}_2$ (q^* is q for $\mathcal{E}_1$ or q^w for $\mathcal{E}_2$)		
ID	**Name**	**Definition**
3	$rank$	position of e's URL in $U(q^*)$
4	$EDTitle$	$MinED(\mathcal{T}(e), q^*)$
5	$EDTitNP$	$MinED(\mathcal{T}^*(e), q^*)$
6	$minEDBolds$	$\min\{MinED(b, q^*) \ : \ b \in \mathcal{B}(q^*)\}$
7	$captBolds$	number of capitalized strings in $\mathcal{B}(q^*)$
8	$boldTerms$	$(1/\lvert\mathcal{B}(q^*)\rvert) \sum_{b \in \mathcal{B}(q^*)} \lvert b \rvert$

Drawn From Source $\mathcal{E}_3$		
ID	**Name**	**Definition**
9	$freq$	$(\lvert s \in \mathcal{C}(q) \ : \ (\cdot, e) \in \mathcal{A}(s)\rvert)/\lvert\mathcal{C}(q)\rvert$
10	$avgRank$	$(\sum_{i \in [0,25)} p_i)/25$ where $p_i = \begin{cases} i & \text{if } (\cdot, e) \in \mathcal{A}(\mathcal{C}(q)_i) \\ 25 & \text{otherwise} \end{cases}$
11	$pageRank$	PageRank of e in Wikipedia Graph
		P $:= \{\rho(s, m, e) : (m, s) \in X(q)\}$;
12	ρ_{min}	min(P)
13	ρ_{max}	max(P)
14	ρ_{avg}	avg(P)
		$\mathcal{L} := \{lp(m) : (m, s) \in X(q)\}$;
15	lp_{min}	min($\mathcal{L}$)
16	lp_{max}	max($\mathcal{L}$)
		$C := \{comm(m, e) : (m, s) \in X(q)\}$
17	$comm_{min}$	min(C)
18	$comm_{max}$	max(C)
19	$comm_{avg}$	avg(C)
		$A := \{amb(m) : (m, s) \in X(q)\}$;
20	$ambig_{min}$	min(A)
21	$ambig_{max}$	max(A)
22	$ambig_{avg}$	avg(A)
23	$mentMED_{min}$	$\min(MinED(m, q) : (m, s) \in X(q))$
24	$mentMED_{max}$	$\max(MinED(m, q) : (m, s) \in X(q))$

Table 4: Features of a candidate entity e (used by SMAPH-1, SMAPH-S and SMAPH-2) for query q. Let $X(q) := \{(m, s) : s \in \mathcal{C}(q) \wedge (m, e) \in \mathcal{A}(s)\}$.

ID	Name	Definition
25	$anchorsAvgED$	$\frac{\sum_{a \in \mathcal{G}(e)} \left(\sqrt{\mathcal{F}(e, a)} \cdot ED(a, m)\right)}{\sum_{a \in \mathcal{G}(e)} \sqrt{\mathcal{F}(e, a)}}$
26	$minEdTitle$	$MinED(m, \mathcal{T}(e))$
27	$EdTitle$	$ED(m, \mathcal{T}(e))$
28	$commonness$	$comm(m, e)$
29	lp	$lp(m)$

Table 5: Features of a candidate annotation (m, e) (used by SMAPH-S and SMAPH-2), where m is the mention (list of query terms) and e is the entity.

The experimental section will evaluate the performance of SMAPH-S on GERDAQ and will show that relying on individual mention-entity pairs *one by one* to select the binding for the query q is too restrictive and results in low F1. This motivated the design of a *collective disambiguation* process that is the novel core of SMAPH-2, the best performing annotator we propose in this paper.

5.3 SMAPH-2: Joint entity link-back

Instead of judging each annotation individually, the last annotator carries out a collective analysis of all candidate annotations of the input query q, and searches for the binding that maximizes F1, the task's primary evaluation measure. The better a binding is (in terms of F1) the higher its predicted score should be. For this reason we implement a joint mention entity prediction model that predicts the F1 score for a full binding, the complete set of annotations for a single query. Considering the whole set of annotations, in one binding, we can design features that capture properties of the relation between multiple annotations and the input query (e.g., how many query terms are covered by the binding) and among annotations themselves (e.g., the semantic relatedness among their entities). Those features are described in Table 6. In SMAPH-2 we use also the features regarding single entities and annotations (Tables 4 and 5).

ID	Name	Definition
		$\mathrm{R} := \{rel(e_1,e_2) \mid e_1,e_2 \in E(b_q) \ \wedge \ e_1 \neq e_2\}$
30	rel_{min}	$\min(R)$
31	rel_{max}	$\max(R)$
32	$nTokens$	$\lvert q\rvert$
33	$covg$	$\sum_{(m,e)\in b_q}(\lvert m\rvert)/\lvert q\rvert$
34	$sumSegLp$	$\sum_{s\in\ Seg(q)} lp(s)$
35	$avgSegLp$	$\sum_{s\in\ Seg(q)} lp(s)/\lvert s \in\ Seg(q)\rvert$
36	$nBolds$	$\lvert\mathcal{B}(q)\rvert$
37	$nDisBolds$	$\lvert\{b\ :\ b\in\mathcal{B}(q)\}\rvert$
38	$minEdBlds$	$\sum_{b\in\mathcal{B}(q)} MinED(b,q)\rvert$

Table 6: Features of a candidate binding b_q for query q (used by SMAPH-2). E is a function that maps a binding to the entities it contains. $rel(e_1,e_2)$ is the relatedness among entities e_1 and e_2, measured by the Jaccard similarity of the sets of incoming links for e_1 and e_2. Recall also that $Seg(q)$ is the set of all possible segments of a query q (see Section 5.2).

5.3.1 Candidates enumeration

We first enumerate all possible segmentations of q using a BIO encoding, namely sequences of the symbols B-I-O of length $|q|$, which respectively denote the beginning, continuation and absence of a segment. These are $o(3^{|q|})$, as not all possible BIO sequences correspond to valid segmentations (a label I can only follow I or B). Furthermore, the query length is typically short. The final set of segmentations generated by the BIO sequences of q is called $\mathcal{BIO}_q$. For example, for query q =`armstrong mon lading` ($|q|=3$), the set $\mathcal{BIO}_q$ includes the segmentations (vertical bar "|" indicates segment truncation): `armstrong|mon|lading` (corresponding to BIO sequence `BBB`); `armstrong|mon lading` (`BBI`); `armstrong mon|lading` (`BIB`); `armstrong|lading` (`BOB`); `armstrong` (`BOO`); etc.

Complete binding candidates are then generated from each query q by taking each segmentation $G \in \mathcal{BIO}_q$ and assigning to each segment in it any possible entity in $\mathcal{E}_q = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ (with repetitions). This generates a total of $O(|\mathcal{BIO}_q| \times |q| \times |\mathcal{E}_q|)$ training examples for each query q, which are heavily unbalanced towards negative ones. We adopted a simple pruning heuristic to significantly confine the size of this set, so that the model can be trained in around 30 minutes. The heuristic is based on the following algorithm. It assigns, for each segmentation $G \in \mathcal{BIO}_q$, candidate entities drawn from $\mathcal{E}_q$ to segments $s \in G$ (in the algorithm, this set is C_s):

- For each pair $(s,e) \in G \times \mathcal{E}_q$, if $MinED(s, \mathcal{T}(e)) \leq 0.7$, add (s,e) to C_s;
- For each entity $e \in \mathcal{E}_q$ that has not been added in the previous step, add (s,e) to C_s for all $s \in G$.
- The set of candidate bindings generated for G is the Cartesian product $C_{s_1} \times \cdots \times C_{s_{|G|}}$.

The final set of candidate bindings for query q is the union of those Cartesian products for all $G \in \mathcal{BIO}_q$. This simple heuristic reduces the generated examples to an average of 495 bindings per query on GERDAQ train, and it keeps, for most queries, the candidates with highest F1.

5.3.2 Features

Each candidate binding

$$b = (m_1, e_1), (m_2, e_2), \cdots, (m_{|b|}, e_{|b|})$$

is associated with a feature vector $F(b)$ generated as follows: (i) features of Table 4, relative to entities, are computed over entities $e_1, \cdots, e_{|b|}$, and their maximum, minimum and average values for each entity are added to $F(b)$; (ii) features of Table 5 are computed over b's annotations and their maximum, minimum and average values for each annotation are added to $F(b)$; (iii) features of Table 6, relative to the candidate binding as a whole, are computed over b and added to $F(b)$. The final length of the feature vector $F(b)$ is three times (because of min, max and avg) the number of features in Tables 4 and 5 plus the number of features in Table 6, for a total of 96 features.

5.3.3 Learning

For each query q_i and its gold binding b_i in GERDAQ train, we generate all candidate bindings b_{ij} as described above. For each b_{ij} we generate $F(b_{ij})$, the features for candidate b_{ij}. Then we assign a response value to b_{ij} by computing the F1 (y_{ij} below) between b_i and b_{ij}.

This response has a number of desirable properties: by design $y_{ij} \in [0,1]$ since it is the F1 between the gold binding and a candidate binding, e.g., if $b_{ij} = b_i$, then $yij = 1$. The ranking function will learn to prefer the candidate binding that locally maximize F1. Notice also that if no gold binding is generated in the candidate list for q_i, we can still use the rest of the candidates to learn the ranking function as we are able to compute the F1 between any candidate binding and the gold one. In a classification approach, binary or structured, it would not be possible to learn from queries if the gold standard (the true label) is not among the candidates.

Thus, in this framework, it is quite simple to encode the data in a way that suits the problem in a principled way. Furthermore, we can use any off-the-shelf learning to rank library to learn a ranking function. In our case we used a boosted tree ranking model [42].

5.3.4 Best binding prediction

The annotation of a query q consists of four steps: (i) we generate the set of candidate entities $\mathcal{E}_q = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$, as described in Sect 5.1; (ii) we construct the set $\mathcal{BIO}_q$ of all BIO-sequences of q, as done above for training; (iii) we create the set $\mathcal{A}_q$ of candidate bindings by associating to each mention $s \in \mathcal{BIO}_q$ all entities in $\mathcal{E}_q$; the same heuristics employed at training time are adopted for pruning here; (iv) we rank the bindings in $\mathcal{A}_q$ by means of the learned ranker $\mathcal{R}'$ and pick the binding $\mathcal{A}^*(q)$ with the highest rank.

6. EXPERIMENTS

6.1 Evaluation metrics

We evaluate the performance of all query annotators with standard metrics based on precision (P), recall (R) and F1 [5, 8, 40]. We provide an example of the computation of the metrics for a single query. Let $q =$ `armstrong mon lading`, and let the corresponding gold standard binding for the A2W problem be:

$$\{(1, 1, Neil\_Armstrong), (2, 3, Moon\_Landing)\}$$

thus, the first term is a mention of the astronaut, while the second and third form a single mention of the "landing on the Moon" event. Let the following be a system's prediction:

$$\bar{b} = \{(1, 1, Neil\_Armstrong), (2, 2, Moon), (3, 3, Moon\_Landing)\}$$

Annotation $(1, 1, Neil\_Armstrong)$ is a true positive (TP); $(2, 2, Moon)$ and $(3, 3, Moon\_Landing)$ are false positives (FPs); $(2, 3, Moon\_Landing)$ is a false negative (FN). Hence, overall, TP=1, FP=2, FN=1, yielding P=1/3, R=1/2, and F1=2/5. We notice, in passing, that such F1 value would be the response assigned to the system's response $\bar{b}$ above, while training SMAPH-2. Overall, performance on a dataset is obtained by computing the values for TP, FP and FN over the whole dataset. We call these metrics *micro-{P,R,F1}*. As proposed in recent evaluation frameworks [5, 40], we also report the arithmetic average of Precision, Recall and F1, denoted as *average-{P,R,F1}*[6].

It is important to consider that queries sometimes contain no entities (in particular, named entities). The way these cases are accounted for can play a significant role in the final evaluation metrics. The ability of a system to not annotate entities where there are none is crucial. This aspect is better captured by average measures, while micro measures focus on the quality of retrieved entities/annotations.

6.2 The experimented annotators

In our experiments we tested the following annotators (for algorithmic details see the previous sections):

WAT is the improved version of TagME introduced in [31] for the A2W task (annotation detection). As relatedness function in the disambiguation process we used the Jaccard similarity among in-links, because it performed best on GERDAQ.

AIDA is the A2W annotator introduced in [20], we downloaded the code from the official web site[7]. AIDA offers several disambiguation methods, we tested all of them and found that they offer almost the same performance on GERDAQ, so we only report the best number.

NTNU-UiS is a query annotator for the C2W task (entity only detection), introduced in [19] that uses a multi-stage framework, first recognizing entity mentions, next scoring candidate entities using a learning-to-rank method, finally, using a greedy algorithm to find all valid interpretation sets for the query.

NTUNLP introduced for the C2W task in [7] searches the query trying to match freebase surface forms with the longest-match strategy. The disambiguation step is built on top of TagME and Wikipedia.

Seznam introduced for the C2W task in [12] uses Wikipedia and DBpedia to generate candidate annotations, than builds a graph of mentioned entities exploiting the link structure of Wikipedia. The disambiguation step is based on PageRank over this graph that assigns a score to each entity.

SMAPH-1 (Section 5.1) deals with the C2W task.

SMAPH-S (Section 5.2) is our first proposal for the A2W problem, derived from SMAPH-1, evaluates each mention-entity pair individually.

SMAPH-2 (Section 5.3) is our final annotator that deals with the A2W problem by evaluating annotation sets collectively.

The first two annotators (i.e., AIDA and WAT) are the baselines for the C2W and A2W problems, while SMAPH-S serves as a first step towards our best proposal for A2W, namely SMAPH-2. Other annotators employed here are currently (as of October 2015) the top-ranking annotators of the ERD Challenge.

6.3 Datasets

Our experiments have been conducted on two datasets:

ERD The dataset used in the ERD Challenge to test the annotators solving the C2W problem on NEs only [9]. It consists of 500 queries fully annotated with NEs drawn from Freebase. The ERD Challenge dataset is not available off-line. Systems can be tested by sending queries to the ERD Challenge platform to be annotated. The query's gold standard remains unknown, so it does not let one carry out any error analysis.This makes the evaluation against this dataset a real third-party check of the robustness of annotators.

GERDAQ This is the novel dataset we have built via CrowdFlower (Section 4), properly designed to test the query annotators on both the C2W and the A2W problem, including all possible entities of Wikipedia (hence, not just NEs).

A third public dataset, the Webscope L24 (Yahoo Search Query Log To Entities), features a set of annotated queries. Unfortunately its usage terms prevent us to experiment with it, since one of the authors of this paper is employed by a commercial entity.

6.4 Coverage of entity sources

We evaluated the coverage of the three entity sources introduced in Section 5.1 for the SMAPH systems. This is an important analysis, because it allows to quantify the impact

[6]These metrics are also sometimes referred to as *macro-{P,R,F1}*. If the gold binding of a query is the empty set, we define recall to be 1.0. If the proposed binding for a query is the empty set, we define precision to be 1.0.

[7]`http://www.mpi-inf.mpg.de/yago-naga/aida/`

of various signals employed in the design of those sources, and their complementarity. Table 7 reports the coverage and precision of each entity source. $\mathcal{E}_1$, Wikipedia pages appearing when searching q; $\mathcal{E}_2$, Wikipedia pages appearing when searching q + `wikipedia`; $\mathcal{E}_3$, entities found by annotating snippets. Plus their union. Source $\mathcal{E}_3$ is the largest single source of entities, though having small precision. Entities in $\mathcal{E}_1$ are included in $\mathcal{E}_2$: Wikipedia pages found by searching q are also found by searching q + `wikipedia`, hence $\mathcal{E}_1 \subseteq \mathcal{E}_2$. However, $\mathcal{E}_1$ has higher precision than $\mathcal{E}_2$. Merging all sources together adds to plain $\mathcal{E}_3$ a +2.7% coverage on all entities and +0.9% coverage of named entities. An attentive reader might notice that $\mathcal{E}_2 \cup \mathcal{E}_3$ reaches the highest coverage, and that there's no need of adding $\mathcal{E}_1$ to increase it, since $\mathcal{E}_1 \subseteq \mathcal{E}_2$. However, the fact that $e \in \mathcal{E}_1$ provides a stronger signal than $e \in \mathcal{E}_2$ about the likelihood of e being an entity pertinent to the query, and only by including $\mathcal{E}_1$ this signal is exploited. Coverage of $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ is an upper bound for the recall of the three SMAPH systems, as they all employ the three sources of candidate entities.

	$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_3$	$\mathcal{E}_3 \cup \mathcal{E}_1$	$\mathcal{E}_3 \cup \mathcal{E}_2$	$\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$
C_E	14.8	28.7	84.9	86.1	87.6	87.6
P_E	35.3	21.5	23.4	22.5	19.6	19.5
C_{NE}	26.6	41.9	92.7	93.5	94.4	94.4
P_{NE}	44.6	24.2	23.2	22.4	19.2	19.1

Table 7: Coverage (C) and precision (P) of the entity sources $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ on GERDAQ test. Top two rows report coverage and precision about all entities, bottom two rows are limited to Named Entities. C stands for coverage (how many gold entities are found), P stands for precision (how many entities found are gold). Read the leftmost column as "entity source $\mathcal{E}_1$ finds 14.8% of all gold entities and 26.6% of the gold named entities, with a precision of 35.5% among all entities and 44.6% among named entities".

6.5 Feature selection for entity disambiguation and pruning

The final set of features employed by the annotators were presented in Tables 4, 5 and 6, above. They are a subset of a wider number of features from which we discarded those that proved not to be effective via a feature selection process by ablation. Due to the lack of space, we do not report details on the excluded features, but we cite the most eminent ones.

Among the excluded features we mention the score provided by the models introduced in [3] for query entity linking, heavily based on word embeddings (i.e. `word2vec`). These models measure the similarity of the word embeddings of the query terms to the word embeddings of the first paragraph of the Wikipedia page that describes e. Though this score works well if used alone (see [3]), it does not add to our model additional information to decide whether an entity is pertinent for a query or not, because we already exploit a larger (and, experimentally, more effective) context for a query term given by the snippets in which the term appears. This context is then efficaciously used by the text-annotator WAT for entity disambiguation.

We plan to investigate more why it is the case that embeddings were not useful in the current setup, as it seems likely that they should help given that the problem is very high-dimensional at the source (query strings). A possibility is that it might be necessary to re-train the embeddings given the peculiar language of queries. At the same time, there is not much available public data to learn embeddings for queries. It might be possible to see if there is any benefit to this approach by training the embeddings on the rest of the KDD Cup queries.

6.6 Exp #1. NE-only detection (NE-only C2W task)

In this task the goal is to identify named entities mentioned by queries of the ERD and GERDAQ datasets. Some of the annotators we experiment with are designed to detect all entities (general and NEs) and, in some cases, their mentions; however, in this experiment we evaluate their ability to spot *named entities only, without considering the corresponding mentions.*

System	$F1_{avg}$
AIDA	22.1
WAT	58.6
Seznam	66.9
SMAPH-S	67.0
NTU	68.0
SMAPH-1	68.8
NTNU-UiS	69.9
SMAPH-2	**70.8**

Table 8: C2W over ERD dataset (average-F1).

Table 8 reports the average-F1 computed by the ERD online evaluation infrastructure. The table shows that WAT is superior to AIDA over the annotation of queries, but it is up to 10% worse than the winner of the ERD Challenge (SMAPH-1), which is in turn superseded by our new proposal SMAPH-2 by another +2% (absolute) in average-F1. We notice that after the completion of the ERD Challenge, other systems have been proposed and tested on the ERD platform. Systems such as NTNU-UiS and NTU have scored better results than the original SMAPH-1. Nonetheless, SMAPH-2 obtains again the top spot. In absolute terms, these figures show that queries are difficult to annotate. The F1-measure over queries obtained by WAT and AIDA is significantly lower, −16% absolute, than their F1's achievements on short texts [8]. Therefore the design of dedicated annotators for web queries is crucial in order to reach reasonable performance. Another interesting observation is that linking-back entities to mentions seems not useful if not properly implemented; e.g., by a joint full-query prediction approach. In fact, SMAPH-S performs worse even than SMAPH-1.

We also tested annotators on the queries available in the test portion of the GERDAQ dataset, restricting the evaluated entities to NEs only. Table 9 confirms the previous experiment: SMAPH-2 improves SMAPH-1 by about 3% (average-F1) and 5% (micro-F1). Since the ERD dataset's gold standard is unknown, with the GERDAQ dataset we can perform an in-depth analysis reported in the following table. The table shows, among other things, that WAT has higher entity recall than AIDA. This is possibly due to the latter depending on off-the-shelf APIs for preprocessing of named entity detection.

In this experiment, a significant portion of the queries have no gold entities attached, and this explains why micro measures are smaller than average measures.

Annotator	P_{avg}	R_{avg}	$F1_{avg}$	P_{mi}	R_{mi}	$F1_{mi}$
AIDA	94.8	59.6	58.4	31.6	4.8	8.3
TagME	75.4	83.3	63.2	52.1	49.6	50.8
WAT	70.3	85.3	64.4	42.8	66.9	52.2
SMAPH-1	85.5	82.7	74.5	62.0	62.5	62.2
SMAPH-S	82.6	82.3	73.1	58.7	59.7	59.2
SMAPH-2	85.8	84.5	**76.0**	65.5	62.9	**64.2**

Table 9: C2W over GERDAQ dataset (test portion) and NEs only.

6.7 Exp #2. Generic entity detection (C2W task)

This experiment is similar to Exp #1 but without any restriction on the kind of detected entities, which may be now all entities represented in Wikipedia, including generic concepts. Because of the features of the ERD dataset, we can perform this experiment only on GERDAQ, which also features general entities, and not just NEs.

Annotator	P_{avg}	R_{avg}	$F1_{avg}$	P_{mi}	R_{mi}	$F1_{mi}$
AIDA	94.0	12.2	12.6	28.6	1.5	2.8
TagME	60.4	51.2	44.7	52.7	49.6	51.1
WAT	49.6	57.0	46.0	43.0	56.4	48.8
SMAPH-1	77.4	54.3	52.1	58.5	54.0	55.9
SMAPH-S	64.8	56.2	51.4	57.0	54.7	55.8
SMAPH-2	72.1	55.3	**54.4**	64.1	51.3	**57.0**

Table 10: C2W results on GERDAQ test, all entities.

In comparison to the NE-annotation of the previous section, we notice that detecting generic entities is a harder problem. F1 decreases by about 7% (micro F1) and 22% (average F1). The significant decrease might be attributable to the fact that NEs are easier to detect because they are less ambiguous.

Again, text annotators such as AIDA and WAT are significantly worse than query annotators. SMAPH-S is again worse than SMAPH-1, but with a smaller gap, thus providing some credit for the usefulness of local link-back. SMAPH-2 is still the best entity annotator with a significant increase with respect to WAT of about 9% in average/micro F1 (absolute) and about 2% with respect to SMAPH-1.

6.8 Exp #3. Annotation detection (A2W task)

The goal of this experiment is to evaluate annotators over the most general scenario of the detection of all entities and their mentions. As in Exp #2, we will use here only GERDAQ, the only dataset that provides well-curated annotations on queries.

We first notice that on the general mention-entity annotation, F1 is lower than F1 achievable for C2W. In fact, Table 10 shows that the best average-F1 on C2W (i.e. 54.4% of SMAPH-2) decreases on A2W to 51.4%, thus confirming that the A2W problem is more difficult.

Again, off-the-shelf text annotators (AIDA and WAT) are worse than query annotators, but now SMAPH-S improves over them by about 3-4% in average/micro F1. SMAPH-2 is still the best annotator with a gap over SMAPH-S slightly larger on A2W than that observed in the C2W task; about +5% average-F1 and +3% micro F1. This enforces our previous observation on the importance of designing specific query annotators. We notice that link-back, even in the local-disambiguation approach, improves text annotators by at least 3%. The increased gap between SMAPH-S and SMAPH-2 in the A2W task, with respect to the C2W one, seems to provide solid evidence of the value of the simple joint entity mention detection and linking we have introduced in this paper.

Annotator	P_{avg}	R_{avg}	$F1_{avg}$	P_{mi}	R_{mi}	$F1_{mi}$
AIDA	94.0	12.2	12.6	28.6	1.5	2.8
TagME	58.4	49.7	43.0	50.3	47.9	49.1
WAT	47.2	54.3	43.6	40.3	53.0	45.8
SMAPH-S	59.5	50.6	46.3	51.0	49.4	50.2
SMAPH-2	68.4	52.3	**51.4**	59.9	47.9	**53.2**

Table 11: A2W results on GERDAQ test. Metrics based on Strong Annotation Match (namely, exact match on both entities and mentions).

7. CONCLUSIONS

In this paper we have investigated the problem of entity linking of open-domain web search queries. Specifically, we implemented and evaluated variants of piggyback models, a second-order approach in which web search engine results for the input query are analyzed to generate candidate mentions, candidate entities and features for prediction with machine learning. We designed a system, called SMAPH-2, that introduces two major novel algorithmic ideas: (i) a link-back approach that serves the purpose of validating the candidates generated by the piggyback step that otherwise tends to overgenerate and (ii) a simple and effective joint mention and linking approach based on learning to rank that optimizes directly the F1 metric between predicted and gold annotations over the full query. SMAPH-2 yields state-of-the-art results in the online ERD@SIGIR2014 challenge evaluation framework. Furthermore, we built and will share GERDAQ, a novel dataset we developed specifically for web-query entity linking via a crowdsourcing effort, and show that SMAPH-2 outperforms the benchmarks by comparable margins on GERDAQ.

Our study paves the way to other investigations that employ better annotations discovered by SMAPH-2 in several IR applications, such as query classification and clustering [4, 35], query expansion [10] and, possibly, query segmentation [18], e.g., via the mention-detection step which is robust with respect to misspelling and shuffling of query terms.

8. ACKNOWLEDGMENTS

This work, at the University of Pisa and at the Ludwig Maximilian University of Munich, was partially supported by two Google Faculty research awards, TAGME and PIGGYBACK. The University of Pisa group was also partially supported by the EU H2020 Program under the scheme "INFRAIA-1-2014-2015: Research Infrastructures" grant agreement #654024 "SoBigData: Social Mining & Big Data Ecosystem".

9. REFERENCES

[1] A. Alasiry, M. Levene, A. Poulovassilis. Detecting candidate named entities in search queries. In *SIGIR*, 1049–1050, 2012.

[2] M. Bendersky, W.B. Croft, D.A. Smith. Joint Annotation of Search Queries. In *ACL-HLT*, 1:102–111, 2011.

[3] R. Blanco, G. Ottaviano, E. Meij. Fast and Space-Efficient Entity Linking for Queries. In *WSDM*, 179–188, 2015.

[4] I. Bordino, G. De Francisci Morales, I. Weber, F. Bonchi. From Machu-Picchu to "rafting the urubamba river": anticipating information needs via the entity-query graph. In *WSDM*, 275–284, 2013.

[5] D. Carmel, M.-W. Chang, E. Gabrilovich, B.-J. Hsu, K. Wang. ERD 2014: Entity Recognition and Disambiguation Challenge. In *SIGIR Forum*, 2014.

[6] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. In *ACM Transactions on Intelligent Systems and Technology*, 27:1–27:27, 2011.

[7] Y.-P. Chiu, Y.-S. Shih, Y.-Y. Lee, C.-C. Shao, M.-L. Cai, S.-L. Wei, H.-H. Chen. NTUNLP Approaches to Recognizing and Disambiguating Entities in Long and Short Text in the 2014 ERD Challenge In *Workshop on Entity Recognition & Disambiguation* (ERD), hosted by SIGIR, 3–12, 2014.

[8] M. Cornolti, P. Ferragina, M. Ciaramita. A framework for benchmarking entity-annotation systems. In *WWW*, 249-260, 2013.

[9] M. Cornolti, P. Ferragina, M. Ciaramita, S. Rüd, H. Schütze. The SMAPH system for query entity recognition and disambiguation. In *Workshop on Entity Recognition & Disambiguation* (ERD), hosted by ACM SIGIR, 25–30, 2014.

[10] J. Dalton, L. Dietz, J. Allan. Entity query expansion using knowledge base links. In *SIGIR*, 365–374, 2014

[11] J. Du, Z. Zhang, J. Yan, Y. Cui, Z. Chen. Using search session context for named entity recognition in query. In *SIGIR*, 765–766, 2010.

[12] A. Eckhardt, J. Hreško, J. Procházka, O. Smrf. Entity linking based on the co-occurrence graph and entity probability. In *Workshop on Entity Recognition & Disambiguation* (ERD), hosted by SIGIR, 37–44, 2014.

[13] A. Eiselt, A. Figueroa. A Two-Step Named Entity Recognizer for Open-Domain Search Queries. In *IJNLP*, 829–833, 2013.

[14] P. Ferragina, U. Scaiella. Fast and accurate annotation of short texts with Wikipedia pages. *IEEE Software*, 29(1): 70-75, 2012. Also *CIKM* 2010.

[15] E. Gabrilovich, S. Markovitch. Wikipedia-based semantic interpretation for natural language processing. *J. Artif. Int. Res.*, 34(1):443–498, 2009.

[16] J. Guo, G. Xu, X. Cheng, H. Li. Named Entity Recognition in Query. In *SIGIR*, 267–274, 2009.

[17] Z. Guo, D. Barbosa. Robust entity linking via random walks. In *CIKM*, 499–508, 2014.

[18] M. Hagen, M. Potthast, A. Beyer, B. Stein. Towards Optimum Query Segmentation: In Doubt Without. In *CIKM*, 1015–1024, 2012.

[19] F. Hasibi, K. Balog, S. E. Bratsberg. A Greedy Algorithm for Finding Sets of Entity Linking Interpretations in Queries. In *Workshop on Entity Recognition & Disambiguation* (ERD), hosted by SIGIR, 75–78, 2014.

[20] J. Hoffart, M. A. Yosef, *et alii*. Robust disambiguation of named entities in text. In *EMNLP*, 782–792, 2011.

[21] A. Jain, M. Pennacchiotti. Domain-independent entity extraction from web search query logs. In *WWW*, 63–64, 2011.

[22] R. Jones, B. Rey, O. Madani, W. Greiner. Generating query substitutions. In *WWW*, 387âĂŞ-396, 2006.

[23] M. Joshi, U. Sawant, S. Chakrabarti. Knowledge Graph and Corpus Driven Segmentation and Answer Inference for Telegraphic Entity-seeking Queries. In *EMNLP*, 1104–1114, 2014.

[24] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in web text. In *KDD*, 457–466, 2009.

[25] X. Li. Understanding the semantic structure of noun phrase queries. In *ACL*, 1337–1345, 2010.

[26] Y. Li, B.-J.P. Hsu, C. Zhai, K. Wang. Unsupervised query segmentation using clickthrough for information retrieval. In *SIGIR*, 285–294, 2011.

[27] M. Manshadi, X. Li. Semantic tagging of web search queries. In *ACL*, 861–869, 2009.

[28] E. Meij, K. Balog, D. Odijk. Entity Linking and Retrieval for Semantic Search. In *WSDM*, 683–684, 2014.

[29] D. Milne and I. H. Witten. Learning to link with wikipedia. In *CIKM*, 509–518, 2008.

[30] M. Pasca. Weakly-supervised discovery of named entities using web search queries. In *CIKM*, 683–690, 2007

[31] F. Piccinno and P. Ferragina. From TagME to WAT: a new entity annotator. In *Workshop on Entity Recognition & Disambiguation* (ERD), hosted by SIGIR, 55–62, 2014.

[32] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *ACL-HLT*, 1375–1384, 2011.

[33] K. Risvik, T. Mikolajewski, P. Boros. Query segmentation for web search. In WWW (poster), 2003.

[34] S. Rüd, M. Ciaramita, J. Müller, and H. Schütze. Piggyback: using search engines for robust cross-domain named entity recognition. In *ACL-HLT*, 965–975, 2011.

[35] U. Scaiella, P. Ferragina, A. Marino, M. Ciaramita. Topical clustering of search results. In *WSDM*, 223-232, 2012.

[36] A. Sil, A. Yates. Re-ranking for Joint Named-Entity Recognition and Linking. In *CIKM*, 2369–2374, 2013.

[37] S. Guo, M.-W. Chang, E. Kiciman. To Link or Not to Link? A Study on End-to-End Tweet Entity Linking. In *NAACL-HLT*, 1020–1030, 2013

[38] F. Suchanek and G. Weikum. Knowledge Harvesting in the Big-data Era. In *SIGMOD*, 933–938, 2013.

[39] B. Tan, F. Peng. Unsupervised query segmentation using generative lannguage models and Wikipedia. In *WWW*, 347–356, 2008.

[40] R. Usbeck, *et alii*. GERBIL: General Entity Annotator Benchmarking Framework. In *WWW*, 1133–1143, 2015.

[41] X. Wei, F. Peng, B. Dumoulin. Analyzing web text association to disambiguate abbreviation in queries. In *SIGIR*, 751–752, 2008

[42] Q. Wu, C. Burges, K. Svore, J. Gao. Ranking, boosting, and model adaptation. Technical report, Microsoft Research, 2008

[43] X. Yin, S. Shah. Building taxonomy of web search intents for name entity queries. In *WWW*, 1001–1010, 2010.