SMAPH: A Piggyback Approach for Entity-Linking in Web Queries

MARCO CORNOLTI[*], Google
PAOLO FERRAGINA, University of Pisa
MASSIMILIANO CIARAMITA, Google
STEFAN RÜD and HINRICH SCHÜTZE, LMU Munich

We study the problem of linking the terms of a web-search query to a semantic representation given by the set of entities (aka concepts) mentioned in it. We introduce SMAPH, a system that performs this task using the information coming from a web search engine, an approach we call "piggybacking". We employ search engines to alleviate the noise and irregularities that characterize the language of queries. Snippets returned as search results also provide a context for the query that makes it easier to disambiguate the meaning of the query. From the search results, SMAPH builds a *set of candidate entities* with high coverage. This set is filtered by *linking back* the candidate entities to the terms occurring in the input query, ensuring high precision. A greedy disambiguation algorithm performs this filtering; it maximizes the *coherence* of the solution by iteratively discovering the pertinent entities mentioned in the query. We propose three versions of SMAPH that outperform state-of-the-art solutions on the known benchmarks and on the GERDAQ dataset, a novel dataset that we have built specifically for this problem via crowd-sourcing and that we make publicly available.

CCS Concepts: • **Information systems** → **Web search engines**; **Query representation**; **Query intent**; • **Computing methodologies** → *Machine learning approaches*; Classification and regression trees; Support vector machines;

Additional Key Words and Phrases: Entity linking, query annotation, ERD, piggyback

ACM Reference format:
Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Stefan Rüd, and Hinrich Schütze. 2017. SMAPH: A Piggyback Approach for Entity-Linking in Web Queries. *ACM Transactions on Information Systems* 0, 0, Article 0 (2017), 41 pages.
DOI: 0000001.0000001

1 INTRODUCTION, MOTIVATION, AND PROBLEM DEFINITION

As conversational interfaces become more popular in web applications, human-computer 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 through natural language in ubiquitous computing devices. An important aspect of

[*] Work mainly done while this author was a Post-doc at the University of Pisa.

We gratefully acknowledge support for this work through the following grants. (i) European Commission H2020 Program on "SoBigData: Social Mining & Big Data Ecosystem" (INFRAIA-1-2014-2015: Integrating and opening existing national and regional research infrastructures of European interest, Grant No. 654024); (ii) two Google Awards to Paolo Ferragina and Hinrich Schütze; and (iii) European Research Council Advanced Grant NonSequeToR, No. 740516.

Author's addresses: M. Cornolti and M. Ciaramita, Google, Zürich, Switzerland, {massi, cornolts}@google.com; P. Ferragina, Department of Computer Science, University of Pisa, paolo.ferragina@unipi.it; S. Rüd and H. Schütze, Center for Information and Language Processing, LMU Munich.

DOI: 0000001.0000001

capturing the semantics of a linguistic document is *entity-linking*, the task of linking terms in a text to the entity they refer to. By "entity" we refer to a concept with distinct and independent existence. We rely on knowledge bases as catalogs of entities. In this paper, we consider three types of entities:

- A **NamedEntity** is a real-world object, e.g., a person, location or organization, that can be denoted with a proper name.
  Examples: *Neil_Armstrong*, *United_Nations*, *Chicago*, *Semptember_11_attacks*, *Batman*[1].
- An **AbstractEntity** is an abstract concept.
  Examples: *peace*, *mathematics*, *centripetal_force*.
- A **CategoryOfEntities** is a set of things sharing some properties.
  Examples: *astronaut*, *superhero*, *physical_force*.

We use the word "entity" to refer to all three types of entities. When necessary for clarity, we will use the term **GenericEntities** to refer to all three types; that is, NamedEntities, AbstractEntities and CategoriesOfEntities are all GenericEntities.

Knowledge bases (such as Wikidata) are repositories of structured information about GenericEntities. For example, they may include the fact that *Neil_Armstrong* was an *astronaut*, and that he was born on August 5, 1930. For applications that need to retrieve these types of structured pieces of information, entity-linking is the first necessary step (see, e.g., [13, 35, 36, 50, 55]).

1.1 Problem definition

Prior to defining the problems we tackle in the paper, let us give two definitions:

- A ***mention*** is a span (encoded as the index of the first and last character) in a textual document that *explicitly* refers to an entity;
- An ***annotation***, indicated as $\mathtt{m} \mapsto e$, indicates that mention m refers to entity e.

We address the following two entity linking tasks:

- **Query-level entity-linking**[2] is the task of finding all entities mentioned in the query. The entities are found for the entire query, without determining which tokens mention them.
- **Token-level entity-linking**[3] is the task of finding all annotations in a query. In other words, all entities mentioned in the query and their mentions.

To understand the difference between query-level and token-level entity-linking, consider the following text:

$d =$ `After Armstrong stepped off the Apollo 11, he hummed "Hello, Dolly!" by Armstrong`

which contains two occurrences of the word `Armstrong` that are mentions of different entities. The output of query-level entity-linking is a set of entities:

$$\{\mathit{Neil\_Armstrong}, \mathit{Louis\_Armstrong}, \ldots\}$$

In contrast, the output of token-level entity-linking is a set of annotations:

$$\{m_1 \mapsto \mathit{Neil\_Armstrong}, m_2 \mapsto \mathit{Louis\_Armstrong}, \ldots\}$$

where mentions $m_1 = (6, 14)$ and $m_2 = (73, 81)$ are the two occurrences of term `Armstrong`.

[1] Though Batman arguably does not exist in the real world, its character does.

[2] We used the name **C2KB** for query-level entity-linking in earlier work, i.e., Concept to Knowledge Base. See Cornolti et al. [9], Usbeck et al. [53].

[3] We used the name **A2KB** for token-level entity-linking in earlier work, i.e., Annotation to Knowledge Base. See Cornolti et al. [9], Usbeck et al. [53].

Some queries are inherently ambiguous, for example `life of armstrong`. In such cases, we define both our problems as those of finding the entities according to the interpretation most humans (speaking the language of the query) would make of the query.

There are also queries in which one token is part of two mentions that refer to distinct entities, and the definition of one entity entails the other. For example, in the query `president of us 2016`, the mention `president of us` refers to the institution *President_of_the_United_States*, while `us` refers to *United_States*. In such cases, we define both our problems as those of finding the entailing entity (in the example, *President_of_the_United_States*) only. The rationale behind this is that the entailing entity has a more specific meaning, and better captures the semantics of the query. Note that the entailed entity can be retrieved from a knowledge base.

1.2 Motivation

Entity-linking in queries is a recent algorithmic challenge [5] that faces two main issues: (i) the noisy language of queries, characterized by misspellings and unreliable tokenization, capitalization and word order, and (ii) their brevity, as queries typically consist of just a few terms. Issue (i) makes it hard to find what entities a query term may refer to. This is a key task in entity-linking and typically relies on a dictionary generated from well-edited texts, such as Wikipedia articles. Issue (ii) results in the lack of context that can be leveraged to assist the disambiguation of the query terms and model the coherence of the interpretation. As a consequence, known entity-linking algorithms with good performance on longer and well-formed documents (such as books, blog posts or news, see e.g., [17, 22, 27, 42, 46]) are less effective when applied to queries.

Search engines, on the other hand, are very good at processing queries and pointing users to the information most relevant to their needs. Users interact with search engines by means of a query or a short sentence, that search engines have to interpret. Eventually, search engines return the list of Web documents that are pertinent to a user query, but they also provide important accessory information: for each retrieved Web document, they include a *snippet* of well-formed sentences extracted from those documents in which the query terms appear in a cleaner, normalized form, usually shown in bold.

We propose to deal with the challenges posed by open-domain web-search queries via a piggyback approach that exploits a web-search engine and the Wikipedia knowledge graph. In our terminology, *piggybacking* refers to the practice of building a system on top of the information provided by a search engine, which is treated as a black box. The intuition behind the *piggybacking* approach, first introduced in [48], 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 two issues presented above because they have been designed to deal with queries, as this is the main form of interaction with the user.

1.3 SMAPH: An entity-annotator for queries

In this paper we describe SMAPH, an *entity-annotator* (i.e., a system that performs entity-linking) for web-search queries that employs piggybacking and the Wikipedia knowledge graph. We describe three versions of SMAPH, denoted as SMAPH-1, SMAPH-S, and SMAPH-3. Each one of these versions has been developed to address a distinct research question that we state and discuss in this section. Starting at Section 4, we will describe the three versions of SMAPH in detail and present experimental results that support our claims.

1.3.1 Research Question #1: Can piggybacking improve the quality of entity-linking on queries?

This question is concerned with the performance of traditional entity-annotators when applied to open-domain web-search queries. To answer this question we compare traditional entity-annotators, which are designed for natural language entity-linking, against SMAPH-1, which is specifically designed for query entity-linking. SMAPH-1 relies on applying a robust natural language entity-annotator (i.e., WAT [45]) on the snippets returned by a search engine, to obtain a set of entities that are candidates for the annotation of the query. SMAPH-1 then chooses which of the candidate entities to keep and which to discard based on a machine-learned binary classifier that processes each entity individually and independently of the others, based on a set of features extracted from the snippets.

In order to train and evaluate the classifier, we built (via crowd-sourcing) GERDAQ, a dataset consisting of 992 queries and their ground-truth annotations. The entity knowledge base is the whole Wikipedia. It includes all three types of GenericEntities (see Section 1). GERDAQ is split into three portions: train (497 queries), development (249 queries), and test (246 queries).

Despite its simplicity, SMAPH-1 outperforms traditional entity-annotators designed for natural language text by a significant margin, with the idea of piggybacking being responsible for an important gain in precision. As a historical note, we observe that SMAPH-1 was designed to participate in the query-annotation track at *ERD'14* [5], a challenge hosted by SIGIR 2014, that has seen the participation of 19 teams competing to build an entity-annotator for queries. SMAPH-1 took the first place in the query annotation track.

1.3.2 Research Question #2: Can annotation quality be improved by enforcing the bond between a candidate entity and the query terms? An error analysis of SMAPH-1 highlighted an abundance of false negative entities, which hurt recall. These are candidate entities that, despite being explicitly mentioned in the query, are associated to poor signals and thus get discarded by SMAPH-1. This motivated the design of SMAPH-S, a query entity-annotator that performs **token-level entity-linking** by generating a set of candidate annotations and evaluating each candidate independently of the others via supervised machine-learning. Note that by solving token-level entity-linking on queries efficiently, we will get a better solution for query-level entity-linking too – a token-level entity-linking for a query can be trivially simplified to a query-level entity-linking by dropping the mentions and keeping the entities referenced by the query annotations.

The linking-back from entities to their mentions implemented by SMAPH-S lets it choose entities with higher confidence compared to SMAPH-1. This not only makes it better at finding more entities mentioned in the query (increasing recall), but also at discarding entities that are not mentioned by the query (increasing precision). Our experiments will show that, by enforcing the link between mention and entity, SMAPH-S increases the macro-F_1 (i.e. the average F_1 measure across queries of the dataset, see Section 9.1) reached by SMAPH-1 on the GERDAQ dataset by 3.5% when piggybacking on Bing, and by 4.2% when piggybacking on Google (results refer to NamedEntities recognition). Results on ERD-dev, a dataset similar to GERDAQ released by the ERD Challenge organizers (see description at Section 9.2.2), show a similar trend. When we consider GenericEntities in the evaluation, SMAPH-S obtains a macro-R higher than SMAPH-1, though the macro-F_1 is similar.

1.3.3 Research Question #3: Can the quality of token-level entity-linking be improved by modeling the coherence among the entities included in a solution? The linking-back of entities to mentions makes it easier to discard wrong entities, and thus improves precision. Analyzing the annotations produced by SMAPH-S we observed several incoherent solutions, i.e., annotations containing two entities that are mutually exclusive from a semantic point of view. To solve this issue, we propose SMAPH-3, the last and best performing version of SMAPH, that improves the previous versions by taking into account the coherence of the set of annotations forming the solution. SMAPH-3 builds the solution by adding one annotation at a time. This way, it can take into account the coherence between a new annotation and the others previously added. The training process of SMAPH-3 has the objective function of maximizing directly the macro-F_1 measure. This leads to the following improvements on GERDAQ. For query-level entity-linking, when considering NamedEntities only, the improvement in macro-F_1 is about 6% over SMAPH-1, and about 2% over SMAPH-S; when considering GenericEntities, the improvement is in the range $3.6 - 7\%$ over SMAPH-1 and SMAPH-S (depending on the search engine it piggybacks on, either Bing or Google). For token-level entity-linking, the improvement in macro-F_1 over SMAPH-S is 4% when piggybacking on Bing, and 5% when piggybacking on Google. With respect to natural language entity-annotators, such as WAT, SMAPH-3 improves macro-F_1 by 16.3%.

1.4 Entity-linking for queries: Our contribution

In summary, this paper makes four main contributions to the topic of entity-linking for queries.

- This article is the first one to investigate the problem of identifying GenericEntites (not just NamedEntities) in a query. The detection of AbstractEntities and CategoriesOfEntities has been recognized as a key feature of modern query-annotation tools [13].
- We build and release to the public the GERDAQ (General Entity Recognition, Disambiguation and Annotation in Queries) dataset that provides a ground truth for token-level entity-linking on 992 annotated queries.
- We propose three versions of a query-annotator, called SMAPH, providing state-of-the-art performance for open-domain query annotation. The core algorithm underlying the best performing version of SMAPH, called SMAPH-3, makes a greedy choice of annotations from a set of candidates, by using a model whose parameters are optimized on a training set. In contrast to prior work, we directly optimize F_1, the top-line metric of evaluation.
- We present an extensive set of experiments that evaluate the three SMAPH versions on GERDAQ and, when possible, on the datasets of the ERD Challenge: ERD-dev and ERD-online.

Part of this paper was previously published in the proceedings of the ERD 2014 Workshop [5], where we presented SMAPH-1 and the general idea of piggybacking on search engines. In the proceedings of the WWW 2016 conference [10] we published our first proposal for link-back, namely SMAPH-S, and designed a preliminary approach to collective disambiguation, which we called SMAPH-2. In these two papers we experimented with these entity-annotators by piggybacking on Bing only.

Later on we devised SMAPH-3 which, despite being designed to tackle the same issues as SMAPH-2 (lack of solution coherence), employs a radically different algorithm to choose which annotations to include in the solution. Since we consider SMAPH-3 a more elegant algorithmic approach than SMAPH-2, and since it experimentally turns out to be more accurate and faster, we decided to omit the description of SMAPH-2 from the present paper.

With respect to the previously published material, this paper presents five novel contributions: (i) a formulation of entity-linking as the problem of selecting a coherent subset of annotations from a pool of candidates, maximizing F_1; (ii) our best performing algorithm SMAPH-3 and its novel approach to selecting query annotations; (iii) an in-depth analysis of the contribution of each iteration of SMAPH-3 to the construction of the final query annotation. In addition to these algorithmic and methodological contributions, we present an extended experimental analysis of all three SMAPH versions. In particular, (iv) we use Google, in addition to Bing, as a piggyback search engine, which lets us test the robustness of SMAPH with respect to the search engine we piggyback on; (v) we perform a deeper analysis of results quality, including the robustness on less popular queries. This paper is also the first to present a coherent synthesis of the SMAPH entity-annotators.

2 RELATED WORK

In recent years, significant effort has been made to move beyond representing textual documents as a bag of words. One important line of work towards semantic representations relies on entity-annotators (see e.g. [12, 17–20, 31, 37, 38, 40, 45, 46]), with several interesting and effective algorithmic approaches to solve the mention-entity match problem. Most of these entity-annotators are developed for long documents. Recent works [50] adopt information drawn from Wikipedia, possibly in structured forms such as DBpedia or Wikidata (which also includes data from Freebase, a knowledge base widely used in the literature but now deprecated) in order to detect entity mentions and disambiguate them.

While a comprehensive review of known entity-annotators for natural language is beyond the scope of this paper, we introduce one of them, WAT [45], as we employ it as a component of SMAPH. WAT models entity disambiguation as a Learning-to-Rank task, in which candidate entities for each mention are ordered by the likelihood of them being the pertinent entity for that mention. As ranking algorithm, the author uses LambdaMART [4]. Features associated with a candidate entity and used for ranking are based on (i) the word embeddings of the mentions of the candidate entity in the articles of Wikipedia, and (ii) a random walk on the Wikipedia graph similar to *DeepWalk* [43]. The solution is generated by linking the top-ranking candidate entities to their corresponding mentions.

Most research concerning entity-based query understanding focuses on NamedEntity Recognition [41] (the task of finding what terms are mentions of NamedEntities, without linking them to the entity), possibly associated to *query intent* discovery [32] or *query classification* into pre-defined classes [16, 21, 35]. Some work has also focused on linguistic analysis of queries, for example by POS tagging terms or tagging them with a limited number of classes and other linguistic structures [1, 2], or assigning a coarse-grained purpose to each segment [30]. Wei et al. [54] present a method to do abbreviation disambiguation in queries.

As a source of information, these works may either use knowledge bases or information derived from web search such as query logs (see e.g., [28]), click through information [33], search sessions [14], top-k snippets from search engines [1], web phrase DBs [2, 23], or large manually annotated collections of open-domain queries to extract robust frequency or mutual-information features and contexts [16].

Another interesting line of research about queries is that of query segmentation. Its goal is to find the lexical units, either compounds or single-token units, in a query. The authors of these works show that a search engine can exploit such segmentation to increase result precision, since documents that do not contain the lexical units in proximity or even in the exact same order can be discarded (see e.g., [29, 47, 51]). More recently, a series of papers by Hagen et al. (see e.g., [23]) proposed simple and effective scoring functions for lexical units that use a weighted sum of normalized web-phrase frequencies (taken from the Google *n*-gram corpus), and showed via a large experimental test that query segmentation based on titles of Wikipedia articles is very effective.

The literature also features works that explicitly treat entity-linking in queries. Blanco et al. [3] propose a fast and space-efficient entity-linking method leveraging information from query logs and anchor texts. Their method solves a *ranking version* of token-level entity-linking by returning a ranking of annotations. The entity-annotator was evaluated on the Webscope L24 (Yahoo Search Query Log To Entities) dataset taking into account entities only (i.e., mentions are not evaluated). This way the authors did not determine the final set of annotations for a query but, rather, a ranked list of (possibly several) entities which was then evaluated by means of typical ranking metrics. The authors did not evaluate their entity-annotator on the dataset of the ERD Challenge nor against the entity-annotators participating in the ERD Challenge.

Another interesting work is that presented by Hasibi et al. [25] (a follow-up of the NTNU-UiS entity-annotator presented at the ERD Challenge), where the authors describe a method to solve the query-level entity-linking problem. The method is based on three phases: (i) candidate annotations are generated aiming at maximum recall, by exploiting two sources: DBpedia and Google's Freebase Annotations of the ClueWeb Corpora (FACC); (ii) candidate annotations are assigned a score by combining, via a generative model, the Mixture of Language Models (MLM) with a commonness score; (iii) interpretations are iteratively generated by picking non-overlapping annotations, starting from the ones with higher score. The entity-annotator is evaluated on a cleaned version of the Webscope L24 dataset, where only explicitly mentioned NamedEntities are kept (i.e., GenericEntities that are not NamedEntities and implicit mentions are removed). This entity-annotator outperforms TagMe, which is used as a baseline. Hasibi et al. [25] address the problem of *semantic mapping*, i.e., finding the ranked list of pertinent entities, possibly without explicit mentions in the query (for example, *Ann_Dunham* for query `obama mother`). In a follow up [26], they propose an annotator that employs supervised learning for the entity ranking step while tackling disambiguation with an unsupervised algorithm. The annotator of [26] has a performance slightly lower than SMAPH-2.

Finally, the problem of entity-linking on queries has been approached by Tan et al. [52]. The underlying idea of this entity-annotator (we will refer to it as *Tan et al.*) is to search Wikipedia articles for sentences similar to the query, and rank these articles using a linear model based on features such as link-probability, context matching, word embeddings, and relatedness among candidate entities. The entity-annotator has been tested on GERDAQ, where it reaches a performance slightly lower than our SMAPH-3. Experiments have also been done on the dataset of the ERD Challenge, though no direct fair comparison with prior work is provided since the entity-annotator was trained on a portion of the dataset that was never released to the public.

Unlike other approaches, SMAPH does not treat entity recognition and entity disambiguation as two separate problems. Though SMAPH is the first method to do entity-linking on queries treating these problems jointly, this approach has already been explored for entity-linking on natural language documents by Sil and Yates [49], who proposed to generate candidates with independent NamedEntity recognition and multiple entity-annotators, and then re-rank candidates with a linear maximum entropy model trained to maximize the L2-regularized conditional log-likelihood. This approach has also been explored for entity-linking on tweets by Guo et al. [22], who proposed to generate candidate annotations with a base linking model and represent the set predicted annotations as a binary vector, so that learning can use as loss function the Hamming distance between the predicted vector and the gold vector.

3 GERDAQ, A DATASET FOR TRAINING AND TESTING QUERY ENTITY-ANNOTATORS

In this section, we explain how we built the GERDAQ dataset. GERDAQ provides ground truth annotations for 992 queries. It can be used as a source of examples to do supervised machine learning or to test the quality of query entity-annotators.

GERDAQ is the result of a cooperation between the University of Pisa, University of Munich, and Google. The cost of creating this dataset was roughly $2000, funded by a Google Research award in 2013. Queries are derived from the KDD Cup 2005 dataset [34] and have been annotated by workers on Crowdflower[4], an on-line meta-crowdsourcing engine. The dataset has been released[5] for free to the community, under a Creative Commons license, to assist the development entity-annotators and support research on query entity-linking.

Even for experts on the subject, it is often hard to find correct annotations for a query. Human annotation requires an understanding of what the user had in mind when she typed the query. The human has to spot the mention of an entity and pick the entity it references from a knowledge base. No single worker has knowledge of the whole catalog of entities provided by a knowledge base; and no worker can interpret queries typed by any user, each with different background and culture.

In order to get as close as possible to the goal of finding high quality annotations, we had multiple human workers annotate the same query. After selecting the queries to include in GERDAQ, the first phase was aimed at *maximizing coverage*, the second at *refining precision*. In the rest of this section, we give details on the dataset construction workflow.

3.1 Phase 0: Query selection

The queries forming GERDAQ instances have been sampled from the KDD-Cup 2005 competition dataset, which consists of 800,000 queries. These queries are taken from MSN search logs with some preliminary filtering.

First we cleaned the dataset by discarding the queries that looked like web addresses (i.e., those containing `www` or `http`), then we randomly sampled 992 queries[6]. Figure 1 presents example queries from the random sample.

[4] www.crowdflower.com

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

[6] The number of queries was originally 1000, eight were later removed because they featured illegal or potentially offensive content.

```
south st philly stores
cooking school "mont st. michel
muskingum collage music dept.
anteques motorcycles
sidney lumet familt
metronome setting of allegro
```

Fig. 1. Some queries extracted from the KDD-Cup 2005 competition that became instances of GERDAQ.

Query: `south st philly stores`	
`stores` ↦ *Retail*	3/11
`south st philly` ↦ *South_Street_(Philadelphia)*	6/11
`philly` ↦ *Philadelphia*	2/11

Query: `cooking school "mont st. michel`	
`cooking school` ↦ *Cooking_school*	7/10
`mont st. michel` ↦ *Mont_Saint-Michel*	8/10

Query: `photos of stary night`	
`photos` ↦ *Photograph*	6/11
`stary` ↦ *Star*	2/11
`night` ↦ *Night*	4/11
`stary night` ↦ *The_Starry_Night*	6/11

Table 1. Annotations proposed by workers in Phase 1 for three queries. Right column indicates how many workers spotted an annotation.

3.2 Phase 1: Maximizing coverage

This phase aimed at maximizing the coverage of the annotation process for the queries of GERDAQ, without considering the precision of the annotations. We set up a job on Crowdflower and, for each query, asked workers to spot annotations in queries, namely a mention and the entity referenced by it, in the form of a Wikipedia URL. Workers were instructed to not be conservative and to spot as many annotations as they could, up to 10 annotations per query. The job was set up so to accept only mentions that were actual substrings of the query and URLs that were existing English Wikipedia articles.

The quality of annotations was covertly tested during the execution of the job. Queries of the GERDAQ dataset to be annotated were issued to the workers, but, among them, we inserted a set of 70 additional quality-control queries, in a way that workers could not distinguish them. Quality-control queries were chosen so as to be of simple interpretation and not ambiguous. For those queries, we also built a ground truth. A worker response for a quality-control query was considered acceptable if the worker spotted at least one annotation of the ground truth built by us. When workers issued wrong responses for quality-control queries, they were prompted with an error message explaining the correct annotation process, but workers who persisted in failing were permanently excluded from the job, and their previous responses ignored.

Since no worker has full knowledge of all domains, high coverage could only be reached if this job employed as many workers as possible, so as to cover different backgrounds and cultures. For this reason, for each query

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".

we collected responses from at least 10 different workers. The job completed in a few hours and collected a total of 10,038 responses (not counting those given by unreliable workers and those given for quality-control queries). A total of 271 workers took part in the job; they processed 37 queries each on average, and found a total of 3,197 distinct annotations (3.2 per query).

Table 1 shows the output from Phase 1 for three queries. Each annotation is associated with the number of workers that found it. As figures from a later step of refinement show (see Tables 2 and 3), the fact that an annotation was found only by a few workers does not indicate that it is wrong. For example, in query `south st philly stores`, the user intent is to get information about stores in South Street, a street in Philadelphia that is one of the city's largest tourist attractions. 6 out of 11 workers correctly recognized that `south st philly` refers to that street, but only three annotators found that term `store` refers to retail stores. Nonetheless, both annotations are correct. On the other hand, in the query `photos of stary night`, the user intent is to find pictures of the night view of the sky in which stars are visible. Entities *Photograph*, *Star*, *Night* are correctly found, but 6 out of 11 workers identified `stary night` as being a mention of Van Gogh's famous painting "The Starry Night". Both are reasonable interpretations, and the subsequent Phase 2 of refinement will have to choose which is the most common interpretation.

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, as we will see later, most of them were nonetheless judged as being correct in Phase 2.

3.3 Phase 2: Refining precision

Phase 2 aimed at discarding bad annotations found by workers in Phase 1. We created a second job on CrowdFlower, asking workers to judge, on a scale from 1 to 10, the likelihood that a certain 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 Wikipedia article about the candidate entity (e.g., *Neil Alden Armstrong was an American astronaut...*), to better distinguish correct entities from wrong ones.

Similarly to Phase 1, Phase 2 featured covert quality-control. For the same set of 70 queries used for quality-control in Phase 1, we manually generated 76 correct and 69 wrong annotations. Like in Phase 1, we chose simple, unambiguous cases. These annotations were covertly provided to workers that had to judge them. To have their responses on quality-control instances considered acceptable, workers had to assign a score between 1 and 4 to wrong annotations, and a score between 7 and 10 to correct annotations. Workers failing to recognize multiple quality-control annotations were excluded from the job and their contribution not taken into account.

Each query was processed by at least 3 workers, for a total of 390 workers who took part in the job and processed 26 queries each on average, generating a total of 9,612 annotation scores. The distribution of average scores assigned to annotations in Phase 2 is shown in Table 3. Numbers show that workers of Phase 2 considered as correct a big fraction of annotations, even among those that were found by a limited number of workers in Phase 1. Sampled output from Phase 2 is shown in Table 4.

0	**.1**	**.2**	**.3**	**.4**	**.5**	**.6**	**.7**	**.8**	**.9**	**1.0**
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 [0.1, 0.2). 375 annotations were given the maximum score by all workers (last column).

Query: south st philly stores		
Annotation	**Ph. 1**	**Avg. Score**
stores ↦ *Retail*	3/11	0.778
philly ↦ *Philadelphia*	2/11	0.889
south st philly ↦ *South_Street_(Philadelphia)*	6/11	0.778
Query: cooking school "mont st. michel		
Annotation	**Ph. 1**	**Avg. Score**
cooking school ↦ *Cooking_school*	7/10	0.867
mont st. michel ↦ *Mont_Saint-Michel*	8/10	0.814
Query: photos of stary night		
Annotation	**Ph. 1**	**Avg. Score**
photos ↦ *Photograph*	6/11	1.000
stary ↦ *Star*	2/11	0.800
night ↦ *Night*	4/11	0.800
stary night ↦ *The_Starry_Night*	6/11	0.222

Table 4. Score assigned by workers to annotations in Phase 2. Central column indicates how many workers spotted an annotation in Phase 1, right column indicates the average of the scores assigned by workers in Phase 2.

3.4 Phase 3: Manual refinement by experts

A non-trivial issue was that of defining a threshold on the average score to discard wrong annotations. We decided to leave this job to expert human judgment. By randomly sampling annotations, we observed that all annotations with a score smaller than 0.58 were wrong, and thus to be discarded. Similarly, all annotations with a score above 0.65 were correct. Annotations in the range 0.58 – 0.65 (a few dozen) were manually double checked for correctness until complete agreement between two members of our research team was reached.

Even after filtering out the annotations with a low score, there can be two annotations for the same query having overlapping mentions. Since a term cannot be part of two distinct mentions, one annotation had to be discarded. This happened for 90 mentions over a total of 2043. This may occur because of three reasons:

(1) **Actual ambiguity of the query:** uncertainty on which entity to link. For example, in query armstrong moon, mention armstrong can either be interpreted as mentioning *Neil_Armstrong* or *Louis_Armstrong*. Since we are building a dataset that labels the most common interpretation (according to the problem definition, see Section 1.1), we solved these cases by keeping the annotation with the highest score.

(2) **Unclear mention but unambiguous entity:** uncertainty on what mention to link to the entity. We solved these cases by choosing the annotation with highest score among those linking the same entity.

Portion	queries	queries with ≥ 1 anns	Avg. anns per non-empty query	Avg. anns per query	Avg. query length (chars)
GERDAQ-train	497	446	2.07	1.87	25
GERDAQ-dev	249	221	2.05	1.82	22
GERDAQ-test	246	220	1.95	1.75	23

Table 5. GERDAQ dataset statistics for each portion. The second column indicates the number of queries in a portion; the third column indicates the number of queries having at least one annotation (*non-empty* queries); the fourth column indicates the average number of annotations among non-empty queries while the fifth indicates the same quantity among all queries; the last column indicates the average query length in characters.

(3) **Entity entailment.** For example, in query `president of u.s. 2006`, mention `president of u.s.` refers to the institution *President_of_the_United_States*, but an alternative annotation might link `u.s.` to *United_States*. In such cases, according to the problem definition (see Section 1.1), we discarded the annotation referencing the entailed entity (*United_States*) and keep the other annotation (the one referencing *President_of_the_United_States*).

In case of actual query ambiguity (case 1), the GERDAQ dataset also features secondary interpretations found by the workers, which are available for future work, though not considered in this paper.

Phase 3 keeps 2043 (out of 3197) distinct annotations (an average of 2.0 annotations per query), constituting the ground truth for the GERDAQ dataset (see Table 5 for basic statistics).

We randomly split GERDAQ into *training set* (GERDAQ-train, 497 queries), *development set* (GERDAQ-dev, 249 queries), and *test set* (GERDAQ-test, 246 queries). We encourage researchers to train and tune their entity-annotators on the first two portions and keep the test set for the final evaluation report.

4 CANDIDATE ENTITY GENERATION

We start the description of SMAPH by presenting its first step, which aims at finding a set of entities that are good candidates for being part of the solution. This step is shared among all versions of SMAPH. Its purpose is to find a set of candidate entities for input query q. The reader might find it useful to follow the flow chart in Figure 2, where candidate entity generation appears in the upper part.

For all versions of SMAPH, this is the only step in which new entities come into play (some of them will be discarded in later steps). For this reason, this step aims at maximizing the coverage of the candidate set, which poses an upper bound to the recall of the eventual solution for q.

Candidate entities are generated by processing the results returned by the public `API` of a search engine (in our experiment, we will use either `Bing` or `Google`), and consists of two phases.

4.1 Phase 1: Fetching

Search results for two queries are fetched from the search engine. The first query is the input query q, the second query is q^w, consisting of q concatenated with the word `wikipedia`. For both queries, we enable the search engine's *spelling correction* feature, so that results are not affected by spelling errors possibly present in the query. The second query q^w boosts results from Wikipedia. Note that search engines also support domain-restricted queries that could be used to search among Wikipedia articles only. We decided not to use this type of search because it tends to return articles loosely related to the actual query. Instead, by simply appending the word `wikipedia`, we give the search engine a soft suggestion about the kind of pages we are interested in, and also obtain, as their rank, a signal of their pertinence to query q.

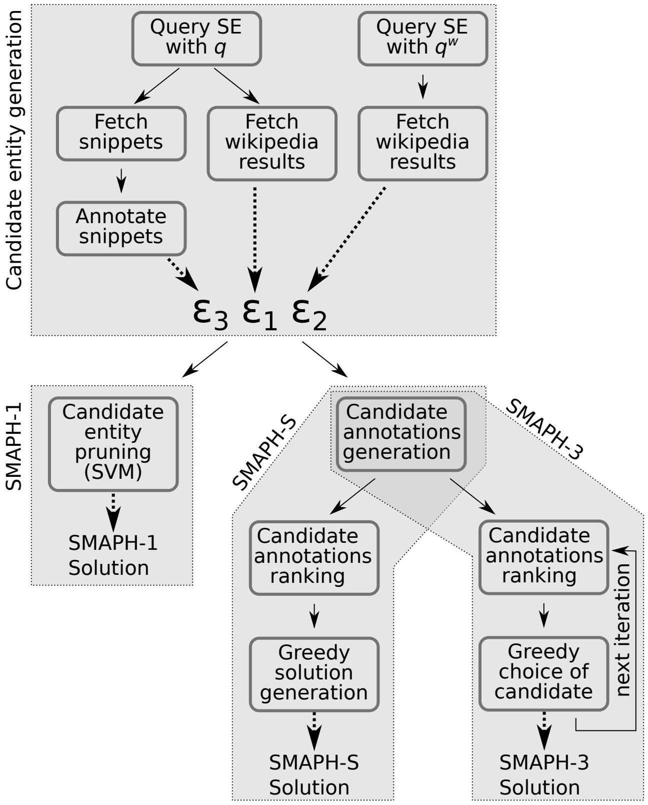

Candidate entity generation
Query SE with q
Query SE with q^w
Fetch snippets
Fetch wikipedia results
Fetch wikipedia results
Annotate snippets
$\mathcal{E}_3$ $\mathcal{E}_1$ $\mathcal{E}_2$
SMAPH-1
Candidate entity pruning (SVM)
SMAPH-1 Solution
SMAPH-S
SMAPH-3
Candidate annotations generation
Candidate annotations ranking
Greedy solution generation
SMAPH-S Solution
Candidate annotations ranking
Greedy choice of candidate
next iteration
SMAPH-3 Solution

Fig. 2. Overview of the three SMAPH entity-annotators. Continuous lines indicate workflow; dashed lines indicate generated data. The first step, *candidate entity generation*, is shared among the three versions of SMAPH. The solution provided by SMAPH-1 is found by pruning the set of candidate entities. SMAPH-S and SMAPH-3 share the step of *candidate annotations generation*. The solution provided by SMAPH-S and SMAPH-3 is a subset of candidate annotations, but they differ in the way this subset is chosen: SMAPH-S builds the solution by judging candidate annotations independently, while SMAPH-3 iteratively builds the solution also taking into account their coherence.

4.2 Phase 2: Candidate-entity generation

Entities are drawn from three sources:

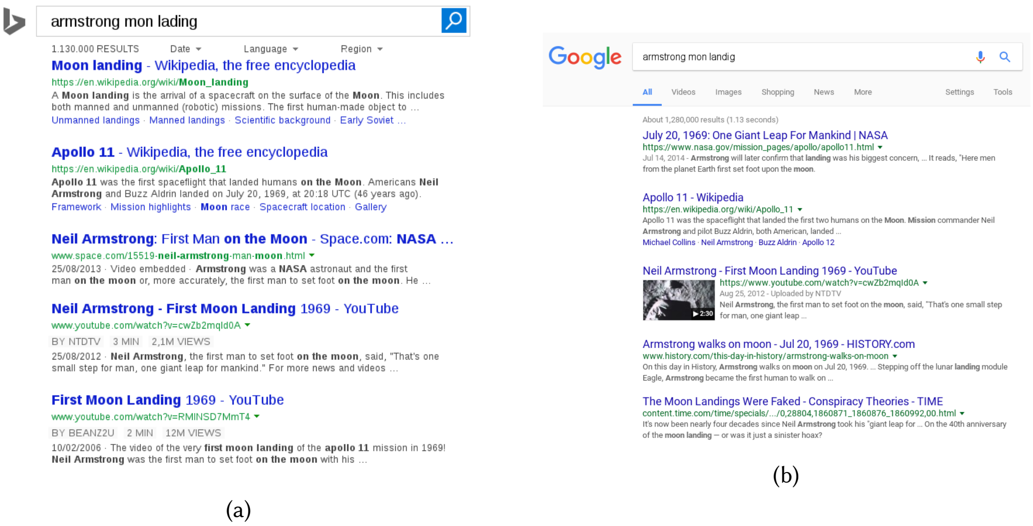
Source 1. Among the top-5 URLs returned by the search of q, we find those that point to Wikipedia articles. Corresponding entities form the set $\mathcal{E}_1$.

Source 2. The same is done with the top-10 URLs returned by the search of q^w. These form the set $\mathcal{E}_2$.

Source 3. Snippets of the top-15 results of the first search are, independently of each other, fed to the WAT entity-annotator[7]For each snippet, WAT returns a set of annotations. Among these, we consider only the ones overlapping with a bold-highlighted substring of the snippet. Their entities form the set $\mathcal{E}_3$.

WAT has very good performance on annotating sentences excerpted from longer documents [44], and snippets are indeed excerpts from web pages of a few dozen terms. WAT finds mentions in the snippets and disambiguates

[7] We tried several annotators other than WAT, but they yielded worse performance when annotating snippets.



Moon landing - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Moon_landing**
A **Moon landing** is the arrival of a spacecraft on the surface of the **Moon**. This includes both manned and unmanned (robotic) missions. The first human-made object to …
Unmanned landings · Manned landings · Scientific background · Early Soviet …

Apollo 11 - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Apollo_11**
Apollo 11 was the first spaceflight that landed humans **on the Moon**. Americans **Neil Armstrong** and Buzz Aldrin landed on July 20, 1969, at 20:18 UTC (46 years ago).
Framework · Mission highlights · **Moon** race · Spacecraft location · Gallery

Neil Armstrong: First Man **on the Moon** - Space.com: **NASA** …
www.space.com/15519-**neil-armstrong**-man-**moon**.html
25/08/2013 · Video embedded · **Armstrong** was a **NASA** astronaut and the first man **on the moon** or, more accurately, the first man to set foot **on the moon**. He …

Neil Armstrong - First Moon Landing 1969 - YouTube
www.youtube.com/watch?v=cwZb2mqld0A
BY NTDTV 3 MIN 2,1M VIEWS
25/08/2012 · **Neil Armstrong**, the first man to set foot **on the moon**, said, "That's one small step for man, one giant leap for mankind." For more news and videos …

First Moon Landing 1969 - YouTube
www.youtube.com/watch?v=RMINSD7MmT4
BY BEANZ2U 2 MIN 12M VIEWS
10/02/2006 · The video of the very **first moon landing** of the **apollo 11** mission in 1969! **Neil Armstrong** was the first man to set foot **on the moon** with his …

(a)

About 1,280,000 results (1.13 seconds)

July 20, 1969: One Giant Leap For Mankind | NASA
https://www.nasa.gov/mission_pages/apollo/apollo11.html
Jul 14, 2014 - **Armstrong** will later confirm that **landing** was his biggest concern, ... It reads, "Here men from the planet Earth first set foot upon the **moon**.

Apollo 11 - Wikipedia
https://en.wikipedia.org/wiki/Apollo_11
Apollo 11 was the spaceflight that landed the first two humans on the **Moon**. **Mission** commander Neil **Armstrong** and pilot Buzz Aldrin, both American, landed ...
Michael Collins · Neil Armstrong · Buzz Aldrin · Apollo 12

Neil Armstrong - First Moon Landing 1969 - YouTube
https://www.youtube.com/watch?v=cwZb2mqId0A
▶ 2:30
Aug 25, 2012 - Uploaded by NTDTV
Neil **Armstrong**, the first man to set foot on the **moon**, said, "That's one small step for man, one giant leap ...

Armstrong walks on moon - Jul 20, 1969 - HISTORY.com
www.history.com/this-day-in-history/armstrong-walks-on-moon
On this day in History, **Armstrong** walks on **moon** on Jul 20, 1969. ... Stepping off the lunar **landing** module Eagle, **Armstrong** became the first human to walk on ...

The Moon Landings Were Faked - Conspiracy Theories - TIME
content.time.com/time/specials/.../0,28804,1860871_1860876_1860992,00.html
It's now been nearly four decades since Neil **Armstrong** took his "giant leap for ... On the 40th anniversary of the **moon landing** — or was it just a sinister hoax?

(b)

Fig. 3. Top 5 results returned by Bing (a) and Google (b) for query armstrong mon lading.

them by exploiting the context provided by the snippet. The reason why we only keep the annotations that overlap with a bold-highlighted substring of the snippet is that these bold-highlighted substrings are usually the form in which entities mentioned by the query are mentioned in web pages. In snippets, these forms generally occur in a canonical, spell-corrected form.

We set a limit to the number of search engine results that are considered for the generation of sets $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ (set, respectively, to 5, 10 and 15 top results). The objective of this phase is to reach high coverage. We explored values for these limits by means of a simple grid search and found that, as detailed in Section 9.3, the limits we set guarantee the coverage of the union of the sources to be 87.9% for GenericEntities and 96.7% for NamedEntities[8], which we consider satisfactory.

While Sources 1 and 2 are straightforward to understand, the explanation of Source 3 may benefit from an example. Consider the query:

$$q = \texttt{armstrong mon lading}$$

This query features two misspellings in the writing of terms moon and landing, and the ambiguous term armstrong, which is a reference to *Neil_Armstrong*. Figures 3a and 3b show the top 5 results returned by Bing and Google, respectively, when searching for query q. Consider as an example the third snippet by Bing:

> Video embedded - **Armstrong** was a **NASA** astronaut and the first man **on the moon** or, more accurately, the first man to set foot **on the moon**. He...

Note that the snippet provides a spelling correction for query term mon. Intuitively, snippets provide a rewritings of query terms in a form that was used in a web page. Snippets also put query terms into a context that can be leveraged to assist disambiguation. In fact, when WAT is fed the snippet in the example, it searches the text for potential mentions, finding four sequences of tokens: Armstrong, NASA, astronaut, and man on the moon. The first mention, Armstrong, is ambiguous in that by itself it could refer to *Neil_Armstrong*, *Louis_Armstrong*, *Armstrong_County,_Pennsylvania* or other entities, but it is placed in the same context (the same web page) as other terms such as NASA and astronaut that WAT can use to disambiguate the mention into *Neil_Armstrong*.

[8] Coverage computed on the development set.

WAT also makes mistakes, for example, it may link `man on the moon` to the 1999 movie starring Jim Carrey. In total, WAT finds four annotations for this snippet:

`Armstrong` ↦ *Neil_Armstrong*
`NASA` ↦ *National_Aeronautics_and_Space_Administration*
`astronaut` ↦ *Astronaut*
`man on the moon` ↦ *Man_on_the_Moon_(film)*

The mention `astronaut` does not overlap with any bold portion of the snippet, hence it is discarded. The lack of overlap between mention and bold text indicates that entity *Astronaut* is not explicitly mentioned by the query. This rule has some exceptions, namely terms (in the example, `NASA`) that are not included in the query but are rendered in bold with the purpose of assisting users reading through results. These entities will be discarded in more sophisticated ways (see Section 5.3).

This snippet contributes to $\mathcal{E}_3$ with the following entities: *National_Aeronautics_and_Space_Administration*, *Neil_Armstrong*, *Man_on_the_Moon_(film)*. More entities will be added by analyzing the other snippets.

5 SMAPH-1: INDIVIDUAL ENTITY PRUNING

SMAPH-1 chooses a subset of $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ and returns it as a result. The choice of the subset is implemented with a binary classifier that judges each candidate entity independently and decides whether it should be included in the result or not.

5.1 Entity pruning via SVM binary classification

As binary classifier, SMAPH-1 employs a Support Vector Machine classifier (SVC) with an RBF (Radial Basis Function) kernel. We used the implementation LibSVM [6], a library for training and testing SVM models.

The input instances to the SVM are the candidate entities in $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$. Each candidate entity is associated with a feature vector. Training examples are gathered by running the candidate entity generation step for all queries included in GERDAQ-train, and labeling the candidate entities included in the ground truth as positive examples, the others as negative.

In Section 9.4 we report details on the results of the training procedure, including the values obtained for the hyperparameters. Feature selection was performed by forward selection of features, i.e., in each step the best remaining feature is added that improved the objective. For both parameter optimization and feature selection, the objective function is macro-F_1 on GERDAQ-dev.

SMAPH-1 returns as the result for q those entities of $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ that were classified as positive by the SVM.

5.2 Entity features

The features we associate with each entity take into account the coherence and robustness of the snippet annotation process, the ranking of snippets, the string similarity between q and the snippets' bold text portions, and the string similarity between q and the title of the Wikipedia article about the candidate entity. Before exploring the features presented in Table 6, we need a few definitions:

- $U(q)$ is the ordered list of URLs returned by the search engine for query q;
- $\mathcal{Z}(q)$ is the ordered list of snippets returned by the search engine for q;
- $\mathcal{B}(q)$ is the multi-set of bold portions of all snippets returned by the search engine for q;
- $\mathcal{W}(q)$ is the total number of web pages found by the search engine for q;
- $\mathcal{T}(e)$ is the title of the Wikipedia article about entity e;
- $\mathcal{T}^*(e)$ is $\mathcal{T}(e)$ excluding the final parenthetical-string, if any. e.g., $\mathcal{T}^*($*ER_(TV_series)*$)$ = `ER`
- $\mathcal{A}(s)$ is the set of annotations found by WAT in snippet s that are overlapping with a bold portion of s;

- $X(q) = \{(m, s) : s \in \mathcal{Z}(q) \wedge (m, e) \in \mathcal{A}(s)\}$ are the mentions that have been found by WAT, paired with the snippet where they have been found;
- $\rho(s, m, e)$ is the ρ-score returned by WAT indicating the confidence in the annotation (m, e) for snippet s (see [44]);
- $lp(m)$ is the *link probability* of mention m (see [40]), computed as the ratio between the number of times m is an anchor in Wikipedia divided by the number of all its occurrences in Wikipedia;
- $comm(m, e)$ is the *commonness* of the annotation (m, e) (see [40]), computed as the number of links in Wikipedia having m as anchor and linking to e, divided by the number of times anchor m appears in Wikipedia as a link to *any* article.
- $amb(m)$ stands for *ambiguity* and is the number of distinct Wikipedia articles that a mention m points to;
- $ED(x, y)$ is the Levenshtein distance between strings x and y, divided by $\max(|x|_c, |y|_c)$, where $|x|_c$ is the length of x in characters;
- $MinED(a, b)$ is an asymmetric measure of distance of string a towards string b, defined as follows: let us indicate as $\text{avg}(X)$ the arithmetic mean of the elements of X, and let a_t and b_t be the set of terms in strings a and b, then

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

  In other words, for each term in a, we find the closest term in b; $MinED(a, b)$ is the average distance between them.

The feature vector for each entity is composed of the features in Table 6 (hence, the dimensionality of the original feature space is $p = 24$). With reference to Table 6, features 1 and 2 are relative to entities drawn from any source. Feature *webTotal* is the number (as estimated by the search engine) of documents in the web that matched query q. A high value may indicate that it is more likely for a query to be well formed, thus containing entities. Feature *isNE* indicates whether the entity is a NamedEntity. Compared to other kinds of entities, references to NamedEntities are less ambiguous, hence more likely to be correct.

We also define features 3–8, relative to sources 1 and 2 (Wikipedia articles as web results of q and q^w). The rationale behind Feature 3 is that if an entity has appeared in higher positions among the web results, it is more likely to be mentioned by the query. Features 4, 5 and 6 are the minimum edit distance between (i) the title of the Wikipedia article about the entity and (ii) the query, and between (i) the bold portions of the snippets mentioning the entity and (ii) the query. These distances indicate how likely the entity is mentioned by the query. The average number of terms in bold contained in snippets (Feature 8) is an indicator of how likely the query mentions entities, while the number of capitalized terms in the bold portion (Feature 7) gives an indication of whether the mention is a proper name.

Entities drawn from Source 3, our largest source of candidates, are associated with a set of features (9–24) relative to the process of snippet annotation performed by WAT. Feature *freq* (how many snippets mention the entity) is an obvious indicator of an entity's correctness. Similarly, feature *avgRank* captures where, in the list of web search results, the entity is mentioned: if it is mentioned in higher-ranked snippets, it is more likely to be correct. For each annotation found in a snippet, WAT returns ρ, a confidence score. We also have *lp* (the prior probability that the mention refers to any entity) and *comm* (the prior probability that the mention refers to that particular entity, among the candidates, and not taking context into account). For these three values, we take the minimum, maximum, and average value among the snippets. Higher values indicate strength of the entity. In contrast, *ambig* is the number of senses a mention may have in different contexts, and lower values suggest higher confidence in the annotation. Finally, similarly to the edit-distance measures previously described, Features 23 and 24 indicate to what degree the snippet terms that have been linked to the entity are also contained in the query.

Drawn From All Sources

ID	Name	Definition
1	*webTotal*	$\mathcal{W}(q)$
2	*isNE*	1 if e is a NamedEntity, 0 otherwise. Based on the list of NamedEntities provided by [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*	$(\lvert s \in \mathcal{Z}(q)\ :\ (\cdot, e) \in \mathcal{A}(s)\rvert)/\lvert\mathcal{Z}(q)\rvert$
10	*avgRank*	$(\sum_{i\in[0,25)} p_i)/25$ where $p_i = \begin{cases} i & \text{if } (\cdot, e) \in \mathcal{A}(\mathcal{Z}(q)_i) \\ 25 & \text{otherwise} \end{cases}$
11	*pageRank*	PageRank of e in Wikipedia Graph
		$\mathrm{P} := \{\rho(s, m, e) : (m, s) \in X(q)\};$
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 6. Features of a candidate entity e (used by SMAPH-1, SMAPH-S and SMAPH-3) for query q.

5.3 Limitations of SMAPH-1 and the need of link-back

An error analysis of SMAPH-1, confirmed by the results of experiments in Section 9, shows that SMAPH-1 makes mistakes both in terms of false positives and false negatives. Typically, FNs are entities that appear as candidates, are explicitly mentioned in the query, but are assigned a low score due to their bad feature values and, thus, discarded by the SVM binary classifier (for example, an entity ambiguously mentioned by the query may appear with low frequency in snippets, leading to low values of Feature 9). On the other hand, FPs are typically entities that have good feature values, are somehow related to the query, but are not mentioned by it (for example, an entity strongly related to the query, but not mentioned by it, may be mentioned by snippets with high rank, leading to high values of Feature 10).

We address these errors by explicitly modeling the bond between a candidate entity and its mention, so to prefer entities that are more likely to have a mention in the query (despite having poor feature values) over entities that are less likely to have a mention in the query (despite having good feature values). We call this approach *link-back*.

Another type of mistake is incoherence. Namely, entities whose presence is mutually exclusive from a semantic point of view. In particular, SMAPH-1 may include in the solution two entities that are incompatible interpretation of the same set of query terms, whereas only the most appropriate had to be chosen, or include entities that have no semantic relatedness with each other. For example, in query `armstrong mon lading`, SMAPH-1 may include *Neil_Armstrong* (correct) and *Man_on_the_Moon_(film)* (wrong), even though a third entity, *Moon_Landing* would be more coherent with *Neil_Armstrong*.

To improve coherence we consider annotations in relation with other annotations included in the solution. To this end, we model the problem of entity-linking on queries as the problem of subset selection: from the set of candidate annotations, we choose the ones that form the best solution, according to our models.

6 GENERATION OF CANDIDATE ANNOTATIONS AND SUBSET SELECTION

We now present two algorithms that, in contrast to SMAPH-1, take both mentions and entities into account: SMAPH-S (Section 7) and SMAPH-3 (Section 8). We start by describing the step of candidate annotations generation, which is shared among them.

Given an input 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 candidate entities for q. The set of candidate annotations C_q for q is simply the Cartesian product $C_q = Seg(q) \times \mathcal{E}_q$. This set contains a high number of annotations, and only a few of them are correct. The set also contains a high number of annotations conflicting with other annotations because of their overlapping (possibly equal) mentions.

SMAPH-S and SMAPH-3 determine a solution for the token-level entity-linking problem by choosing a subset of C_q. We can formalize this subset selection problem over C_q as the problem of choosing the elements of C_q that form the best solution according to metric F_1. In this problem there are constraints to avoid the presence of incompatible elements in the solution. Formally, the problem is that of finding the subset $S^* \subseteq C_q$ such that:

$$\begin{cases} S^* = \arg\max_{S \subseteq C_q} F_1(S) \\ \forall x, y \in S^*,\ x \neq y \Rightarrow \neg K(x, y) \end{cases} \tag{1}$$

where K is a binary relation on C_q that is verified iff two annotations have overlapping mentions. In other words, we search for the subset of non-overlapping annotations that maximizes F_1.

Both SMAPH-S and SMAPH-3 use statistical machine learning to find an approximate solution to the subset selection problem, but they follow different approaches:

SMAPH-S builds the solution by judging candidate annotations independently, so it can model the mention-entity link strength but not the coherence among annotations;

SMAPH-3 incrementally builds the solution, deciding at each iteration whether to add any of the candidate annotations to the current solution or to terminate, so it can also model the coherence between annotations included in the solution.

7 SMAPH-S: SINGLE ANNOTATION JUDGMENT

The SMAPH-S algorithm solves the subset selection problem above in two steps: candidate annotations are first independently assigned a score representing the likelihood that they are correct, then the solution is built by choosing non-overlapping annotations, giving priority to those with higher score.

Algorithm 1: SMAPH-S

```
  input  : The query $q$ to annotate.
  output : A set of annotations for $q$.
1 function $smaph\_s(q, \mathcal{R}, t)$ :
2     $C_q \leftarrow \{c \in get\_candidates(q) \ : \ \mathcal{R}(c) \geq t\}$ ;
3     $A \leftarrow sort(C_q, \mathcal{R})$ ;
4     $P \leftarrow \varnothing$ ;
5     for $i = 0, \ldots, (|A| - 1)$ do
6         if $\forall p \in P, \ \neg K(A_i, p)$ then
7             $P \leftarrow P \cup \{A_i\}$ ;
8     return $P$ ;
```

Algorithm 1 shows the pseudo-code of SMAPH-S. Candidate annotations are first assigned a score by a regressor $\mathcal{R}$ (we detail the training of $\mathcal{R}$ in Section 7.1). The higher the score, the more likely an annotation is correct. Annotations with a score lower than the threshold t are discarded (line 2). The others are sorted by their score in descending order (line 3). Let A be this ordered list, the first element A_0 will be the annotation with highest score, A_1 the second best, etc. The solution P (initially empty) is built scanning annotations in A by descending score (line 5), and adding to P the annotations that do not overlap (line 6) with any annotation previously added to P. This guarantees that, in case A contains overlapping annotations, only the one with higher score is kept.

The choice of building such a regressor, as opposed to building a classifier like we did for SMAPH-1, comes from the need of the algorithm to choose between two conflicting annotations via their ranking. $\mathcal{R}$ is a key component of the algorithm, and its accuracy in assigning scores is central, as it defines a policy to resolve conflicts among annotations and a criterion for terminating the annotation process, discarding low-ranked annotations. As shown in the pseudocode, SMAPH-S cannot take into account the coherence of the solution, as annotations are judged independently.

7.1 Scoring of candidate annotations via SVR and choice of threshold

Given an annotation $a = (m, e)$, regressor $\mathcal{R}$ is trained to predict the likelihood that mention m refers to entity e (i.e. the likelihood that annotation a is a correct). This likelihood is expressed as a real number: the higher, the more likely a is correct. Hence $\mathcal{R}$ is a function in the form $C_q \mapsto \mathbb{R}$.

The algorithm we use for regression (i.e. the process of finding regressor $\mathcal{R}$) is support vector regression (SVR). Training examples are candidate annotations C_q for all queries q in the training set. A candidate annotation is considered a *positive* example, and thus assigned a score of 1, if it appears in the ground truth, otherwise it is considered a *negative* example and thus assigned a score of -1. As already observed, training examples are heavily unbalanced towards negative ones. This is not a problem as SVR supports unbalanced training sets. We

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

Table 7. Features of a candidate annotation (m, e) (used by SMAPH-S and SMAPH-3), where m is a mention (list of query terms) and e is an entity.

associate a feature vector to each annotation, as detailed later. Training parameters, the threshold t and the features composing the feature vector are chosen with the objective of maximizing the macro-F_1 achieved by SMAPH-S on the development set, which is distinct from the training set. In Section 9.4 we report details on the hyperparameters resulting from the training procedure and on the generated model.

7.2 Annotation features

The set of features that constitute the feature vector for each annotation $a_i = (m_i, e_i) \in C_q$ are (i) those used by SMAPH-1, modeling properties relative to the entity e_i only (Features 1–24 in Table 6), concatenated with (ii) a set of five new features that capture aspects of the *binding* between mention m_i and entity e_i (Features 25–29 in Table 7). While features 1–24 model the quality of the candidate entity, features 25–29 model the strength of the bond between the candidate entity and its mention. 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 7 uses, in addition to the definitions introduced in Section 5, the following definitions:

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

Let us discuss the rationale behind these features. Feature 25 (*anchorsAvgED*) is the average edit distance between mention m_i and all anchor texts contained in Wikipedia articles that link to the Wikipedia article about e_i. These anchors are forms in which entity e_i can be referred to. Edit distances are weighted with respect to the number of times the Wikipedia artice about e_i is referenced by an anchor (the square root mitigates the effect of high-frequency anchors). The more times e_i has been referenced by anchors similar to m_i, the larger *anchorsAvgED* will be, hence a high value of this feature suggests that mention m_i refers to e_i. Features 26 and 27 aim at measuring the string similarity between mention m_i and the title of the Wikipedia article about the candidate entity e_i: *edTitle* is simply their edit distance, while *minEdTitle* is the minimum word-to-word edit distance[9]. Lower values of Features 26 and 27 indicate a higher similarity between mention m_i and title of entity e_i, and thus a stronger bond between the two. Feature 28 (*commonness*) is another measure for the mention-entity bond strength: the more frequently m_i refers to e_i (as opposed to other entities) in Wikipedia, the stronger is the bond. Feature 29 (*lp*) is instead a measure of how likely mention m_i is actually a mention of anything. A higher value of *lp* indicates that the whole annotation is more likely to be correct.

[9] See definition at Section 5.2

7.3 Notable excluded features

The set of features employed by the SMAPH entity-annotators (those described in the previous section) is the result of forward feature selection on a much broader set of features. Features were not considered if they did not increase macro-F_1 on the development set. Similarly to the features that were selected, the ones that were not selected consider other aspects of the relation between an entity and a query. It would be uninteresting to describe all of them in detail, but it is worth mentioning that, among the tested features that were excluded, we considered the scores provided by the models of [3]. These models, based on word embeddings generated with `word2vec`, measure the similarity between the word embeddings of the query terms and the word embeddings of the first paragraph of the Wikipedia article about e. They model the context of a query mention as the word embeddings of surrounding terms and exploit that context for disambiguating the mention. Though this score works well if used alone (see [3]), it does not offer to our model any additional information to decide whether an entity is mentioned by a query or not. A reason for that might be that word embeddings do not provide any piece of information that context given by the snippets does not offer already. In other words, snippets seem to offer a larger (and, experimentally, more precise) context that WAT can leverage to perform entity disambiguation.

8 SMAPH-3: GREEDY ITERATIVE SOLUTION ENRICHMENT

SMAPH-3 incrementally builds a solution by judging the contribution of each annotation with respect to the others. This gives us a fine-grained model of annotation selection while also keeping in consideration the coherence of the solution.

The core of SMAPH-3 is a greedy algorithm that iteratively selects candidate annotations to be added to the solution. The process starts from an empty *solution.* At each iteration, it gives an answer to the question: given the solution found so far, which is, among the candidate annotations, the one that most likely improves the current solution? If such an annotation exists, it is added to the solution, otherwise the algorithm terminates and the current solution is returned. It follows that at iteration i, the algorithm will either increase the solution size to $i + 1$ by adding a new annotation, or it will return a solution consisting of i annotations.

The decision on which annotation to add (if any) at iteration i is made by a per-iteration regressor $\mathcal{R}_i$ and an associated threshold t_i. More precisely, let a be the annotation among the candidates that gets the highest score assigned by $\mathcal{R}_i$, then if this score is higher than t_i, a is added to the solution, otherwise the algorithm terminates returning the solution obtained by the previous iterations (which does not include a). It is important to note that at each iteration, a different regressor-threshold pair $< \mathcal{R}_i, t_i >$ makes the decision about either which annotation to add, or to add none and terminate.

SMAPH-3's training procedure is more complex than the other SMAPH versions. To better understand it, we will follow a top-down approach, explaining first how the regressor-threshold pairs are employed to build a solution, and then, in Section 8.1, how they are trained. For now, let us assume we have one regressor-threshold pair $\langle \mathcal{R}_i, t_i \rangle$ for each iteration $i = 0, \dots, n - 1$ (the number of iterations n is determined at training time). These regressors take as input two arguments: the current solution P (a set of annotations) and a candidate annotation a to be added to the current solution, and output a real number indicating the likelihood that $F_1(P \cup \{a\}) > F_1(P)$, in other words, the likelihood that adding a to P would increase the F_1 score of the current solution, as opposed to decrease it. Threshold t_i is interpreted to mean that it is worth adding a to the current solution if, and only if, $\mathcal{R}_i(P, a) \geq t_i$.

We now describe the pseudo-code of SMAPH-3 as shown in Algorithm 2. It starts from an empty solution P and a set of candidate annotations C which is initially set to the Cartesian product $C_q = Seg(q) \times \mathcal{E}_q$ defined before (lines 2–3). At each iteration, SMAPH-3 uses a different regressor to judge the opportunity of adding each candidate annotation $c \in C$ to the current solution P. It is important to note that how an annotation c is judged depends on P (lines 7-8). Considering the annotation a that gets the highest score from the regressor (line

```
Algorithm 2: SMAPH-3
  input  : The query $q$ to annotate.
  output : A set of annotations for $q$.
1  function smaph3($q$, $\langle \mathcal{R}_0, \ldots, \mathcal{R}_{n-1} \rangle$, $\langle t_0, \ldots, t_{n-1} \rangle$) :
2      $P \leftarrow \emptyset$ ;
3      $C \leftarrow get\_candidates(q)$ ;
4      for $i \leftarrow 0, \ldots, n-1$ do
5          if $|C| = 0$ then
6              return $P$ ;
7          $a = \arg\max_{c \in C} \mathcal{R}_i(P, c)$ ;
8          if $\mathcal{R}_i(P, a) < t_i$ then
9              return $P$ ;
10         $P \leftarrow P \cup \{a\}$ ;
11         $C \leftarrow \{c \;:\; c \in C \;\wedge\; \neg K(c, a)\}$ ;
12     return $P$ ;
```

7), if that score is lower than the threshold, it means that no annotations would improve the solution. In this case, the algorithm terminates by returning P (line 9). Otherwise, annotation a is added to the current solution P (line 10), and the candidates that overlap with a are removed from C so that they will not be considered in the next iterations (line 11). This way, at the beginning of each iteration, C will only contain annotations that do not overlap with any annotation contained in P, and thus can be added without violating the constraint K. The algorithm then proceeds to the next iteration. The algorithm terminates either (i) at iteration i, if adding any candidate annotation would not improve the current solution, (ii) at iteration i, if there are no more candidates (i.e., all tokens of the query are linked to an entity or set C lacks in coverage), or (iii) after all n iterations have been executed.

At the beginning of the ith *while* iteration (starting from $i = 0$), the current solution P has size i, and $\mathcal{R}_i$ determines which is the $(i + 1)$th annotation to be added to P, if any. As the reader might have noticed, the algorithm must make decisions based on scenarios that greatly differ from iteration to iteration. This is the reason why we use per-iteration regressor-threshold pairs trained to make decisions in the specific scenarios in which they are called.

As an example, let $q =$ `armstrog mon landign` be the query to annotate, and let C be the set of candidate annotations for that query:

$$C = \left\{ \begin{array}{l} a_1 = \texttt{mon landign} \mapsto \textit{Moon\_Landing} \\ a_2 = \texttt{mon} \mapsto \textit{Moon} \\ a_3 = \texttt{armstrog} \mapsto \textit{Louis\_Armstrong} \\ a_4 = \texttt{armstrog} \mapsto \textit{Neil\_Armstrong} \\ \ldots \end{array} \right\}$$

Before executing iteration 0, the current solution P is empty, and SMAPH-3 has to decide whether to add an annotation to P, or to return the empty solution. Regressor $\mathcal{R}_0$ assigns to each annotation in C a score representing the likelihood that it is correct. Say the annotation with the highest score is a_1, and this score is $\mathcal{R}_0(P, a_1) = 0.6$. Shall we take the risk of adding this annotation to the solution (at the risk of a false positive), or should we be conservative (at the risk of a false negative)? This question is answered by checking if a_1's score is higher than threshold t_0. In case it is, iteration 0 ends by adding annotation a_1 to the current solution. Annotations overlapping

Algorithm 3: SMAPH-3 training

```
input  : Training dataset $D_t$; Development dataset $D_d$.
output : A list of regressors $\mathcal{R}_i$ and associated thresholds $t_i$ (one for each iteration).
function train_smaph3($D_t, D_d$) :
    for $q \in D_t$ do
        $P_q \leftarrow \varnothing$ ;
    $i \leftarrow 0$ ;
    while true do
        /* Terminate if no improvement is possible.                          */
        if $(\forall q \in D_t,\ \nexists c \in C_q\ :\ TP(c))\ \vee\ (\forall q \in D_d,\ \nexists c \in C_q\ :\ TP(c))$ then
            return $\langle R_0, \ldots, R_{i-1} \rangle, \langle t_0, \ldots, t_{i-1} \rangle$
        /* Gather examples.                                                  */
        $E \leftarrow \varnothing$ ;
        for $q \in D_t$ do
            if $|P_q| = i$ then
                for $c \in C_q$ do
                    $v \leftarrow$ gen_ftr_vec$(c, P_q)$ ;
                    $l \leftarrow F_1(P_q \cup \{c\}) - F_1(P_q)$ ;
                    $E \leftarrow E \cup \langle v, l \rangle$ ;
        /* Train regressor.                                                  */
        $R_i, t_i \leftarrow$ train_and_optimize_param$(E, D_d)$ ;
        /* Terminate if no improvement at current iteration.                 */
        if $F_1(M(\langle R_0, \ldots, R_i \rangle, \langle t_0, \ldots, t_i \rangle)(D_d) \le F_1(M(\langle R_0, \ldots, R_{i-1} \rangle, \langle t_0, \ldots, t_{i-1} \rangle)(D_d))$ then
            return $\langle R_0, \ldots, R_{i-1} \rangle, \langle t_0, \ldots, t_{i-1} \rangle$
        /* Update current solutions.                                         */
        for $q \in D_t$ do
            if $|P_q| = i$ then
                $a \leftarrow \arg\max_{c \in C} \mathcal{R}_i(P_q, c)$ ;
                if $\mathcal{R}_i(P, a) \ge t_i$ then
                    $P_q \leftarrow P_q \cup \{a\}$ ;
                    $C_q \leftarrow \{c\ :\ c \in C_q\ \wedge\ \neg K(c, a)\}$ ;
        $i \leftarrow i + 1$
```

with a_1 (a_2 and a_1 itself) are removed from C and will not be considered in the next iterations. At iteration 1, SMAPH-3 has to decide whether to return $P = \{a_1\}$ as is, or to expand it by adding an annotation in $C = \{a_3, a_4\}$ that would improve F_1. A regressor other than $\mathcal{R}_0$ is needed because at this iteration we must also consider the coherence with the annotation added to P at the previous iteration (in the example, a_1). Annotation a_4 has a strong semantic relatedness with a_1, so it receives the highest score, say $\mathcal{R}_1(P, a_4) = 0.3$. If the threshold for this iteration is $t_1 = 0.1$, then a_4 is added to P and a_3, a_4 are removed from C. Since there are no more candidates (C is empty), the algorithm terminates returning as solution for q the set $P = \{a_1, a_4\}$.

8.1 Coherence judgment on candidate annotations through SVR

Similarly to the other versions of SMAPH, regressors $\mathcal{R}_i$ are trained with examples drawn from a training set while their hyperparameters and the threshold t_i are chosen to maximize the macro-F_1 on a distinct development set. Per-iteration regressors are built through support-vector regression (SVR). Regressor $\mathcal{R}_i$ (the regressor that ranks candidates at iteration i) is trained to make decisions in the scenario in which it will be invoked. This means that (i) the training examples it is trained with are only the candidate annotations that are still included in C_q at iteration i, (ii) training examples are only gathered for queries that reach iteration i (i.e., the queries for which SMAPH-3 would provide a solution consisting of at least i annotations, and has to decide the $(i+1)$th annotation to include in the solution, if any), and (iii) a training example associated to a candidate annotation is labeled with the improvement that the annotation would bring to the solution found at iteration $i-1$.

Training is done in cascade: at training time, after $\mathcal{R}_i$ is built, it is invoked to choose which (if any) candidate annotation to add to the current solution for the queries of the training set. As explained, this choice will influence the training of the regressors for the next iterations.

The pseudo-code for the training of SMAPH-3 is presented in Algorithm 3. The ith iteration of the *while*-loop generates $\mathcal{R}_i$ and finds t_i. Lines 6-7 implement the first stop condition: if, in either the training or development datasets, there are no queries for which exists a correct annotation among their candidates ($TP(x)$ indicates that x is a true positive), it means that no query may possibly benefit from iteration i. In this case, we return a model that will execute $i-1$ iterations. In the opposite case, we gather the examples for iteration i (lines 9-14) and train the regressor (line 15). Note that training examples are labeled with the difference in F_1 between the solution with and without a (line 13). Examples are gathered for all queries of the training set having a solution of size i, namely those for which the ith regressor would be invoked. For each candidate annotation c, function `gen_ftr_vec` generates a feature vector and the label associated to it. Note that it takes the current solution P_q as argument: this is necessary to compute the features about the coherence between c and the current solution P_q. The features computed by `gen_ftr_vec` are detailed in the next section. Function `train_and_optimize_param` trains regressor $\mathcal{R}_i$ with the training examples gathered in E. The threshold t_i is selected to maximize macro-F_1 on the development set. Similarly, features are selected by forward feature selection, maximizing macro-F_1 on the development set. For this purpose, function `train_and_optimize_param` will execute a partial SMAPH-3 model consisting of all iterations $0, \ldots, i$ on the queries of dataset D_d, and check its output against the gold standard for those queries. Threshold t_i is chosen through a fixed-width scan of values between the lowest and highest value output by $\mathcal{R}_i$ for the example in E: among the values that maximize macro-F_1, the middle one is chosen.

The second stop condition (lines 16-17) checks if iteration i brought an improvement in terms of macro-F_1 on the development set. If it didn't, the algorithm terminates returning a model consisting of iterations $0, \ldots, i-1$, while regressor $\mathcal{R}_i$ is discarded.

The last block of pseudo-code (lines 18-23) applies the current iteration to the queries of the training set. This consists of updating their solutions by adding the annotation a chosen by regressor $\mathcal{R}_i$ (if any) and removing from the candidate set the annotations conflicting with a (including a itself). Note that this is done only for queries having a solution of size i, namely those on which the ith iteration would be executed. Moreover, the current solution is updated by adding annotation a only if it had been assigned a score above the threshold in the current iteration i (condition at line 21). This guarantees that at any given iteration, the regressor will be trained considering only queries for which that iteration would be reached.

Note that the training algorithm always terminates: for each query in the training set, eventually, either the top annotation will be assigned a score that is below the threshold, or the set of candidate annotations C_q will become empty.

8.2 Annotation and coherence features for SMAPH-3

Each pair consisting of a candidate annotation $a = (m, e)$ and a current solution P is associated with a feature vector $F(a)$ that consists of three groups of features:

Entity features model the strength of entity e, and are mainly based on signals derived from the search engine. They are 24 features computed over entities e, see Table 6.

Annotation features model the strength of the bond between mention m and entity e. They are 5 features computed over annotation (m, e), see Table 7.

Coherence and coverage features model the strength of the decision to add annotation a to the solution P found by the algorithm in the previous iterations. They are 15 features computed by taking into account an annotation (m, e) and a current solution P, see Table 8.

With the third group of features we can finally reach the goal of modeling how entities forming a solution are related to each other. In fact, a high relatedness might indicate a good solution. We also model how favorable it is to add annotation a to P as opposed to returning P. This is the reason why some of the features model the differences between P and $P \cup \{a\}$. Let us now describe the rationale behind each *coherence feature*, keeping in mind that SVR captures, at least to some level, the interdependence of features, whose effectiveness must therefore be considered in combination with others. To model the semantic coherence of the solution, we compute the relatedness between entity e and each entity included in the current solution (e' such that $(m', e') \in P$). We take the minimum, maximum and average of them as features 30, 31, and 32. We also compute the same measures for the current solution P (Features 33, 34, 35) and the difference between each measure computed on the solution P and on the solution $P \cup \{a\}$ (Features 36, 37, 38). The reason why we include features about the solution *before* and *after* the current iteration, and the difference between them, is that they are crucial to decide what annotation to add, if any: for example, if current solution P has a very high minimum relatedness (Feature 30), meaning that all entities cited by a query are semantically related, but adding $\{a\}$ would decrease the minimum relatedness (Features 33, 36), this may suggest that a is a false positive. On the other hand, an increment in maximum relatedness (Features 32, 35, 38), might indicate a true positive. The relatedness between two entities is computed as the *Milne-Witten relatedness*, a measure that takes into account the overlap of incoming links in the Wikipedia articles about the two entities (see [39]).

The remaining features aim at modeling the annotation coverage, namely how many tokens of the query should be part of a reference to an entity. The two main measures for this are the number of tokens being part of a mention (which measures how much the query is covered by a solution, Features 39–41), and the link probability of the mention[10], which models how much it *should* be covered (Features 42–44). Similarly to relatedness features, coverage is computed for the would-be solution $P \cup \{a\}$ (Feature 39), and as the increment of coverage that adding a would bring, in absolute (Feature 41) and relative (Feature 40) terms.

These two groups of features are of particular importance for estimating the recall of a solution and, intuitively, how hazardous it is for SMAPH-3 to add annotation a to the solution: if a query has few terms covered (Feature 39), but a very high link probability sum (Features 42, 43), then the solution probably lacks recall, and it is probably better to add a to the current solution P, than to be conservative and decide to return P as a solution. Moreover, if the mention spans multiple tokens (Features 40, 41), SMAPH-3 may decide to add an annotation even though it has a weak mention-entity bond.

9 EVALUATION OF SMAPH

It is important to point out that the performance of SMAPH heavily depends on the quality of results delivered by the underlying search engines, which depend on many aspects of their structure, including the user-generated data they possess (query logs, click logs), external resources (the Web), internal resources (knowledge bases),

[10] See definition at Section 5.2.

ID	Name	Definition
		$R_a := \{rel(e,e') \ : \ (m',e') \in P\}$
30	rel_{min}	$\min(R_a)$
31	rel_{max}	$\max(R_a)$
32	rel_{avg}	$\text{avg}(R_a)$
		$R_P := \{rel(e',e'') \ : \ (m',e'),(m'',e'') \in P \wedge e' \neq e''\}$
33	$prel_{min}$	$\min(R_P)$
34	$prel_{max}$	$\max(R_P)$
35	$prel_{avg}$	$\text{avg}(R_P)$
36	Δrel_{min}	$\min(R_a) - \min(R_P)$
37	Δrel_{max}	$\max(R_a) - \max(R_P)$
38	Δrel_{avg}	$\text{avg}(R_a) - \text{avg}(R_P)$
39	$covg$	$\sum_{(m,e)\in P\cup\{a\}}(\lvert m\rvert)/\lvert q\rvert$
40	$\delta covg$	$\lvert m\rvert/\lvert q\rvert$
41	$\Delta covg$	$\lvert m\rvert$
42	$\Sigma seglp$	$\sum_{s\in\ Seg(q)} lp(s)$
43	$seglp_{avg}$	$\text{avg}_{s\in\ Seg(q)}\ lp(s)$
44	$seglp_{ratio}$	$\sum_{s\in\ Seg(q)} lp(s)/(\lvert P\rvert + 1)$

Table 8. Features of a candidate annotation $a = (m, e)$, given the current solution P, for query q (used by SMAPH-3). $rel(a, a')$ is the Milne-Witten relatedness [39] among entities e and e' (where $a' = (m', e')$).

their variation over time, and, above all, the algorithm used for ranking web results. It is practically impossible to isolate the contribution of each of these components to the quality of SMAPH results, and we are constrained to treating the whole search engine as a black box. Nonetheless, results will show that the methods proposed by us are applicable on popular search engines such as Google and Bing.

9.1 Evaluation metrics

As evaluation metrics we use those proposed by the BAT-Framework [9, 53]. Recall that SMAPH-1 addresses query-level entity-linking, while SMAPH-S and SMAPH-3 address token-level entity-linking. Any token-level entity-linking can be simplified to a query-level entity-linking by discarding the mentions included in the solution and keeping the entities only, hence all SMAPH versions can be tested with respect to their ability to solve query-level entity-linking while only SMAPH-S and SMAPH-3 can be tested for token-level entity-linking.

For a deeper discussion on the match relations proposed by the BAT-Framework, see [8]. We recall the basics of those metrics through an example of their computation over a single query. Let

$$q = \texttt{armstrong mon lading}$$

and let the corresponding ground truth solution $S^* = \{a_1^*, a_2^*\}$ for token-level entity-linking be composed of two annotations:

$$a_1^* = \texttt{armstrong} \mapsto \textit{Neil\_Armstrong}$$
$$a_2^* = \texttt{mon lading} \mapsto \textit{Moon\_Landing}$$

indicating that the first term is a mention of the astronaut, while the second and third terms form a single mention of the historical event of landing on the moon[11]. Let $\bar{S} = \{\bar{a}_1, \bar{a}_2, \bar{a}_3\}$ be the solution given by an entity-annotator:

$$\begin{aligned}\bar{a}_1 &= \texttt{armstrong} \mapsto \textit{Neil\_Armstrong}\\ \bar{a}_2 &= \texttt{mon} \mapsto \textit{Moon}\\ \bar{a}_3 &= \texttt{lading} \mapsto \textit{Moon\_Landing}\end{aligned}$$

Annotation $\bar{a}_1$ is a true positive (TP); $\bar{a}_2$ and $\bar{a}_3$ are false positives (FPs); a_2^* is a false negative (FN). Counting them, we have

$$|tp(S^*, \bar{S})| = 1 \qquad |fp(S^*, \bar{S})| = 2 \qquad |fn(S^*, \bar{S})| = 1$$

yielding query-wise metrics:

$$P(S^*, \bar{S}) = \frac{1}{3} \qquad R(S^*, \bar{S}) = \frac{1}{2} \qquad F_1(S^*, \bar{S}) = \frac{2}{5}$$

We define F_1 to be 1.0 if both the ground truth and the provided solution are empty[12].

The quality of the solutions found by an entity-annotator on the queries provided by a dataset is measured by computing dataset-wise precision, recall, and F_1 with both *micro* and *macro* weighting schemes. We refer to dataset-wise metrics as, e.g., macro-P and micro-P. A macro-measure is defined as the arithmetic average of a query-wise metric across the queries of a dataset. For example, macro-P is defined as

$$\text{macro-}P(D) = \frac{1}{|D|} \cdot \sum_{q \in D} P(S_q^*, \bar{S}_q)$$

where D is a dataset, S_q^* is the ground truth for query q, and $\bar{S}_q$ is the solution provided by an entity-annotator for query q.

Micro-measures are instead defined as a query-wise metric that considers the count of TP, FP and FN across all queries of a dataset. For example, micro-P is defined as

$$\text{micro-}P(D) = \frac{\sum_{q \in D} |tp(S_q^*, \bar{S}_q)|}{\sum_{q \in D} |\bar{S}_q|}$$

As a consequence, macro-measures evaluate the performance of an entity-annotator on a new single query; they are a better fit for measuring the performance in scenarios in which one query at a time has to be annotated (e.g., a single search on a search engine), while micro-measures are a better fit for scenarios in which aggregations of queries have to be annotated (e.g., query trend analysis).

For queries where either the ground truth or the solution found by the entity-annotator is small, the number of TPs, FPs and FNs tends to be small. Micro-measures therefore do not capture how well an entity-annotator handles such cases, since the count of TPs, FPs and FNs are aggregated over all queries. For this reason, we regard macro-measures as the appropriate evaluation: a correctly handled query with empty ground truth will contribute an F_1 score of 1.0 to the macro average and thus good performance will be rewarded. However, for completeness, we also report micro-measures.

[11] Note that entity *Moon_Landing* entails the entity *Moon*, hence *Moon* is not expected to be part of the result, as per our problem definition (Section 1.1).

[12] As should be clear, F_1 is 0.0 if the ground truth is empty, but the provided solution is not; and F_1 is 0.0 if the provided solution is empty, but the ground truth is not.

9.2 Experimental setting

9.2.1 Tested entity-annotators. In our experiments we test the following entity-annotators:

WAT is the improved version of TagME introduced in [45] for token-level entity-linking. As relatedness function in the disambiguation process we use the Jaccard similarity among in-links, because it performed best on GERDAQ[13].

AIDA is the token-level entity-annotator introduced in [27], we downloaded the code from the official web site[14]. 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 entity-annotator for query-level entity-linking (introduced in [24]) 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 query-level entity-linking 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 query-level entity-linking in [15]) uses Wikipedia and DBpedia to generate candidate annotations, then builds a graph of mentioned entities exploiting the link structure of Wikipedia. The disambiguation step is based on PageRank over this graph and assigns a score to each entity.

Tan et al. is a query entity-annotator introduced by Tan et al. [52] that performs a search of the query in the body of Wikipedia articles, treating search results as candidate entities. These candidates are ranked by means of a linear model that employs features that take into account (i) the quality of the match between query and the body of the Wikipedia article, (ii) the relatedness of the candidate towards other candidates, and (iii) towards the entities directly linked by the article. Given that the entity-annotator has not been published, we cannot experiment with it, and will only report the measures appearing in [52].

SMAPH-1 (Section 5) addresses query-level entity-linking.

SMAPH-S (Section 7) is our first proposal for token-level entity-linking. It evaluates each mention-entity pair individually.

SMAPH-3 (Section 8) is our final entity-annotator that addresses token-level entity-linking by greedily building the solution, adding one annotation at a time, and considering its coherence with respect to previously-added annotations.

The first two entity-annotators (AIDA and WAT) are the baselines for query-level and token-level entity-linking. Other entity-annotators employed here are the top-ranking entity-annotators of the ERD Challenge.

The scores obtained by SMAPH-1 and SMAPH-S are slightly higher than those reported in our previous work [10, 11]. The difference in performance is due to bug fixes in the code[15].

9.2.2 Evaluation datasets. Our experiments have been conducted on three datasets. We briefly summarize their main characteristics.

ERD-online. The dataset used in the ERD Challenge to test the entity-annotators solving query-level entity-linking [5]. The entity knowledge base is a subset of Freebase, namely only its NamedEntities. It consists of 500 annotated queries. The ERD Challenge dataset is not available off-line. In order to enable challenge participants to test their entity-annotators on this dataset without revealing its ground truth, the challenge organizers built an online evaluation platform. The platform, and consequently the

[13] Note that we also employ WAT to annotate snippets in SMAPH's candidate entity generation phase (Section 4). Here instead, we are testing how it performs when applied directly on queries.

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

[15] More specifically, the bug caused anchors from Wikipedia to not be correctly parsed, resulting in incorrect values for features 23 and 24 in Table 6 and feature 25 and 26 in Table 7.

dataset, is currently not accessible. For this reason, we cannot test our most recent proposal on this dataset, and results refer to old runs from the time it was accessible. The evaluation performed by the ERD Challenge platform measures the quality of the entities found by an entity-annotator, but does not take into account the mentions. For this reason, entity-annotators can be tested on this dataset only with respect to the problem of query-level entity-linking. The query's ground truth remains unknown, so no error analysis can be carried out. This makes the evaluation against this dataset a real third-party check of the robustness of entity-annotators. For detailed information about the creation of the ERD-online dataset, see [5].

ERD-dev. The dataset provided by the ERD Challenge organizers as a development set for challenge participants. It provides a token-level entity-linking ground truth for 91 queries. Similarly to the ERD-online dataset, the entity knowledge base consists of NamedEntities.

GERDAQ. This is the novel dataset we have built via CrowdFlower. It provides a token-level entity-linking ground truth for 992 queries. The entity knowledge base is the whole Wikipedia, hence it includes all three types of GenericEntities enumerated at the beginning of Section 1, not just NamedEntities. GERDAQ is split into three portions: train (497 queries), development (249 queries), and test (246 queries). Results will be given for the test portion.

9.3 Coverage of entity sources

We first evaluate the coverage of the *candidate entity generation* process. As explained in Section 4, this step (shared among all the three SMAPH entity-annotators) is the only one that aims at discovering entities that may be mentioned by the query. After candidate entities are generated, some of them can be discarded, but none can be added. For this reason, its performance fixes an upper bound on the recall of our entity-annotators.

We evaluate the coverage of the three entity sources which, we remind the reader, were defined as follows:

- $\mathcal{E}_1$ are Wikipedia articles appearing as search results for the input query q;
- $\mathcal{E}_2$ are Wikipedia articles appearing as search results for the query q + `wikipedia`;
- $\mathcal{E}_3$ are entities found by annotating through WAT the snippets appearing in search results for query q.

The SMAPH entity-annotators employ the union of these entity sources $\mathcal{E}_q = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$. We also remind the reader that each entity source comes with a set of features associated with an entity (see Table 6) that assist later steps of the algorithms.

Table 9 reports, for each entity source, the average coverage and precision computed over the queries of dataset GERDAQ-dev. Read the leftmost column of the table as "when piggybacking on Bing, entity source $\mathcal{E}_1$ provides 20.2% coverage of the ground truth GenericEntities and 58.6% of the ground truth NamedEntities, with a precision of 91.0% for GenericEntities and 97.0% for NamedEntities". Coverage indicates how many entities forming the ground truth are provided by a source, while precision indicates how many entities provided by a source are contained in the ground truth. We can note a few facts:

- Source $\mathcal{E}_3$ is the largest single source of entities, though it has low precision.
- Entities in $\mathcal{E}_1$ are included in $\mathcal{E}_2$ ($\mathcal{E}_1 \subseteq \mathcal{E}_2$) because Wikipedia articles that appear as result when searching for q also appear when searching for q + `wikipedia`. This may wrongly suggest to the reader that $\mathcal{E}_1$ can be discarded, as it does not extend the coverage. But we observe that $\mathcal{E}_1$ has higher precision than $\mathcal{E}_2$; thus, the fact that e is included in both $\mathcal{E}_1$ and $\mathcal{E}_2$ gives a stronger signal of correctness than the case in which it is only included in $\mathcal{E}_2$ (this is captured by Feature 3, see Table 6).
- Merging all sources together increases the coverage of plain $\mathcal{E}_3$ by 1.3% (Bing) and by 3.3% (Google) on GenericEntities, while the coverage of NamedEntities is increased by 0.8% (Bing) and by 0.4% (Google).
- The coverage reached by the union $\mathcal{E}_q$ is rather good for both GenericEntitites (87.9% for Bing, 85.4% for Google) and for NamedEntities (96.7% for Bing, 94.4% for Google). An ideal entity-annotator built on top

	Bing						Google					
	$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_3$	$\mathcal{E}_1 \cup \mathcal{E}_3$	$\mathcal{E}_2 \cup \mathcal{E}_3$	$\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$	$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_3$	$\mathcal{E}_1 \cup \mathcal{E}_3$	$\mathcal{E}_2 \cup \mathcal{E}_3$	$\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$
C_{GE}	20.2	42.1	86.6	86.8	87.9	87.9	19.3	38.6	82.3	82.7	85.4	85.4
P_{GE}	91.0	32.5	11.3	11.3	10.4	10.4	91.4	22.5	23.3	22.8	17.6	17.6
C_{NE}	58.6	76.2	95.9	96.3	96.7	96.7	59.2	69.9	94.0	94.4	94.4	94.4
P_{NE}	97.0	52.8	13.2	13.4	11.9	11.9	94.8	43.2	35.7	35.2	26.8	26.7

Table 9. Average coverage and precision of the entity sources $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ on the GERDAQ-dev dataset. Coverage and precision is given for both the set of NamedEntities (C_{NE}, P_{NE}) and the set of GenericEntities (C_{GE}, P_{GE}).

of these sources, i.e., an entity-annotator that keeps all correct candidate entities and discards all wrong ones, would achieve an impressive F_1 score of around 93% for GenericEntities.

9.4 Training process

All models were trained on GERDAQ-train (497 queries). The piggybacked search engine is either Google or Bing. Features were selected by forward variable selection on GERDAQ-dev (249 queries). Hyperparameters of the SVM models were tuned on GERDAQ-dev, by means of a grid search, maximizing macro-F_1[16].

SMAPH-1's binary classifier (Section 5.1) was trained with 757 positive and 5,567 negative examples (Google) or with 757 positive and 8,496 negative examples (Bing). This means that the set of candidate entities $\mathcal{E}_q$ consists on average of 1.5 correct vs. 11.2–17.1 wrong candidate entities. Macro-F_1 on GERDAQ-dev is 59.44% (Google) and 58.49% (Bing).

SMAPH-S's regressor was trained with 736 positive and 40,803 negative examples (Google) or with 736 positive and 51,171 negative examples (Bing). This implies that the candidate set C_q consists on average of 1.5 correct vs. 82–103 wrong candidate annotations. Macro-F_1 on GERDAQ-dev is 58.03% (Google) and 57.13% (Bing).

SMAPH-3's regressors were trained with a different number of examples at each iteration. For example, when piggybacking on Google, the first iteration regressor is trained with a total of 41,539 vectors, 40,288 of which (97%) are assigned label 0.0, meaning that adding the associated annotation would bring no improvement to the solution, while 116 vectors are assigned label 1.0, meaning that for 116 queries the ideal solution (consisting of one annotation) can be found at the first iteration. With Bing, the number of vectors is 51,907, with similar proportions. The number of iterations (i.e., regressor-threshold pairs) generated by the training process was five for Google and four for Bing. Thresholds t_i for both Bing and Google range in $[-0.03, 0.03]$ across iterations. For a deeper discussion on the models of SMAPH-3, see Section 9.7. Macro-F_1 on GERDAQ-dev is 59.21% (Google) and 61.54% (Bing).

[16] We cannot give a detailed introduction to SVMs here, but for the interested reader we provide the following details about the SVM models and their training. The Support Vector Machines employed by all versions of SMAPH use RBF as kernel function, as this proved to be the one that better exploits the features. The hyperparameters for the training process of SVM are γ (the free parameter of the RBF function) and C (the margin between the hyperplanes). SMAPH-S also has hyperparameters w_-, w_+ (the objective value assigned respectively to bad and good annotations) and t (the threshold on the predicted score, see Section 7).

SMAPH-1. The tuning process took 25 minutes on a consumer personal computer. Value of hyperparameters: $C = 1.0$, $\gamma = 0.01998$ (Google) and $C = 1.0$, $\gamma = 0.02500$ (Bing). Number of support vectors employed by the model: 1,683 (Google), 1,799 (Bing). **SMAPH-S.** The tuning process took less than 3 hours. Value of hyperparameters: $\gamma = 0.01190$, $C = 0.62366$, $w_- = 0.61852$, $w_+ = 2.02761$, $t = -0.85796$ (Google) and $\gamma = 0.01700$, $C = 0.78936$, $w_- = 0.94000$, $w_+ = 3.34372$, $t = -0.79074$ (Bing). Number of support vectors employed by the model: 1,841 (Google), 2,090 (Bing). **SMAPH-3.** Number of support vectors for Google: 3,758 (iteration 1), 2,766 (iteration 2), 568 (iteration 3), 39 (iteration 4), 8 (iteration 5). Number of support vectors for Bing: 5,261 (iteration 1), 3,073 (iteration 2), 1,041 (iteration 3), 42 (iteration 4).

9.5 Benchmark results

9.5.1 Experiment #1: Query-level entity-linking for NamedEntities. This experiment compares the ability of different entity-annotators in finding NamedEntities mentioned by queries of the ERD-dev, ERD-online and GERDAQ-test datasets. Some of the entity-annotators we experiment with are designed to detect GenericEntities and, in some cases, their mentions; however, in this experiment we evaluate their ability to spot *NamedEntities only*. In other words, the evaluation ignores AbstractEntities and CategoriesOfEntities, and ignores mentions. To define which entities are NamedEntities, we use the list of NamedEntities provided by the organizers of the ERD Challenge.

Table 10 reports the results for this experiment on GERDAQ-test. We refer to the versions of SMAPH piggybacking on either Bing or Google as, e.g., *SMAPH-3 (Bing)* and *SMAPH-3 (Google)*. Precision, recall and F_1 measures are reported both micro- and macro-averaged. For each number, we report the bootstrap estimate of its standard deviation[17]. In this and the next tables, we report this value after the ± sign. We also mark with green, yellow and red the first, second and third best result[18]. We also tested the statistical significance of the improvements between entity-annotators in terms of macro-F_1. Let us indicate with $X \leq_t Y$ the fact that entity-annotator X is significantly worse than Y, according to t-test with $p < 0.05$. We verify the following:

$$\begin{array}{l}\text{AIDA} \leq_t \text{TagME} \leq_t \text{WAT}\\ \text{WAT} \leq_t \text{SMAPH-1 (Bing)} \leq_t \text{SMAPH-S (Bing)} \leq_t \text{SMAPH-3 (Bing)}\\ \text{WAT} \leq_t \text{SMAPH-1 (Google)} \leq_t \text{SMAPH-S (Google)} \leq_t \text{SMAPH-3 (Google)}\end{array}$$

As a first result, these figures show that entity-annotators designed for natural language perform poorly on queries. The macro-F_1 obtained by WAT and AIDA on this dataset is 16% lower than the typical macro-F_1 these entity-annotators achieve on natural language documents[19]. Despite its simplicity, SMAPH-1 outperforms WAT, TagME and AIDA by 4.5% macro-F_1 or more. This supports our claim on the importance of designing entity-annotators that specifically address the domain of queries. The idea of link-back (mapping entities to query tokens), even in the simpler version that judges annotations independently (SMAPH-S), improves SMAPH-1 by 3.5% (Bing) and 4.2% (Google) in terms of micro-F_1. In particular, link-back raises the micro-P of SMAPH-1 by 9.4% (Bing) and 8.0% (Google). The key algorithmic idea of SMAPH-3, which is to model the coherence of the entities forming a solution, brings an additional improvement of 5.4% (Bing) and 6.4% (Google) in macro-F_1 with respect to judging annotations independently (SMAPH-S).

Values for recall indicate that AIDA fails in retrieving most entities. The most likely reason is that AIDA expects well-formed natural language input whereas queries are often not well-formed.

In this experiment, many queries have no ground truth entities attached. This explains why micro measures are so much smaller than macro measures, especially for entity-annotators like AIDA that return very few entities.

We perform the same experiment on dataset ERD-offline. Results are reported in Table 11. Running the experiment with dataset ERD-offline, as opposed to GERDAQ-test, leads to identical ranking by macro-F_1 of the entity-annotators. The queries in ERD-offline are easier to annotate than those in GERDAQ-test, in fact, macro-F_1 is 4%–11% higher. This lets us confirm the conclusions drawn while discussing the results for GERDAQ-test. Due to the limited size of the dataset, no significance test could be performed on this dataset.

We also report in Table 12 the results computed by the ERD Challenge evaluation platform. As detailed in [5], using this platform was the only way to test against the ERD-online dataset. Note that the F-measure computed

[17] The bootstrap estimate of the standard deviation is computed on 40 random samples (with replacement) of the dataset instances. The 40 samples have the same size of the dataset.

[18] With respect to the coloring, the aim is to rank the three best algorithms independently of the data they are fed with, so the three versions of SMAPH will be compared with each other only when piggybacking on the same search engine. This leads to two independent rankings, one for the versions of SMAPH piggybacking on Bing, and one for those piggybacking on Google.

[19] For a report on the macro-F_1 achieved by WAT and AIDA on natural language documents, see e.g., [9].

Entity-Annotator	P_{mac}	R_{mac}	$F_{1\text{mac}}$	P_{mic}	R_{mic}	$F_{1\text{mic}}$
AIDA	94.8 ± 2.3	59.6 ± 2.0	58.4 ± 2.5	31.6 ± 3.1	04.8 ± 2.2	08.3 ± 2.2
TagME	77.6 ± 2.1	86.9 ± 2.2	71.6 ± 2.5	54.0 ± 3.4	70.7 ± 3.7	61.3 ± 3.0
WAT	71.3 ± 2.3	86.4 ± 2.3	65.8 ± 2.7	45.3 ± 3.1	69.9 ± 4.1	55.0 ± 3.2
SMAPH-1 (Bing)	79.2 ± 2.2	90.0 ± 1.8	**76.1** ± 2.5	59.1 ± 3.2	78.9 ± 3.5	**67.6** ± 3.0
SMAPH-1 (Google)	79.4 ± 2.5	89.5 ± 1.9	**75.7** ± 2.8	56.9 ± 3.3	77.2 ± 3.8	**65.5** ± 3.1
SMAPH-S (Bing)	88.6 ± 1.3	86.9 ± 2.1	**79.6** ± 2.0	72.9 ± 2.7	69.9 ± 3.4	**71.4** ± 2.6
SMAPH-S (Google)	87.4 ± 2.3	87.1 ± 1.8	**79.9** ± 2.7	70.4 ± 3.7	71.5 ± 2.9	**71.0** ± 2.8
SMAPH-3 (Bing)	88.2 ± 1.9	89.1 ± 1.7	**81.5** ± 2.2	74.0 ± 3.3	74.0 ± 2.5	**74.0** ± 2.5
SMAPH-3 (Google)	89.0 ± 2.3	88.9 ± 2.0	**82.3** ± 2.7	75.0 ± 4.2	75.6 ± 3.8	**75.3** ± 3.4

Table 10. Performance of query-level entity-linking for NamedEntities on GERDAQ-test.

Entity-Annotator	P_{mac}	R_{mac}	$F_{1\text{mac}}$	P_{mic}	R_{mic}	$F_{1\text{mic}}$
AIDA	92.1 ± 3.3	61.1 ± 2.7	60.4 ± 4.4	42.8 ± 6.8	06.0 ± 5.0	10.5 ± 4.8
TagME	83.5 ± 4.6	92.3 ± 2.6	82.1 ± 4.3	67.2 ± 7.4	84.8 ± 5.3	75.0 ± 6.2
WAT	83.5 ± 4.5	87.4 ± 3.8	76.9 ± 4.8	67.3 ± 7.3	71.7 ± 7.3	69.5 ± 6.4
SMAPH-1 (Bing)	87.4 ± 3.2	93.4 ± 2.3	**84.2** ± 3.6	70.2 ± 7.1	87.0 ± 5.1	**77.7** ± 5.8
SMAPH-1 (Google)	88.4 ± 3.5	92.6 ± 2.4	**85.0** ± 3.9	75.0 ± 6.6	91.3 ± 5.2	**82.4** ± 5.0
SMAPH-S (Bing)	86.8 ± 3.6	92.3 ± 3.0	**85.0** ± 3.9	73.6 ± 6.6	84.8 ± 6.0	**78.8** ± 6.1
SMAPH-S (Google)	89.0 ± 3.4	92.3 ± 2.7	**86.0** ± 3.5	79.6 ± 6.4	84.8 ± 5.4	**82.1** ± 5.6
SMAPH-3 (Bing)	93.4 ± 2.5	93.4 ± 2.4	**90.1** ± 3.0	87.0 ± 5.7	87.0 ± 5.0	**87.0** ± 5.0
SMAPH-3 (Google)	89.6 ± 3.4	93.4 ± 2.7	**86.4** ± 4.0	80.0 ± 6.6	87.0 ± 5.5	**83.3** ± 5.7

Table 11. Performance of query-level entity-linking for NamedEntities on ERD-dev.

Entity-Annotator	F-measure
AIDA	22.1
WAT	58.6
Seznam	66.9
SMAPH-S (Bing)	67.0
NTU	68.0
SMAPH-1 (Bing)	**68.8**
NTNU-UiS	**69.9**
SMAPH-2 (Bing)	**70.8**

Table 12. Performance of query-level entity-linking on ERD-online.

by this platform slightly differs from the macro-F_1 used elsewhere in this article[20]. Unfortunately, the platform is not available anymore, and the dataset was not distributed offline due to copyright restraints[21]. For this reason, we cannot test SMAPH-3 on the ERD-online dataset. Nonetheless, we report for completeness the results of the top-scoring entity-annotators, including SMAPH-1, SMAPH-S and SMAPH-2, an intermediate version covered

[20] In fact, the F-measure considers a solution as correct only if all its annotations are correct, see [5] for details.

[21] Private communication with ERD Challenge organizers.

Entity-Annotator	P_{mac}	R_{mac}	$F_{1\text{mac}}$	P_{mic}	R_{mic}	$F_{1\text{mic}}$
AIDA	94.0 ± 2.2	12.2 ± 2.5	12.6 ± 2.2	28.6 ± 1.8	01.5 ± 1.1	02.8 ± 1.0
TagME	59.6 ± 2.1	49.7 ± 2.8	43.5 ± 2.1	51.8 ± 2.2	48.2 ± 2.4	49.9 ± 2.0
WAT	49.6 ± 2.1	57.0 ± 3.0	46.0 ± 2.3	43.0 ± 1.9	56.4 ± 2.6	48.8 ± 2.0
Tan et al. (from [52])	71.5	58.5	56.9	N/A	N/A	N/A
SMAPH-1 (Bing)	63.4 ± 2.3	68.7 ± 2.5	**57.8** ± 2.3	55.6 ± 2.2	66.5 ± 2.3	**60.6** ± 1.9
SMAPH-1 (Google)	64.1 ± 2.3	66.7 ± 2.7	**57.0** ± 2.1	56.6 ± 1.9	64.8 ± 2.5	**60.4** ± 1.9
SMAPH-S (Bing)	72.8 ± 2.2	56.3 ± 2.5	**55.1** ± 2.3	66.2 ± 2.4	54.0 ± 2.1	**59.5** ± 1.9
SMAPH-S (Google)	71.7 ± 2.4	60.9 ± 2.7	**57.3** ± 2.6	65.7 ± 2.3	59.4 ± 2.4	**62.4** ± 2.1
SMAPH-3 (Bing)	70.9 ± 2.5	62.0 ± 2.4	**58.7** ± 2.4	64.2 ± 2.6	60.4 ± 2.5	**62.2** ± 2.3
SMAPH-3 (Google)	73.2 ± 2.3	65.1 ± 2.4	**62.3** ± 2.2	68.6 ± 2.4	63.6 ± 2.3	**66.0** ± 2.0

Table 13. Performance of query-level entity-linking for GenericEntities on GERDAQ-test.

by Cornolti et al. [10] that performs worse than SMAPH-3 on the GERDAQ dataset. The table shows that WAT outperforms AIDA, but its F-measure is 10.3% lower than SMAPH-1. In turn, SMAPH-1 is outperformed by SMAPH-2 by an additional 2% in F-measure.

9.5.2 Experiment #2: Query-level entity-linking for GenericEntities. This experiment is similar to Experiment #1 but does not have any restriction on the kind of entities taken into account for the evaluation: we now consider any GenericEntity included in Wikipedia, i.e., in addition to NamedEntities, we also consider AbstractEntities and CategoriesOfEntities. Since the ERD-online and ERD-dev datasets only cover NamedEntities, we can perform this experiment only on GERDAQ-test.

Results are shown in Table 13. In comparison to experiment #1, we notice that the ranking of the entity-annotators by macro-F_1 is similar. Similarly to experiment #1, significance tests (t-test with $p < 0.05$) confirm the ranking of the entity-annotators by macro-F_1. The numbers also show that finding GenericEntities is harder than finding NamedEntities: F_1 decreases by about 6%–10% (micro-F_1) and 19% – 23% (macro-F_1) with respect to experiment #1. This decrease might be attributable to the fact that NamedEntities are easier to detect because their forms have a lower degree of ambiguity than other types of entities. Moreover, the number of queries with empty ground truth is lower than in experiment #1 (since there are strictly more GenericEntities than NamedEntities).

Again, entity-annotators designed for long text, such as AIDA and WAT, perform worse than entity-annotators designed for queries. SMAPH-S improves SMAPH-1 only when piggybacking on Google. SMAPH-3 is again the best entity-annotator. Macro-F_1 of SMAPH-3 is 12.7% (Bing) and 16.3% (Google) higher, compared to WAT. Macro-F_1 of SMAPH-3 is greater by 0.9% (Bing) and 5.3% (Google) compared to SMAPH-1.

9.5.3 Experiment #3: Token-level entity-linking for GenericEntities. This experiment compares the ability of entity-annotators in finding not only the GenericEntities mentioned by queries, but their mentions too. Similarly to Experiment #2, we can run this experiment only on GERDAQ-test, the only dataset that provides GenericEntities as ground truth. SMAPH-1, which solves query-level entity-linking, cannot be tested in this experiment. Results for this experiment are reported in Table 14.

Similarly to the other two experiments, significance tests (t-test with $p < 0.05$) confirm the ranking of the entity-annotators by macro-F_1. We first notice that token-level entity-linking performs worse than query-level entity-linking (Table 13). This can be explained by the fact that a solution for token-level entity-linking is intrinsically harder to find than a solution for query-level entity-linking: not only the entity has to be correct, but the mention too. For example, the best macro-F_1 on query-level entity-linking, i.e., 63.3% of SMAPH-3 (Google),

Entity-Annotator	P_{mac}	R_{mac}	$F_{1\text{mac}}$	P_{mic}	R_{mic}	$F_{1\text{mic}}$
AIDA	94.0 ± 2.2	12.2 ± 2.5	12.6 ± 2.2	28.6 ± 1.8	01.5 ± 1.1	02.8 ± 1.0
TagME	57.2 ± 2.0	48.0 ± 2.8	41.5 ± 2.2	49.0 ± 2.2	46.2 ± 2.4	**47.5** ± 2.1
WAT	47.8 ± 2.2	54.9 ± 2.9	**44.5** ± 2.3	41.5 ± 1.9	54.5 ± 2.5	47.1 ± 2.0
SMAPH-S (Bing)	69.0 ± 2.3	52.6 ± 2.6	**51.6** ± 2.3	61.2 ± 2.5	50.6 ± 2.1	**55.4** ± 2.1
SMAPH-S (Google)	68.5 ± 2.5	57.7 ± 2.9	**54.3** ± 2.7	61.4 ± 2.6	56.0 ± 2.6	**58.6** ± 2.4
SMAPH-3 (Bing)	67.8 ± 2.6	58.6 ± 2.5	**55.6** ± 2.5	59.9 ± 2.7	56.7 ± 2.7	**58.3** ± 2.6
SMAPH-3 (Google)	69.9 ± 2.4	62.0 ± 2.6	**59.3** ± 2.3	64.5 ± 2.5	60.4 ± 2.6	**62.4** ± 2.3

Table 14. Performance of token-level entity-linking for GenericEntities on GERDAQ-test.

decreases on token-level entity-linking to 59.3%, indicating that SMAPH-3 finds a number of annotations having correct entity but a mention different than that provided by the ground truth.

Again, off-the-shelf natural language entity-annotators (AIDA and WAT) are worse than entity-annotators built specifically for queries, and SMAPH-S improves over them by 7.1%–12.8% in macro-F_1. SMAPH-3 is still the best entity-annotator with an improvement over SMAPH-S of 4.0% (Bing) and of 5.0% (Google) in macro-F_1.

After this comparison we conclude: (i) SMAPH-3 is the state of the art for entity-linking on queries; (ii) the idea of generating candidate entities by piggybacking on search engines is promising; and (iii) joint annotation models the task of query annotation better than individual entity/annotation judgment, and this lets us build solutions with higher precision and recall.

9.5.4 Experiment #4: Performance on tail queries. This experiment is specific to SMAPH, and aims at assessing its robustness when annotating tail queries. We define a tail query as a query that carries a more specific information need, and for which search engines provide fewer results. For example, a query about a small village in Kazakhstan (`economy of karsakbay`) is a tail query, whereas a query about an event widely covered by the media (`2017 superbowl`) is not a tail query.

The experiment consists of measuring the performance of SMAPH on queries at various positions in the tail. Results are reported for the best-performing entity-annotator: SMAPH-3 piggybacking on Google. We divide the queries of GERDAQ-test in four buckets of equal size (around 62 queries per bucket) according to the number of web results returned by Google for those queries. The first result tells us that the number of web results follows a power law distribution: the quantiles computed at probabilities $p = (0, 0.25, 0.5, 0.75, 1)$ are found respectively at 0, 5k, 68k, 1.3M, and 517M number of results. There are five queries with no results (for them, SMAPH will return an empty solution, though two of them mention an entity), and the query with the highest number of results (517M) is `home book`.

The second result regards the robustness of SMAPH on tail queries. Table 15 reports, for each bucket, the macro-F_1 reached by SMAPH-3 (Google) for query-level entity-linking. Keeping in mind that the macro-F_1 computed on the whole dataset is 62.3% (see Table 13), we note that its value varies across buckets in the range 59.5%–68.3%. The queries for which the search engine finds a lower number of results are those on which SMAPH reaches a lower F_1, which can be explained by the lack of information provided by the search engine. The result is also slightly lower than the average for non-tail queries. This may be due to the excess of information that "dilutes" web results, interleaving those that provide information that is useful to disambiguate the query with those that don't. We conclude that the performance of SMAPH depends on the quality of results returned by the search engine, that it effectively exploits the information that search engines provide, and that it is rather robust on both tail and non-tail queries.

Web results	P_{mac}	R_{mac}	$F_{1\text{mac}}$
0 - 5k	78.5	62.9	59.5
5k - 68k	67.9	67.3	62.8
69k - 1,370k	74.2	68.3	66.7
1,630k - 517,000k	72.0	61.7	60.1

Table 15. SMAPH-3 (Google) performance of query-level entity-linking on GERDAQ-test, bucketed by number of web results.

Query	**Missing entity**
`marital arts for toddlers georgia`	*Toddler*
`minnesota spring lake ice fishing report`	*Spring_Lake,_Minnesota*
`paying taxes on the telephone`	*Payment*
`used instrument`	*Used_Good*
`alcove elementary school`	*Primary_school*

Table 16. Error analysis: examples of entities missing from the candidate set.

9.6 Detailed error analysis and possible improvements

We manually analyzed the results of SMAPH-3 (Google), our best performing entity-annotator, with the aim of learning what types of errors it makes and how they can be avoided. Errors can either be false positives or false negatives, which respectively hurt precision and recall. We found three main reasons for these errors.

(1) The candidate set $\mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3$ does not include an entity mentioned in the query. Since SMAPH only considers entities in the candidate set, this case will generate a false negative. Examples are listed in Table 16. As the examples show, most missing entities are not central to the meaning of the query, in fact they are in most cases modifiers of the central entity (looking at the examples, "for *Toddlers*" is a modifier of *Martial_Art*; *Used_Good* is a modifier of *Musical_Instrument*). The reason why these non-central entities do not appear in the search results is that central entities are predominant among both Wikipedia results (Sources 1 and 2) and the snippets (Source 3). The fact that these missing entities are not central does not mean they are of secondary importance to the semantics of the query. These entities may be included in the candidate set by the same methods that natural language entity-annotators [17] use, e.g., by searching the query for substrings that may be references to a Wikipedia article, and adding the entity corresponding to this article to the candidate set.

(2) The candidate set includes the correct entity, but the model discards it. There are two scenarios that can have this outcome: (a) the learned model does not recognize the link between entity and mention to be strong enough; it decides to be conservative and not link the mention to any entity, thus generating one false negative, or, even worse (b) the model decides that a (wrong) candidate entity which is competing for the same mention is stronger than the correct entity, and adds it to the solution, thus generating one false positive and one false negative. Table 17 shows such examples. We can see from the examples that this type of error occurs mostly when the mention is distant from the most common way of referencing the correct entity, either because of spelling errors (e.g., `healht` vs. `health`) or because of synonymy (e.g., `brigids day` vs. `imbolc`). These cases can be solved by developing features that better model the strength of the mention-entity link, taking into account spelling errors and synonyms.

(3) A sequence of tokens does not refer any entity, yet the model is not conservative enough and assigns it to the most likely entity, thus generating one false positive. Table 18 shows such examples. In these

Case	Query	Good candidate	Chosen candidate(s)
(a)	nail **healht**	*Health*	<none>
(a)	**el paso**isdjobline	*El_Paso,_Texas*	<none>
(a)	**brigids day**	*Imbolc*	<none>
(b)	modern **dishes of pakistan**	*Pakistani_Cusine*	*Pakistan*
(b)	monetry from **first century**	*1st_century*	*Christianity_in_the_1st_century*
(b)	**hotel pontchartrain** detroit	*Pontchartrain_Hotel*	*Crowne_Plaza_Detroit_Convention_Center*

Table 17. Error analysis: examples of entities that are part of the candidate set but are not included in SMAPH-3's result, either because it is too conservative and leaves the mention unlinked (case a) or because it chooses a bad candidate (case b). Correct mention is in bold.

Query	Chosen candidate (wrong)
beta mennan	*Beta_(film)*
moorefeild high school **yellow jackets** football	*Yellowjackets_(Jazz_band)*
baseball hat **hooks**	*Hook_(music)*

Table 18. Error analysis: examples of sequences of tokens that do not reference any entity, and yet they are assigned one. The sequence of tokens is in bold.

cases, there is typically a strong bond between the sequence of tokens and the entity (e.g., beta towards *Beta_film*), but the entity does not contribute to form a coherent interpretation of the query. A better modeling of interpretation coherence would solve these types of errors.

9.7 On the SMAPH-3 models

In this section, we analyze SMAPH-3, the best performing algorithm, in more detail by addressing the following questions: What is the outcome of the training phase (see Section 8.1)? How does the choice of the underlying search engine (Bing or Google) affect the training phase? What is the contribution of each iteration of SMAPH-3 to the solution?

9.7.1 Model description. The outcome of Algorithm 3 (SMAPH-3 training) is a list of regressors $\mathcal{R}_i$ and associated thresholds t_i, one for each iteration of the algorithm. The choice of whether to produce a pair $\langle \mathcal{R}_i, t_i \rangle$ at iteration i, as opposed to terminating and returning the list of regressors obtained at the previous iterations $0, \ldots, i-1$, is determined by three factors: (i) the annotations chosen by the regressors in the previous iterations, which determines what subset of queries will be processed at iteration i and their current solutions P_q, (ii) the availability of correct candidates in C_q, and (iii) the quality of features, that determine whether or not a regressor can be built such that it improves the current solution.

We executed the training pipeline by piggybacking on either Bing or Google. The two runs produce two lists of regressor-threshold pairs. Here we describe the main differences between them. The number of iterations (i.e. the number of regressors) is four for Bing and five for Google. This means that in no case SMAPH-3 will produce a solution larger than four (five) annotations when piggybacking on Bing (Google). The most important difference is that training over Google terminated because no improvement was possible (i.e. all candidate annotations were false positives, see first stop condition in Algorithm 3), while training over Bing terminated because the 5th regressor did not improve the solution obtained at the previous iteration, and hence was discarded (second

	SMAPH-3 (Bing)								**SMAPH-3 (Google)**									
	J-	**J+**	TP	FP	FN	P_{mac}	R_{mac}	$F_{1\text{mac}}$	**J-**	**J+**	TP	FP	FN	P_{mac}	R_{mac}	$F_{1\text{mac}}$	S	S_{TP}
Start			0	0	409	100	10.6	10.6			0	0	409	100	10.6	10.6	100	100
Iteration 0	41	205	141	64	268	74.0	42.9	47.3	37	209	147	62	262	74.8	44.9	49.5	75.6	85.8
Iteration 1	71	134	216	123	193	69.5	56.2	55.2	83	126	228	107	181	71.5	59.3	58.6	75.8	88.5
Iteration 2	86	48	232	155	177	67.8	58.6	55.6	87	39	245	129	164	70.3	61.8	59.4	74.8	88.7
Iteration 3	48	0	232	155	177	67.8	58.6	55.6	32	7	247	134	162	70.0	62.0	59.4	74.6	88.6
Iteration 4	*(not executed)*								5	2	247	136	162	69.9	62.0	59.3	74.5	88.6

Table 19. Contribution of each iteration of SMAPH-3. Performance of token-level entity-linking on GERDAQ-test.

stop condition in Algorithm 3). In other words, signals provided by Google lets the training process produce regressors that explore the whole set of correct candidates, while signals provided by Bing don't.

9.7.2 Contribution of each iteration. To have a deeper understanding of the functioning of SMAPH-3, we dissect its behavior by looking at the solutions achieved at each iteration. Table 19 reports token-level entity-linking results achieved at each iteration for the 246 GERDAQ-test queries. At the beginning of the algorithm, all solutions are empty. This achieves a macro-F_1 of 10.6% (which indicates that 10.6% of queries have an empty ground truth). At each iteration, part of the queries will be assigned a new annotation (column **J+**) while for the remainder (**J-**), their current solution will be returned as final result.

We can see that the distribution of solution sizes between Bing and Google is similar: the empty solution is returned for around 40 queries, a solution of one or two annotations is returned for around 80 queries, a solution of three or more annotations is returned for around 40 queries. Each iteration increases recall by 32.3%, 13.3%, 2.4%, 0% (piggybacking on Bing) and 34.3%, 14.4%, 2.5%, 0.2%, 0% (piggybacking on Google) at the cost of sacrificing a bit of precision by up to 4.5% (Bing) and up to 3.3% (Google). As a consequence of this, macro-F_1 increases at each iteration (with the exception of the last iteration of the Google version, where it slightly decreases).

9.7.3 Difference between piggybacking on Google or Bing. Column S reports the similarity between the solutions found by the two algorithms at each iteration. As suggested in [9], we compute the average similarity at iteration i as $S_i = \sum_{q \in D_t} J(P_{B,q,i}, P_{G,q,i})/|D_t|$, where D_t is the GERDAQ-test dataset, J is the Jaccard measure, $P_{B,q,i}$ and $P_{G,q,i}$ are the solutions obtained for query q at iteration i, piggybacking respectively on Bing and Google. We can see that from the first iteration on, the solution similarity is around 75%, meaning that the models built on top of Google and Bing return quite different annotations. Do the two entity-annotators differ in the correct annotations, in the wrong ones, or both? The last column S_{TP} indicates the similarity of the true positives contained in the solutions, which is 88.6% at the last iteration. This indicates that the two entity-annotators differ both in the true and false positives. The fact that the macro-F_1 measure is similar among the two entity-annotators, though the solutions they provide are different, indicates that SMAPH-3, and in particular its training process, is general enough to be employed with both search engines, and its performance is robust.

9.8 Runtime evaluation

One context where entity-linking on queries may be deployed is web search, which is performed by search engines in tenths of a second. To make it feasible to have a step of entity-linking as part of the web search process, entity-linking must be performed in a comparable time span. To shed some light on the feasibility of this, we measure the runtime of SMAPH. It is important to note that our main focus during the development of SMAPH was the quality of the query annotations rather than the runtime performance, and further work may lead to

Entity-Annotator	**Time** *avg* ± *stdev* **(msec)**	
	Bing	Google
SMAPH-1	90.5 ± 38.4	82.8 ± 34.1
SMAPH-S	344.6 ± 314.4	179.2 ± 249.3
SMAPH-3	316.6 ± 561.3	226.5 ± 491.5

Table 20. Time employed by SMAPH versions to annotate a query. Time is additional with respect to querying the search engine.

important gains in runtime, keeping a high quality. Hence the figures presented in this experiment must be taken as preliminary.

The reader may refer to Figure 2 for an overview of the steps performed by the three versions of SMAPH. The runtime measure does not consider the time needed to fetch the search results (the two top boxes in the figure), as this time is not under our control and varies considerably. In other words, we measure the additional time needed by the versions of SMAPH to annotate a query, after the query has been processed by the search engine.

The experiments run on a consumer PC (Intel i7 CPU), deploying the process on a single core. The runtime has been measured several times showing differences below 1% among runs.

The steps performed by the three versions of SMAPH are somehow incremental: SMAPH-1 (Section 5) generates candidate entities and judges each of them through an SVM prediction, hence the runtime is proportional to $|\mathcal{E}_q|$; SMAPH-S (Section 7) generates candidate annotations by linking candidate entities to all segments of the query, and must express a judgment for each annotation, hence the runtime is proportional to $|\mathcal{E}_q| \cdot |Seg(q)|$; finally, SMAPH-3 (Section 8) decides, at each iteration, which of the candidate annotations have to be added to the solution, leading to a complexity of $n \cdot |\mathcal{E}_q| \cdot |Seg(q)|$, where n is the number of iterations.

Results show that SMAPH-1 has an average annotation time of 82–90 milliseconds. Considering mention-entity pairs (SMAPH-S) increases the runtime by a factor of 2-3, while processing the query iteratively (SMAPH-3) has a runtime similar to SMAPH-S (this is due to the fact that fewer and fewer queries reach the later iterations of SMAPH-3).

In conclusion, SMAPH-1 reaches quite good macro-F_1 performance for query-level entity-linking on GenericEntitites (57.8% with Bing, 57.0% with Google, see Table 13) and is fast enough to be deployable without increasing search time by a significant margin; while SMAPH-3 (59.7% with Bing, 62.3% with Google) may need further optimization to be deployed in on-the-fly query annotation.

10 CONCLUSION AND FUTURE WORK

In this paper we presented and evaluated extensively SMAPH, a family of entity-annotators that perform the task of entity-linking on queries using the information coming from a web search engine, an approach we called "piggybacking". We employ search engines to alleviate the noise and irregularities that characterize the language of queries. Snippets returned as search results also provide a context for the query that makes it easier to disambiguate its terms. From the search results, SMAPH builds a *set of candidate entities* with high coverage. This set is filtered by *linking back* the candidate entities to the terms occurring in the input query, ensuring high precision. A greedy disambiguation algorithm performs this filtering; it maximizes the *coherence* of the solution by iteratively discovering the entities mentioned in the query. We proposed three versions of SMAPH and presented an extensive set of experiments that evaluate them on the GERDAQ dataset, a novel dataset that we have built specifically for this problem via crowd-sourcing, and, when possible, on the dataset of the ERD Challenge. These experiments show that, on GERDAQ-test, SMAPH-3 achieves macro-F_1 scores of 62.3% for GenericEntities and of 82.3% for NamedEntities, while on ERD-dev, SMAPH-3 achieves 90.1% for NamedEntities. On ERD-online, the benchmark dataset of the ERD Challenge, we improved the F-measure by 2.0% with respect to SMAPH-1. This was the best result at the time of the contest and is also the best result among all other entity-annotators that have been tested on the ERD-platform since the end of the challenge (August 2014).

Overall we can conclude that SMAPH-3 is the state-of-the-art for entity-linking on the domain of queries. Though runtime was not our primary concern, and some algorithm and software engineering may improve it, SMAPH-3 is quite efficient, and annotates a query in about one third of a second. To encourage further experiments and uses of SMAPH, we have released its source code on Github[22] and published the SMAPH algorithms as a web service[23]. SMAPH and GERDAQ are released under free licenses.

The idea of piggybacking is to carry out language understanding tasks using search results, thus leveraging the ability of a search engine to understand a query, retrieve relevant results and meaningful snippets. It could be argued that the commercial search engines (such as Google and Bing) already do some sort of entity-linking on queries. Unfortunately, the details of the functioning of these search engines are not published. This paper, on the other hand, has publicly described an efficient method to do entity-linking in queries. An interesting future work is to study how entity-linking by means of piggybacking may help yielding a better query understanding than the underlying search engine already has. We suggest to address this question on open-source search engines, such as ElasticSearch, Sphinx or Lemur/Indri, to check whether SMAPH allows to improve both their ranking and snippets retrieval. A possible experiment in this direction would be to build a search engine that, given a query, executes the following steps: (i) feed the query to a secondary open-source search engine; (ii) annotate the query with SMAPH, using the search results found in the previous step; (iii) use the entities found by SMAPH to improve the search results found in the first step (either improve ranking or snippets quality). The success of this experiment would further validate the piggyback approach.

REFERENCES

[1] Areej Alasiry, Mark Levene, and Alexandra Poulovassilis. 2012. Detecting Candidate Named Entities in Search Queries. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*. ACM, New York, NY, USA, 1049–1050. DOI: http://dx.doi.org/10.1145/2348283.2348463

[2] Michael Bendersky, W. Bruce Croft, and David A. Smith. 2011. Joint Annotation of Search Queries. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 102–111. http://dl.acm.org/citation.cfm?id=2002472.2002486

[3] Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and space-efficient entity linking for queries. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 179–188.

[4] Christopher J. C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report. Microsoft Research. http://research.microsoft.com/en-us/um/people/cburges/tech_reports/MSR-TR-2010-82.pdf

[5] David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. 2014. ERD'14: entity recognition and disambiguation challenge. In *ACM SIGIR Forum (48)*. ACM, 63–77.

[6] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 3 (2011), 27.

[7] Yen-Pin Chiu, Yong-Siang Shih, Yang-Yin Lee, Chih-Chieh Shao, Ming-Lun Cai, Sheng-Lun Wei, and Hsin-Hsi Chen. 2014. NTUNLP Approaches to Recognizing and Disambiguating Entities in Long and Short Text at the ERD Challenge 2014. In *Proceedings of the First International Workshop on Entity Recognition &#38; Disambiguation (ERD '14)*. ACM, New York, NY, USA, 3–12. DOI: http://dx.doi.org/10.1145/2633211.2634363

[8] Marco Cornolti. 2016. *Entity Linking on Text and Queries*. Ph.D. Dissertation. Corso di Dottorato in Informatica, University of Pisa.

[9] Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. 2013. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 249–260.

[10] Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Stefan Rüd, and Hinrich Schütze. 2016. A Piggyback System for Joint Entity Mention Detection and Linking in Web Queries. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 567–578.

[22] http://github.com/marcocor/smaph

[23] https://sobigdata.d4science.org/group/smaph

[11] Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Hinrich Schütze, and Stefan Rüd. 2014. The SMAPH system for query entity recognition and disambiguation. In *Proceedings of the first international workshop on Entity recognition & disambiguation*. ACM, 25–30.

[12] Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *In Proc. 2007 Joint Conference on EMNLP and CNLL*. 708–716.

[13] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity Query Feature Expansion Using Knowledge Base Links. In *Proceedings of the 37th International ACM SIGIR Conference on Research &#38; Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 365–374. DOI:http://dx.doi.org/10.1145/2600428.2609628

[14] Junwu Du, Zhimin Zhang, Jun Yan, Yan Cui, and Zheng Chen. 2010. Using search session context for named entity recognition in query. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 765–766.

[15] Alan Eckhardt, Juraj Hreško, Jan Procházka, and Otakar Smr&#383;. 2014. Entity Linking Based on the Co-occurrence Graph and Entity Probability. In *Proceedings of the First International Workshop on Entity Recognition &#38; Disambiguation (ERD '14)*. ACM, New York, NY, USA, 37–44. DOI:http://dx.doi.org/10.1145/2633211.2634349

[16] Andreas Eiselt and Alejandro Figueroa. 2013. A Two-Step Named Entity Recognizer for Open-Domain Search Queries.. In *IJCNLP*. 829–833.

[17] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM '10)*. ACM, New York, NY, USA, 1625–1628. DOI:http://dx.doi.org/10.1145/1871437.1871689

[18] Paolo Ferragina and Ugo Scaiella. 2011. Fast and Accurate Annotation of Short Texts with Wikipedia Pages. *IEEE Software* 29, undefined (2011), 70–75. DOI:http://dx.doi.org/doi.ieeecomputersociety.org/10.1109/MS.2011.122

[19] Evgeniy Gabrilovich and Shaul Markovitch. 2009. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research* 34, 2 (2009), 443.

[20] Octavian-Eugen Ganea, Marina Ganea, Aurelien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic Bag-Of-Hyperlinks Model for Entity Linking. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 927–938.

[21] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named Entity Recognition in Query. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*. ACM, New York, NY, USA, 267–274. DOI:http://dx.doi.org/10.1145/1571941.1571989

[22] Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To Link or Not to Link? A Study on End-to-End Tweet Entity Linking.. In *HLT-NAACL*. 1020–1030.

[23] Matthias Hagen, Martin Potthast, Anna Beyer, and Benno Stein. 2012. Towards optimum query segmentation: in doubt without. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 1015–1024.

[24] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2014. A Greedy Algorithm for Finding Sets of Entity Linking Interpretations in Queries. In *Proceedings of the First International Workshop on Entity Recognition &#38; Disambiguation (ERD '14)*. ACM, New York, NY, USA, 75–78. DOI:http://dx.doi.org/10.1145/2633211.2634356

[25] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2015. Entity Linking in Queries: Tasks and Evaluation. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval (ICTIR '15)*. ACM, New York, NY, USA, 171–180. DOI:http://dx.doi.org/10.1145/2808194.2809473

[26] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2017. Entity linking in queries: Efficiency vs. effectiveness. In *European Conference on Information Retrieval*. Springer, 40–53.

[27] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Conference on Empirical Methods in Natural Language Processing, Edinburgh, Scotland, United Kingdom 2011*. 782–792.

[28] Alpa Jain and Marco Pennacchiotti. 2011. Domain-independent entity extraction from web search query logs. In *Proceedings of the 20th international conference companion on World wide web*. ACM, 63–64.

[29] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*. ACM, 387–396.

[30] Mandar Joshi, Uma Sawant, and Soumen Chakrabarti. 2014. Knowledge Graph and Corpus Driven Segmentation and Answer Inference for Telegraphic Entity-seeking Queries.. In *EMNLP*. 1104–1114.

[31] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of Wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*. ACM, New York, NY, USA, 457–466. DOI:http://dx.doi.org/10.1145/1557019.1557073

[32] Xiao Li. 2010. Understanding the Semantic Structure of Noun Phrase Queries. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1337–1345. http://dl.acm.org/citation.cfm?id=1858681.1858817

[33] Yanen Li, Bo-Jun Paul Hsu, ChengXiang Zhai, and Kuansan Wang. 2011. Unsupervised query segmentation using clickthrough for information retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval.* ACM, 285–294.

[34] Ying Li, Zijian Zheng, and Honghua Kathy Dai. 2005. KDD CUP-2005 report: Facing a great challenge. *ACM SIGKDD Explorations Newsletter* 7, 2 (2005), 91–99.

[35] Mehdi Manshadi and Xiao Li. 2009. Semantic Tagging of Web Search Queries. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2 (ACL '09).* Association for Computational Linguistics, Stroudsburg, PA, USA, 861–869. http://dl.acm.org/citation.cfm?id=1690219.1690267

[36] Edgar Meij, Krisztian Balog, and Daan Odijk. 2014. Entity Linking and Retrieval for Semantic Search. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM '14).* ACM, New York, NY, USA, 683–684. DOI: http://dx.doi.org/10.1145/2556195.2556201

[37] Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. 2012. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining (WSDM '12).* ACM, New York, NY, USA, 563–572. DOI: http://dx.doi.org/10.1145/2124295.2124364

[38] Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM '07).* ACM, New York, NY, USA, 233–242. DOI: http://dx.doi.org/10.1145/1321440.1321475

[39] David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *AAAI 2008.* http://www.aaai.org/Papers/Workshops/2008/WS-08-15/WS08-15-005.pdf

[40] David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management (CIKM '08).* ACM, New York, NY, USA, 509–518. DOI: http://dx.doi.org/10.1145/1458082.1458150

[41] Marius Paşca. 2007. Weakly-supervised Discovery of Named Entities Using Web Search Queries. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management (CIKM '07).* ACM, New York, NY, USA, 683–690. DOI: http://dx.doi.org/10.1145/1321440.1321536

[42] Aasish Pappu, Roi Blanco, Yashar Mehdad, Amanda Stent, and Kapil Thadani. 2017. Lightweight multilingual entity extraction and linking. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining.* ACM, 365–374.

[43] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14).* ACM, New York, NY, USA, 701–710. DOI: http://dx.doi.org/10.1145/2623330.2623732

[44] Francesco Piccinno. 2016. *Algorithms and data structures for big labeled graphs.* Ph.D. Dissertation. Corso di Dottorato in Informatica, University of Pisa.

[45] Francesco Piccinno and Paolo Ferragina. 2014. From TagME to WAT: a new entity annotator. In *Proceedings of the first international workshop on Entity recognition & disambiguation.* ACM, 55–62.

[46] L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *ACL.* http://cogcomp.cs.illinois.edu/papers/RRDA11.pdf

[47] Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. 2003. Query Segmentation for Web Search.. In *WWW (Posters).*

[48] Stefan Rüd, Massimiliano Ciaramita, Jens Müller, and Hinrich Schütze. 2011. Piggyback: Using search engines for robust cross-domain named entity recognition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1.* Association for Computational Linguistics, 965–975.

[49] Avirup Sil and Alexander Yates. 2013. Re-ranking for joint named-entity recognition and linking. In *Proceedings of the 22nd ACM international conference on Conference on information &#38; knowledge management (CIKM '13).* ACM, New York, NY, USA, 2369–2374. DOI: http://dx.doi.org/10.1145/2505515.2505601

[50] Fabian Suchanek and Gerhard Weikum. 2013. Knowledge Harvesting in the Big-data Era. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13).* ACM, New York, NY, USA, 933–938. DOI: http://dx.doi.org/10.1145/2463676.2463724

[51] Bin Tan and Fuchun Peng. 2008. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of the 17th international conference on World Wide Web.* ACM, 347–356.

[52] Chuanqi Tan, Furu Wei, Pengjie Ren, Weifeng Lv, and Ming Zhou. 2017. Entity Linking for Queries by Searching Wikipedia Sentences. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.* 68–77.

[53] Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, and others. 2015. GERBIL: General entity annotator benchmarking framework. In *Proceedings of the 24th International Conference on World Wide Web.* ACM, 1133–1143.

[54] Xing Wei, Fuchun Peng, and Benoit Dumoulin. 2008. Analyzing web text association to disambiguate abbreviation in queries. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 751–752.

[55] Xiaoxin Yin and Sarthak Shah. 2010. Building Taxonomy of Web Search Intents for Name Entity Queries. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 1001–1010. DOI: http://dx.doi.org/10.1145/1772690.1772792