# The SMAPH System for Query Entity Recognition and Disambiguation

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

Massimiliano Ciaramita
Google Research
Zürich, Switzerland
massi@google.com

Stefan Rüd, Hinrich Schütze
Center for Information & Language Processing
University of Munich, Germany
inquiries@cislmu.org

## ABSTRACT

The SMAPH system implements a pipeline of four main steps: (1) Fetching – it fetches the search results returned by a search engine given the query to be annotated; (2) Spotting – search result snippets are parsed to identify candidate mentions for the entities to be annotated. This is done in a novel way by detecting the keywords-in-context by looking at the bold parts of the search snippets; (3) Candidate generation – candidate entities are generated in two ways: from the Wikipedia pages occurring in the search results, and from an existing annotator, using the mentions identified in the spotting step as input; (4) Pruning – a binary SVM classifier is used to decide which entities to keep/discard in order to generate the final annotation set for the query. The SMAPH system ranked third on the development set and first on the final blind test of the 2014 ERD Challenge short text track.

## Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services, Web-based services

## Keywords

Entity linking, query disambiguation, ERD 2014 Challenge

## 1. INTRODUCTION

This paper describes the SMAPH system that participated in the short track of the 2014 Entity Recognition and Disambiguation (ERD) Challenge [2] hosted by SIGIR. The purpose of the short track challenge is to develop and evaluate a working system that performs entity linking on search engine queries.

Entity mention recognition and linking towards large knowledge repositories (aka knowledge bases or knowledge graphs), is quickly emerging as one of the key pre-processing components for natural language understanding of open domain text [15, 23]. Concerning web search, annotating (open domain) queries with entities is an important part of query intent understanding [2, 13, 24]. Due to their idiosyncratic structure, search queries are particularly challenging for modern natural language processing tools. Two key issues are (i) the noisy language, characterized by misspellings, unreliable tokenization, capitalization and word order, and (ii) brevity, as queries typically consists of just a few terms. Issue (i) has the main effect of degrading the coverage of the string-to-entity mapping, which is a primary component of entity annotators and is typically generated from well-edited text such as Wikipedia. Issue (ii) affects the availability of context that can be leveraged to assist the disambiguation of the spotted strings. As a consequence, the "coherence" estimates that are typically implemented in entity annotators on top of a pairwise entity similarity function (see e.g. [6, 9]), and are used to assess a set of candidate assignments, are much less reliable when applied on queries rather than on well-formed documents, such as tweets, blog posts or news.

We propose to deal with these problems by piggybacking on web search engines [22]. The intuition behind the piggyback approach is that search engines can be viewed as the closest available substitute for the world knowledge that is required for solving complex language processing tasks. Specifically, search engines tend to be robust to the issues raised by queries, because they have been designed to deal with them. Search engines typically deal effectively with surface problems like misspellings, tokenization, capitalization and word order; e.g., a query such as "armstrog mon landign" is quickly resolved to "armstrong moon landing". The piggyback approach is also robust with respect to the lack of context because search engines can leverage extremely large indexed document collections (the whole web), link graphs and log analysis. Also, the typical entity annotator might not find much support, in terms of coherence, because of a small entity candidate set, e.g., in the query "In what year did the first moon landing occur?" there is just one entity to be detected; in contrast, a search engine makes full use of all n-grams.

The SMAPH system is conceptually quite simple, it works as follows. Given an input query it uses a web search engine to fetch the corresponding search-result snippets. We use the bolded text highlighted in these snippets as mentions and the Wikipedia URLs in the search results as candidate entities. The candidate entity set is augmented with the output of an existing annotator, WAT (an improved version of TagMe [6, 20]), applied to the set of mentions. A binary clas-


*ERD'14,* July 06 - 11 2014, Gold Coast, QLD, Australia

http://dx.doi.org/10.1145/2633211.2634348.

sifier is eventually used to make a final decision on whether to add an entity candidate to the final set of predictions. Experiments on the online ERD development evaluation system showed that the basic TagMe, a strong baseline having been developed mainly for short texts, achieves about 51% F1 whereas our system obtains about 63% F1. The SMAPH system ranked third on the development set and first on the final blind test.

The reminder of the paper is structured as follows. Section 2 compares our approach to related work. Section 3 presents the SMAPH system in detail. Section 4 summarizes the main outcomes of our experiments. Section 5 presents some conclusions and ideas for future research.

# 2. RELATED WORK

According to [8, 24], over 70% of web search queries contain named entities (NEs). Entity analysis is becoming an important component of web search technology as systems implement more user-oriented strategies with changes geared towards improving the precision, contextualization, and personalization of search results. In particular, NEs are normally utilized for devising the layout and the content of the result page of a search engine. However, we do not simply piggyback on the entities already detected by the search engine. Rather, our approach can be seen as a second-order method, where the search engine is used as *mention* generator and as one of the sources for *entity* candidate generations.

In recent years, the linguistic annotation of search queries has received increasing attention specifically with respect to the problem of finding NEs occurring in queries. The first paper dates back to 2003 [21]. Then the number of publications increased rapidly covering query segmentation, POS tagging, NER tagging with a limited number of classes (e.g. person, organization, location), tagging with linguistic structures, NE recognition (possibly associated to *intent* or a few pre-defined classes), just to cite a few. One of the first papers on the subject of disambiguating query terms is [12], which determines the senses of (single) words in queries by using WordNet. More recently, first [16] and then [1] studied the problem of linking search queries to Wikipedia concepts. The former used training data and performed worse than TagMe [14], the latter avoided training but its experiments were limited to 96 queries. The piggyback approach, applied to the task of named entity recognition on web queries, was introduced in [22].

In our system we will make use of entity annotators that were designed and optimized for short texts – such as tweets, news items and blog posts – that are somewhat longer than queries. So these short-text entity annotators cannot be successfully applied to queries without some modification. These annotators have attracted a lot of interest in the last few years, with several interesting and effective algorithmic approaches to solve the mention-entity match problem, possibly using other knowledge bases such as DBpedia, Freebase or Yago (see e.g. [5, 6, 9, 16, 18, 7, 17, 10]). According to [4], TagME is one of the best performing systems to date, in terms of F1 and efficiency. In the annotation step of the SMAPH pipeline we use a recent variant of TagMe, called WAT [20], which will be applied to a short text suitably constructed to include proper mentions, or variations of them, of the input query (for details see Section 3.2).

# 3. THE SMAPH SYSTEM

## 3.1 Overview

The approach implemented by the SMAPH system consists of two main phases: *candidate entity generation* and *entity pruning*. The goal of the first phase is to generate a broad set of entities that may be linked to the query. Candidate entities are generated from three different sources. In this phase, the focus is on recall rather than precision, as the ideal outcome of this phase is a set containing all correct entities, at the cost of some amount of false positives. The second phase is meant to prune away all wrong entities, refining the set of candidate entities to the final output. The first phase is heuristic, the second phase uses a binary SVM classifier trained on a set of annotated queries.

An overview of the algorithm is shown in Figure 1. Schematically, the candidate generation phase implements the following three components:

**S1 - Annotator.** It consists of four steps:

1. The input query $q$ is issued to a search engine.
2. Snippets returned by the search engine are parsed to identify bold text regions (*bolds* for short), which are generally a modified (often corrected) version of the keywords (or sequences of keywords) of the original query.
3. Bolds are filtered, so to discard those that do not appear in the input query. The filter is based on the minimum edit distance between the bolds and the query n-grams (see Algorithm 1 for details).
4. The string concatenation of the bolds is issued, as a single document, to the annotator for disambiguation. This way, the disambiguation is informed with a richer context and a richer mention corpus than that provided by the query. Ids in the annotations returned by the annotator form the set of entities provided by Source 1.

**S2 - NormalSearch.** The top-k web search results for query $q$ are searched for Wikipedia pages. Results that are Wikipedia pages form the set of entities provided by Source 2.

**S3 - WikiSearch.** The input query $q$ is concatenated with the word `wikipedia` and issued again to the search engine. Similarly to Source 2, the results that are Wikipedia pages form the set of entities provided by Source 3.

The union of the sets of candidates generated by the three sources constitutes the final entity candidate set for the query. Each source also provides a few features for each candidate that are used by the binary classifier to decide whether the entity should be kept or not. In the following sections, we provide details, motivation and examples for these modules. First, we describe the first pass annotator used in source 1.

## 3.2 A text annotator

Since the concatenation of snippets that need to be annotated do not compose a well-formed text, we need to use a tagger that does not rely on a natural language parser. To this end, we decided to use WAT, an improved version of TagMe.

**TagMe** searches the input text for mentions defined by the set of Wikipedia page titles, anchors and redirects. Each

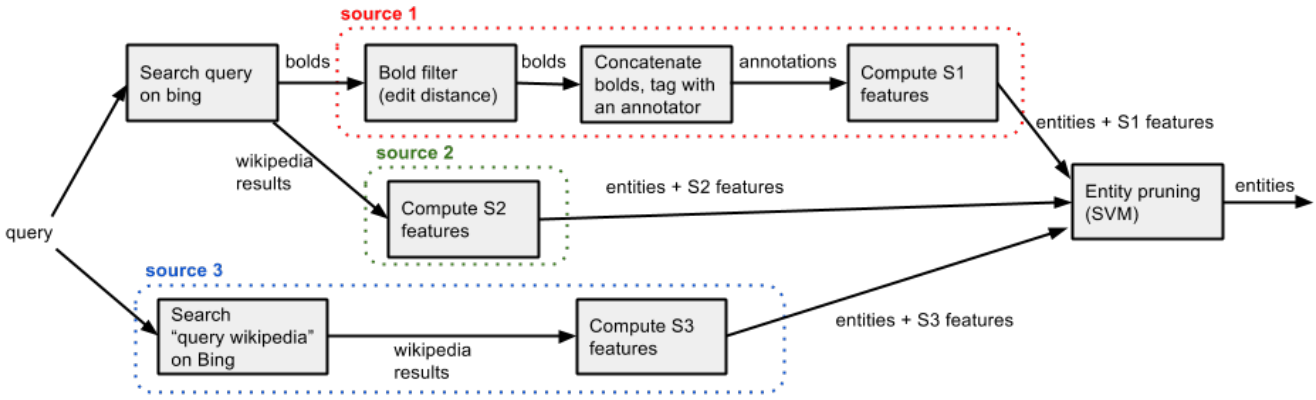


**Figure 1: Overview of the SMAPH query-annotation system.**

mention is associated with a set of candidate entities. Disambiguation exploits the structure of the Wikipedia graph, according to the *relatedness measure* introduced in [19] which takes into account the amount of common incoming links between two pages. TagMe's disambiguation is enriched with a *voting scheme* in which each possible binding between a mention and an entity is scored and then it expresses a vote for all other bindings. The best annotation for each mention is then selected heuristically. TagMe has been mainly designed to deal with short texts that are somewhat longer than queries.[1]

**WAT** follows the main algorithmic structure of TagME with two notable changes: it uses Jaccard similarity between the inlinks of two pages as a measure of their relatedness (instead of [19]) and it uses PageRank (instead of their commonness) to sort the candidate entities that may annotate a mention. This improved variant of TagME was released in June 2014 for efficient and accurate annotation of both long and short texts [20]. In the algorithm of Source 1 presented in this paper, we employ the D2W interface of WAT, namely the interface that takes as input both the text and its mentions and returns the entities associated to those mentions.

## 3.3 Source 1 - Annotator

Here follows a detailed description of the algorithm employed by Source 1.

**Querying the search engine.** The query $q$ to be annotated is issued to the `Bing` search engine, by deploying its public `API` and enabling the *spelling correction* feature. This way the results will not be affected by spelling-errors possibly present in the query.[2]

**Fetching result snippets.** The first 50 snippets returned by `Bing` are analyzed, and all **bold**-highlighted substrings are identified: say $b_0, b_1, \ldots b_z$.[3] Notice that a snippet may contain zero or more bolds, depending on the query. According to the way search engines compose snippets, those substrings offer to the user an effective highlighting of the query terms within the result pages, providing many forms in which a keyword appearing in the query could look like in an actual (generally well-composed) web page. This is a key feature as these bolds are used to form the input of a text annotator, and taking keyword forms that appear in actual documents increases the probability that the form is contained in the annotator's dictionary. Looking at those bold-spots offers three clear advantages with respect to segmenting directly the query terms: (i) an automatic correction of the possible spelling errors, which may occur in terms present in $q$; (ii) an automatic reordering of the query terms, according to their most relevant occurrences on the web, given the assumption that top results in web search engines are the most relevant ones; (iii) we get a plethora of possible segmentations of the permutations of the query terms, according to their most notable occurrences on the web.

**Filtering out poor bold-spots.** The collected bolds are analyzed in order to filter out $b_i$s that do not appear in the query. This is necessary as the search engine does not only return bolds contained in the query, but also other parts of the snippets to assist users eyeballing `Bing`'s results. Specifically, we use a method to find the word-by-word edit distance between the bold and the query. Using our own training dataset (see below), we empirically found the best threshold value to be 0.7.

The algorithm employed to compute the minimum edit distance between a bold $b$ and the query $q$ is Algorithm 1. For each word in the bold, we find the closest word in the query and get the (normalized) Levenshtein distance between them. We return the average of those distances. We call this value $\mathcal{D}(b, q)$. E.g. for $b =$ "neil armstrong junior" and $q =$ "neil amstrng moon" (note the typo!), the edit distance is $(0 + \frac{2}{9} + \frac{4}{6})/3$.

**Annotating bold spots.** Bolds that survived the previous filtering step are concatenated to form an artificial text which is submitted to WAT for annotation. Together with the text, we submit to WAT the list of mentions (one for each bold) that it has to consider. This way we override WAT's spotter and force it to tag the bolds as they appear in the snippets. Describing the internal mechanism employed by WAT to disambiguate the bolds is beyond this paper's scope[4]. In short, for each bold, WAT spots a set of candidate entities (i.e. entities that have been linked in

[1] `http://acube.di.unipi.it/tagme`.
[2] `http://datamarket.azure.com/`
[3] We tried other values for the number of fetched results, but 50 gave the best performance on the development set.
[4] See [20, 6] for a detailed description.

**Algorithm 1** Word-by-word minimum edit distance $\mathcal{E}(b,q)$ between a bold $b$ and a query $q$. Function `repl_non_words_lc(t)` turns $t$ lower-case and replaces all non-word characters of the input text with a space; function `tokenize(t)` splits $t$ into substrings using whitespace as delimiter; $lev(a,b)$ is the Levenshtein distance between $a$ and $b$.

$t_b \leftarrow tokenize(repl\_non\_words\_lc(b))$
$t_q \leftarrow tokenize(repl\_non\_words\_lc(q))$
$d \leftarrow 0$
**for all** $b_i \in t_b$ **do**
  $d \leftarrow d + \min_{q_i \in t_q}\{lev(b_i,q_i)/\max(|b_i|,|q_i|)\}/|t_b|$
**end for**
**return** d

a Wikipedia article using that bold as anchor), and picks, among the candidates, the one that maximizes the coherence with the other candidates. This returns at most one (pertinent) entity $e_i$ per bold $b_i$.

**Features.** Source 1 provides features 1-9 described in Table 1. For each annotation $(b_i, e_i)$ spotted by WAT for bold $b_i$, Source 1 provides the SVM pruner with a set of nine features, some concerning the entity $e_i$, and some concerning the bold $b_i$. About the entity, Source 1 provides the following four features: (i) a score $\rho(b_i, e_i)$, which quantifies the strength of the WAT's annotation $(b_i, e_i)$; (ii) the local coherence $coh_l(e_i)$, that is the average coherence with the entities found by WAT for the other bolds; (iii) the PageRank $pr(e_i)$ of the entity $e_i$ in the Wikipedia graph; (iv) the commonness $comm(b_i, e_i)$, which denotes the frequency with which $b_i$ links to $e_i$ in Wikipedia. Five more features depend on the bold only: (v) the frequency of the bold $b_i$ in the results returned by Bing (i.e. how many results featured $b_i$); (vi) the average rank of the results featuring bold $b_i$; (vii) the link probability $lp(b_i)$, which denotes the probability that bold $b_i$ occurs as an anchor in Wikipedia; (viii) the ambiguity of bold $b_i$, that is the number of distinct entities that anchor $b_i$ links to in Wikipedia; (ix) the minimum edit distance between $b_i$ and the query, as computed in Algorithm 1.

**An example.** The power of this new spotting step is that it could identify bold-spots which do not match exactly substrings of $q$ because of misspellings or query-term permutations. In order to appreciate this novelty let us consider the query $q$ = "Armstrong landing mon", which presents a misspelling in the writing of the term "moon", and a swapping of the terms "moon" and "landing". This type of input would be troublesome for classic topic annotators, such as TagMe or WAT. In fact, they would not find any annotations at all. Using the approach described above we would get bold-spots for "moon", "moon landing" and also "Armstrong", thus simplifying significantly the disambiguation task of a standard annotator.

It goes without saying that the situation is not as easy as the example could lead to argue, because there are cases in which the returned bold-spots are not all useful/pertinent. Take as an example the query "armstrong moon". The API-interface of `Bing` returns, among others, the bold-spot "Armstrong World Industries", a company headquartered in Pennsylvania. This spot should be discarded, and this motivates the latest step of this phase.

| ID | Name | Definition |
|---|---|---|
| 1 | $\rho(b,e)$ | $\rho$-score, see [20] |
| 2 | $coh_l(e)$ | local coherence, see [20] |
| 3 | $pr(e)$ | Wikipedia graph PageRank of $e$ |
| 4 | $comm(b,e)$ | commonness, see [20] |
| 5 | $freq(b)$ | $(\lvert r \in R \ : \ b \in \mathcal{B}(r)\rvert)/\lvert R\rvert$ |
| 6 | $avgRank(b,R)$ | $(\sum_{i=0\rightarrow\lvert R\rvert} r_i)/\lvert R\rvert$ where $r_i = \begin{cases} i/\lvert R\rvert & \text{if } i \in \mathcal{I}(b) \\ 1 & \text{otherwise} \end{cases}$ |
| 7 | $lp(b)$ | link probability, see [20] |
| 8 | $ambig(b)$ | ambiguity, see [20] |
| 9 | $EditDist(b,q)$ | $\mathcal{E}(b,q)$ |
| 10 | $rank(e)$ | $i \ : \ R[i] = \mathcal{U}(e)$ |
| 11 | $webTotal(q)$ | $\mathcal{W}(q)$ |
| 12 | $EDTitle(t,q)$ | $\mathcal{E}(\mathcal{T}(e),q)$, see Algorithm 1 |
| 13 | $EDTitNP(t,q)$ | $\mathcal{E}(\mathcal{T}^*(e),q)$, see Algorithm 1 |
| 14 | $avgEDBolds(e)$ | $\mu\{\mathcal{E}(b,q) \ : \ b \in \mathcal{B}(\mathcal{U}(e))\}$ |

**Table 1: Features used by the SVM pruner. $R$ is the array of (max. 50) URLs returned by Bing; $\mathcal{B}(u)$ is a function that maps an URL $u$ returned by Bing to the list of bolds contained in the snippets associated to $u$; $\mathcal{I}(b)$ is a function that maps a bold $b$ to the list of result indexes whose snippets contain $b$; $\mathcal{E}(b,q)$ is the word-by-word minimum edit distance between $b$ and $q$ as computed by Algorithm 1; $\mathcal{W}(q)$ is the total number of webpages found by `Bing` for query $q$; $\mathcal{U}(e)$ is the Wikipedia URL of entity $e$; $\mathcal{T}(e)$ is the title of entity $e$; $\mathcal{T}^*(e)$ is $\mathcal{T}(e)$ excluding the final parenthetical, if any.**

On the other hand, in the above query "Armstrong landing mon", the spot "moon" is right but the corresponding entity should be discarded because it is not totally pertinent to the intent of the query. The next steps are intended to address the issues of spot pruning, spot disambiguation for entity generation, and entity pruning.

## 3.4 Source 2 - NormalSearch

**Entity candidates.** The second source re-uses the search results from Source 1. The idea is to check whether in the top-10 results occurs a link to a Wikipedia page, and in this case add this page as a candidate entity. Not all URLs starting with `en.wikipedia.org/wiki/` are actual Wikipedia articles, hence we need to filter out service pages. This is done heuristically by keeping only pages that do not start with *Talk:*, *List of:*, *Special:*, and a few other keywords.

**Features.** Source 2 provides features 10-14 described in Table 1: (i) the $rank(e_i)$ of the Wikipedia page in the search results; (ii) the number of web pages returned for that query; (iii) the minimum edit distance between the title of the entity and the query (see Algorithm 1); (iv) the minimum edit distance (see Algorithm 1) between the title (excluding final parentheticals) and the n-grams of the query[5]; (v) the average minimum edit distance (see Algorithm 1) between the bolds and the the query.

[5]The motivation behind this feature is that many entities have titles in the form *Chicago (musical)*, and the edit distance removing *(musical)* is more informative.

### 3.5 Source 3 - WikiSearch

**Entity candidates.** The third source is similar to the second one but the query issued to `Bing` is obtained by concatenating the original query with the term `Wikipedia`. This way, we give a light suggestion to the search engine to spot Wikipedia pages. The search engine will return Wikipedia pages in its top-K results only in case there is a reasonable evidence that the page is linked to the query, otherwise it will return non-wikipedia pages, that get discarded.

This approach is opposed to doing a site-constrained search on Wikipedia by adding `site:en.wikipedia.org` to the input query, thus limiting the scope of the search engine to Wikipedia. We found that this site-constrained search was too rigorous a constraint and returned too many false positives if there were no good Wikipedia pages (but the search engine was still forced to provide them). Instead, with the proposed approach, the search engine is provided with a soft suggestion that we'd like the result to include mainly Wikipedia pages, but only if pertinent.

**Features.** The features extracted for this source are the same as the ones described for Source 2.

### 3.6 Entity Pruning

**A dataset of annotated queries.** To train the final binary classifier for entity pruning, we used a dataset of annotated queries. For the sake of space and focus, we will not get into the details of the creation of the dataset here, which was generated independently of the ERD contest, as a continuation of previous work [22, 4]. In short, we randomly picked a 1000 query subset of the 800,000 queries provided by the KDD 2005 Cup. Then we set up two crowd-sourcing jobs for annotating the queries on the Crowdflower platform. In the first job, we asked contributors to spot meaningful mentions in the query and link them to all possible candidate entities. In the second job, we asked contributors to select only appropriate annotations and discard the others. This process of annotation was not limited to named entities as in the ERD contest and we filtered out annotations not present in the ERD list before training.

**Binary classification.** The last phase is aimed at pruning the entities $e_i$ that are not pertinent for annotating $q$. In order to solve this issue we designed a binary classifier, implemented via LibSVM [3], and used the set of features returned in the previous phase, rescaled to the $[-1, 1]$ range. The classifier was trained over our dataset, keeping as a model selection objective function maximizing F1 over the online development set. The entities positively classified were added to the annotation set for $q$.

**Parameters.** We used a binary SVM classifier with an RBF kernel. We set the values for $\gamma$ and $C$ by means of a simple grid search, the final values were respectively, 0.03 and 5. We set the value of the class imbalance to 3.8 (positive) and 6 (negative). All parameter optimizations were performed with respect to the online development set evaluation.

Finally, the Wikipedia Ids not in the provided list of named entities were rejected and Ids mapped to Freebase Ids[6].

[6]The list of named entities was defined by the ERD Challenge organizers.

### 3.7 Promising failures and known limitations

There are a few modifications to the above algorithm we implemented and evaluated but did not yield any advantage. We briefly describe the main one that seemed most promising beforehand.

Since a significant number of queries (according to our estimates, around 30% of the first 100-query, online NER Challenge evaluation development set) did not contain any entities, we thought of building a preliminary SVM classifier to discriminate empty queries (queries with no associated entities) from non-empty ones, and thus executed the above algorithm only in the latter case, while returning an empty set in the first case. We addressed this problem by designing a statistical classifier implemented via LibSVM [3], and used a set of features extracted from $q$ and from the bolds $b_i$s: number of pages found by `Bing` for the issued query $q$, minimum average rank of the snippets containing each bold $b_i$; maximum frequency among the bolds $b_i$; presence of a result pointing to a Wikipedia page. This classifier has been trained aiming for the maximization of the *true negatives* (hence to detect correctly the empty queries), constraining the *false negatives* rate under 2%. Though the filter worked fine, it turned out it did not give any improvement, as all queries that were marked as empty by this filter would anyway have an empty set of entities using the above algorithm. Following the "keep it simple" maxim, we removed this filter.

There is also one obvious limitation of the current system we are aware of, but we did not have time to fix, even though it has the following simple solution. As each decision on whether to keep an entity or not is independent, we may return multiple entities for the same mention. For example, the SMAPH system can return both *Texas* and *Austin,_Texas*. The simplest fix possible is a filter that only keeps the longest entity; we did not have time to implement this before the deadline.

## 4. EXPERIMENTAL RESULTS

We carried out several experiments to assess the coverage of the three sources (thus, excluding the Entity Pruning stage) on the fully-annotated 91-queries ERD dataset released by the Challenge organizers as development set. Results show that sources 1, 2 and 3 respectively cover 75%, 50%, and 78% of the gold standard. The union of the three sources candidate sets yields an impressive coverage of 98.2% (56 true positives, 252 false positives, 1 false negative). This is evidence that search results complement well the prior knowledge compiled from Wikipedia that the WAT annotator can put to use.

The annotation evaluation was performed via the on-line ERD evaluation system. According to it, the SMAPH system reached an F1 of 62.9% (compared to a baseline of 50.9% reached by Tagme) on the development data, and an F1 score of 68.6% in the final evaluation, obtaining the best result amongst the participants of the ERD short track Challenge. We believe that these convincing results are mainly due to the novel design features of the system. First, the mentions generation method, based on bolded snippets regions, ingeniously exploits the search engine ability to normalize and contextualize queries; this, we believe, provides a significant technical contribution in and of itself. Secondly, TagMe and piggyback nicely complement each other in the candidate generation task providing excellent candidate cov-

erage. Finally, the simplicity of the final classifier which is based on just 14 features proved robust to over-fitting.

## 5. CONCLUSION AND FUTURE WORK

As information technology increasingly focuses on conversational and intelligent methods for human-machine interaction, the natural language understanding problem becomes more and more central. The ERD task has quickly established itself as the first, crucial, step in language processing in mainstream applications. The main reason is probably that it has offered, for the first time, a practical handle on long standing problems such as *synonymy* and *ambiguity*; that is, on the problem of modeling "what is talked about", e.g., on the web. Notwithstanding its success, ERD is by no means a solved problem, especially for less frequent entities. The main intuition behind the SMAPH system has been to investigate how search engines can be put to use in this task, as, we argued, search engines can be viewed as the closest available substitute for the world knowledge that is required for solving complex language processing tasks.

As for future work, in the immediate future we are planning to investigate the following issues. First of all, mention detection, which was not required in the 2014 ERD task but is an important component of the ERD problem. Given that queries are typically short one can envision a broad set of combinatorial algorithmic approaches to search the space of possible segmentations. The second problem we want to address is modeling the selection of the set of entities returned globally, rather than independently, as the SMAPH system currently does. This problem is actually tightly related to the segmentation problem and we envision solutions that will deal with both sub-tasks jointly.

## 6. ACKNOWLEDGMENTS

We thank the ERD Challenge organizers for the opportunity to focus on the challenging task of NE detection and annotation in queries.

The WAT system [20] was developed by Francesco Piccinno (University of Pisa) who kindly provided us access to his code and continuous support in querying it. Our system can be queried through an API that has been implemented by Diego Ceccarelli (ISTI CNR) who kindly shared its code with the other ERD Challenge participants.

This work, at the University of Pisa and the University of Munich, was partially supported by two Google Faculty research awards. The University of Pisa group was also partially supported by the MIUR PRIN grant ARS-Technomedia.

## 7. REFERENCES

[1] C. Boston, H. Fang, S. Carberry, H. Wu, X. Liu. Wikimantic: Toward effective disambiguation and expansion of queries. *Data & Knowledge Engineering*, 90: 22-37, 2014.

[2] D. Carmel, M. Chang, E. Gabrilovich, B. Hsu and K. Wang. ERD 2014: Entity Recognition and Disambiguation Challenge. SIGIR Forum, ACM, 2014.

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

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

[5] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. EMNLP and CNLL*, 708–716, 2007.

[6] P. Ferragina and U. Scaiella. Fast and accurate annotation of short texts with Wikipedia pages. *IEEE Software*, 29(1): 70-75, 2012. Also in *ACM CIKM*, 1625–1628, 2010.

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

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

[9] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *Proc. EMNLP*, 782–792, 2011.

[10] N. Houlsby and M. Ciaramita. A Scalable Gibbs Sampler for Probabilistic Entity Linking. In *Proceedings of ECIR*, 335–346, 2014.

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

[12] S. Liu, C. Yu, W. Meng. Word Sense Disambiguation in Queries. In *CIKM*, 525–532, 2005.

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

[14] E. Meij. A Comparison of five semantic linking algorithms on tweets. Personal Blog: `http://alturl.com/aujuc`, 2012.

[15] E. Meij, K. Balog, D. Odijk. Entity linking and retrieval for semantic search. In Procs ACM WSDM, 683–684, 2014.

[16] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *Proc. WSDM*, 563–572, 2012.

[17] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proc. ACM CIKM*, 233–242, 2007.

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

[19] D. Milne and I. H. Witten. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. *AAAI Workshop on Wikipedia and Artificial Intelligence*, 2008.

[20] F. Piccinno, P. Ferragina. From TagME to WAT: a new entity annotator. In *Entity Annotation and Disambiguation Challange (ERD)*: Long track, ACM SIGIR Forum, 2014.

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

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

[23] F.M. Suchanek, G. Weikum. Knowledge harvesting in the big-data era. In *ACM SIGMOD*, 933-938, 2013.

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