From TagME to WAT: a new Entity Annotator

Francesco Piccinno and Paolo Ferragina
Dipartimento di Informatica
University of Pisa
{piccinno, ferragina}@di.unipi.it

ABSTRACT

In this paper we propose a novel entity annotator for texts which hinges on TAGME's algorithmic technology, currently the best one available [6, 4]. The novelty is twofold: from the one hand, we have engineered the software in order to be modular and more efficient; from the other hand, we have improved the annotation pipeline by re-designing all of its three main modules: spotting, disambiguation and pruning. In particular, the re-design has involved the detailed inspection of the performance of these modules by developing new algorithms which have been in turn tested over all publicly available datasets (i.e. AIDA, IITB, MSN, AQUAINT, and the one of the ERD Challenge).

This extensive experimentation allowed us to derive the best combination which achieved on the ERD development dataset an F1 score of 74.8%, which turned to be 67.2% F1 for the test dataset. This final result was due to an impressive precision equal to 87.6%, but very low recall 54.5%. With respect to classic TAGME on the development dataset the improvement ranged from 1% to 9% on the D2W benchmark, depending on the disambiguation algorithm being used.

As a side result, the final software can be interpreted as a flexible library of several parsing/disambiguation and pruning modules that can be used to build up new and more sophisticated entity annotators. We plan to release our library to the public as an open-source project.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.5 [**Information Storage and Retrieval**]: Online Information Services, Web-based services; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

Keywords

TagMe; Entity Annotation; Wikipedia; Graph-based Algorithms

1. INTRODUCTION

Enriching texts with proper semantic annotations is today a proven technique used to increase the quality of several IR tasks, ranging from indexing to classification and retrieval. The classic BoW paradigm is starting to show its limitations, fact confirmed by the recent paradigm shift in the IR community which is now trying to design more reliable *entity annotators* (see e.g. [11, 16]).

An entity annotator is a tool that applied to a text is capable of identifying short-and-meaningful sequences of terms (also called *mentions*) and annotate them with unambiguous identifiers (also called *entities*) drawn from a catalog (usually Wikipedia). The process involves three main steps: (i) *spotting* of the input text, that is finding mentions and sets of candidate entities they could link to; (ii) *disambiguation* of mentions, in which each mention is linked to the most pertinent entity that best describes it; (iii) *pruning* of a mention, that discards a mention and its linked entity in the case that the annotation is not considered correct or consistent with the overall interpretation of the input text.

In this work we introduce our new entity annotator WAT, the successor of TAGME [6] the best known annotator to date [4], redesigned from the ground up in order to easily accommodate modifications and tweaking of each different step of the annotation pipeline. The paper will provide an analysis of each single step of the entire pipeline, quantifying how the performance of each single module can influence the performance of the successive one, and how this will impact the final performance of the system.

2. RELATED WORKS

The first work that introduced the entity linking problem was WIKIFY [12], followed by [5]. Milne and Witten [13, 14], improved the original idea by proposing a new annotator that hinges on (i) the identification in the input text of non-ambiguous mentions C which links to only one entity; (ii) the introduction of a *relatedness* function $rel(e_1, e_2)$ between two Wikipedia entities e_1 and e_2 based on the size of the overlap between their in-linking pages; (iii) and a notion of coherence of an entity e with other un-ambiguous entities C.

Chakrabarti et al [9] improved further the previous approaches by introducing two scores for each annotation: one *local*, modeling the mention-entity linkage compatibility, and the other *global*, modeling the coherence among all the entities chosen to disambiguate all the mentions. The authors then used a collective annotation approach, via the (slow) solution of a quadratic assignment problem, to find the best mapping that maximizes the sum of the two scores.

Subsequently, Ferragina and Scaiella [6] proposed a fast yet accurate local disambiguation procedure that disambiguates one mention at the time by means of a voting scheme which operates pairwise and links it to its top-scoring candidate entity. This voting scheme aims at finding the collective agreement between all the mention-entity bindings by exploiting the relatedness function introduced by [14] and other statistics derived from a pre-processing of Wikipedia.

A completely different approach is the one followed by the authors of AIDA [8]. The authors proposed to solve the disambiguation problem by exploiting a new form of coherence graph, called *Mention-Entity Graph*, a weighted undirected graph in which nodes are either mentions or candidate entities. They considered weighted edges between mentions and entities, capturing context similarities and weighted edges among entities to capture coherence among entities. They then solved the disambiguation problem by looking for a dense sub-graph that would contain all the mentions nodes and exactly one mention-entity edge for each mention. Since the dense sub-graph problem is NP-hard they adopted a light approximation algorithm [15].

In [7], the authors proposed a similar framework by exploiting the so called *Referent Graph* that strictly resembles the *Mention-Entity Graph* of [8]. The difference being the disambiguation method which here is based on PageRank.

More recently [18] proposed another approach to entity disambiguation based on the HITS algorithm which is run over a sub-graph of the RDF Knowledge Base (DBPedia [1]), derived using a truncated BFS visit. The seeds node used in the BFS exploration are the entities found in the spotting phase. A set of heuristics, such as co-reference resolution and normalization and expansion of surface forms, are used to achieve good performance for the proposed system. The authors claim an increase in accuracy up to 20% with respect to [8] but a comparison over the AIDA CONLL dataset is missing.

In designing WAT we took inspiration from the recent trends in entity annotator design, and set up a *modular framework* which hinges on the TAGME's approach but, in addition, offers the most well known disambiguation algorithms (based on voting scheme or on collective linking using the Mention Entity Graph model) and other software modules that can be flexibly combined to design and test new ideas in the entity annotators/entity linking landscape.

3. TERMINOLOGY

The objective of an entity annotator is to annotate a text t producing a set of annotations $a \in A_t$. Each annotation a is a binding between a *mention* m, occurring in t, and an *entity* e and can be represented by the pair (m, e). In our context an *entity* is a Wikipedia article identified by its page-ID, while a mention is a sequence of terms located in the input text t. Each mention is uniquely identified by a pair (s, l) (mention's span) where s is the position of the first character in t of the mention and l is the length of the mention.

In order to be annotated, an input text has to go through the annotation pipeline that consists of three main steps:

Spotting: the entity annotator will scan the input text t looking for interesting sequences of terms and it will produce a set M_t of possible mentions. Additionally a list of candidate entities E_m will be retrieved for every single mention $m \in M_t$. The candidate list will contain all possible senses that can be associated to a specific mention. We can distinguish between unambiguous mentions, i.e. $|E_m| = 1$, and ambiguous mentions, i.e. $|E_m| > 1$.

Disambiguation: in this phase the entity annotator will associate a score s_e to each candidate entity $e \in E_m$. The score models how well the entity e describes the mention m in the context of the input text t. The top-scoring bindings will be the candidate annotations A_t.

Pruning: in this phase the entity annotator has to choose whether or not to discard an annotation from the set A_t.

4. FRAMEWORK

In this section we will briefly describe the changes and additions implemented by WAT with respect to our reference annotator TAGME.

4.1 Spotter

The possible set of mentions that WAT can recognize in a given input text t derives from an offline pre-processing of Wikipedia. As in [6] we used wiki-anchors, titles, redirect pages to create the database of possible spots. Additionally every single mention in the spot database has a link probability attribute $lp(m) = link(m)/freq(m)$ where $freq(m)$ denotes the number of times the given mention m occurs in Wikipedia (as an anchor or not); whereas $link(m)$ denotes the number of times the mention m occurs as an anchor in Wikipedia's pages. The spot database provides for each mention m a list of candidate entities E_m which are ranked according to the prior-probability attribute (also known as *commonness*) defined as $Pr(e|m)$, which is the number of times the mention m is used to link the entity e.

In addition to the classic attributes, we added a diverse set of statistics to each mention of the spot database. The set of features can be consulted by looking at Table 4.3 (first half). They are used by WAT to train an optional binary classifier (a SVM with linear or RBF kernel) which is used to increase the performance of the spotting procedure.

Most of the features (FID 1-7) try to catch the syntactic properties of a mention and are self-explanatory, while others (FID 8-12) deserve a better explanation. Specifically, the feature *boostedLinksRatio* is defined as $\frac{link(m)}{boostedLink(m)}$, where *boostedLink*$(m)$ is a modified version of $link(m)$ that also accounts for artificial links introduced during the processing of entities relative to persons (two artificial anchors are created respectively for the given name and the first name). The feature *ambiguityRatio* is defined as $\frac{|E_m|}{links(m)}$, while *documentProb* is the $\frac{freq(m)}{|W|}$ with $|W|$ being the number of Wikipedia pages (and thus entities). The feature *mutualDependency* is $\frac{Pr(t_1,\cdots,t_n)^2}{Pr(t_1)\cdots Pr(t_n)}$ where $t_1, \cdots t_n$ are the terms forming the mention m, as in [17]. The features with FID between 13 and 15 are binary features indicating whether the most common sense that might be linked to m is respectively of type PERSON, LOCATION or ORGANIZATION. The same applies to features 16–18 whose value is 1 if the mention m is found inside a span that the OpenNLP [1] NER system recognized as such. The last two features, namely

[1]`http://opennlp.apache.org/`

hasSuperior and *isWhitelisted*, are binary as well, with the former representing the fact that mention m is overlapped by a longer mention m' (with greater length); the latter is only considered in the context of the ERD dataset and it is set to 1 in case the most common entity is part of the entity subset that the ERD challenge [2] assumes valid. In addition WAT provides the possibility to access a set of binary features (with FID starting from 37) indicating the presence of a specific NLP tag inside the mention span.

4.2 Disambiguator

WAT offers a suite of disambiguation algorithms, which can be divided in two main categories: (i) the ones based on the voting framework introduced for the classic TAGME [6]; and (ii) the ones which rely on the *Mention-Entity* graph as a framework for designing collective entity-linking algorithms. Both families exploit a context window surrounding the mention m, denoted by $W(m)$, in order to derive a proper disambiguation context. The function $W(m)$ applied to a given mention m occurring in a text t returns a predefined number (30 by default) of closest mentions surrounding m.

All the implemented disambiguation algorithms support the so called ϵ-trick that works as follows [6]: instead of assigning to the mention m the top-scoring entity e, the system will choose from the top-ϵ scoring entities the one with greater *commonness* or *PageRank* (evaluated on the graph of Wikipedia using SNAP [2]).

4.2.1 Voting algorithms

We can easily describe the original voting procedure introduced in [6] as depicted in Algorithm 1. At the end, each entity E_m will have associated a score s_e, and entity e with the highest score will be selected as representative for the mention m. Scores are then normalized mention-wide.

Algorithm 1 `base` method (TAGME voting scheme)

```
for all $m' \in W(m)$ do
    for all $e' \in E_{m_1}$ do
        $s_e \leftarrow s_e + rel(e, e') \cdot Pr(e'|m')$
    end for
end for
```

In addition to the `base` method, WAT provides three variants in which either (a) trigram-similarity (jaccard) between the title *title*(e) of the entity e and mention m, or (b) the number n of positive votes (number of times when the relatedness is greater than 0), or both indicators are used in addition to the semantic relatedness between e and e' as a way to derive a score for the entity e. We will refer to these variations as: `base-t` (trigram-only), `base-n` (votes only), and `base-nt` (trigram + votes) methods.

4.2.2 Graph-based algorithms

As in [8], the *Mention-Entity* graph $G_{me} = (V_m \cup V_e, E_{me} \cup E_{ee})$ is a graph in which mention nodes are linked to a set of candidate entities, indicating the possible senses for a given mention. Entities are interconnected through weighted edges indicating the semantic similarity among them.

The construction of the graph proceeds as follows: the set of mentions m and its candidate entities E_m found by the

[2]`http://snap.stanford.edu/`

Name	Similarity	Graph Algorithm
`pr`	IDENTITY	PageRank
`ctx-pr`	CONTEXT	PageRank
`comm-pr`	COMMONNESS	PageRank
`ppr`	IDENTITY	Personalized PageRank
`ctx-ppr`	CONTEXT	Personalized PageRank
`comm-ppr`	COMMONNESS	Personalized PageRank
`ppr-uniform`	IDENTITY	Personalized PageRank
`ctx-ppr-uniform`	CONTEXT	Personalized PageRank
`comm-ppr-uniform`	COMMONNESS	Personalized PageRank
`hits-auth`	IDENTITY	HITS - Authority score
`ctx-hits-auth`	CONTEXT	HITS - Authority score
`comm-hits-auth`	COMMONNESS	HITS - Authority score
`hits-hub`	IDENTITY	HITS - Hub score
`ctx-hits-hub`	CONTEXT	HITS - Hub score
`comm-hits-hub`	COMMONNESS	HITS - Hub score
`salsa-auth`	IDENTITY	SALSA - Authority score
`ctx-salsa-auth`	CONTEXT	SALSA - Authority score
`comm-salsa-auth`	COMMONNESS	SALSA - Authority score
`salsa-hub`	IDENTITY	SALSA - Hub score
`ctx-salsa-hub`	CONTEXT	SALSA - Hub score
`comm-salsa-hub`	COMMONNESS	SALSA - Hub score

Table 1: Disambiguation algorithms offered by the WAT framework to work on the Mention-Entity graph.

spotter are respectively used to create the mention nodes (V_m) and the entities nodes (V_e) of the final graph G_{me}. The edges of type E_{me} can be weighted according to one of the following mention-similarity functions: (i) *identity* (always 1), (ii) *commonness* or (iii) *context similarity*, that is the normalized BM25 similarity score [3] between the query q (text around m not including other mentions) and each Wikipedia entity $e \in E_m$ (considered as text documents). A similar weighting strategy is applied to the edges of type E_{ee}. In this case the weighted are calculated using one of the *relatedness* functions that will be introduced in Section 4.4.

Once the graph G_{me} is constructed, a disambiguation procedure is applied to it. The final goal is to derive for each mention node V_m a single entity V_e that represents the "best" sense for that mention, in the context of the mention-entity graph, given the text t. We argue that the "best" entity V_e for the mention V_m is the entity node with the highest score, evaluated by means of one of the well known graph analysis algorithms: such as PageRank, Personalized PageRank, HITS, and SALSA [10]. The application of PageRank is trivial, since it does not require any other parametrization except the damping factor, whose value was set to the classic 0.85. Conversely, the personalized PageRank requires the specification of an initial distribution. WAT offers two alternatives: (i) uniform prior probability or (ii) prior probability derived from the $lp(m)$ attribute of each mention involved in G_{me}. HITS and SALSA algorithms, unlike Pagerank, provide two probabilistic scores: an authority score and a hub score. Table 1 summarizes all possible configurations that can be used to solve the entity linking problem.

4.2.3 Optimizer

After the disambiguation step, it is possible to enable the so called *optimizer* that will perform a second disambiguation pass using the `base-nt` method but with a simple trick: the voting procedure will just consider the votes coming from

[3]`http://lucene.apache.org/`

the entities associated to the just derived annotations A_t instead of using all the possible candidate entities E_m of every mention m. The method `base-nt` was chosen with the following rationale: (i) after the first disambiguation step we argue that the majority of annotations are correct so that (ii) we just need to reconsider just few bindings, and so we link the entity that not only receives the best score and has the greatest trigram similarity but also the ones that was voted by the "right" majority of other senses (forcing cohesion in the annotations).

4.3 Pruner

An ideal pruner has the objective to prune useless annotations computed in the disambiguation step, in order to increase the precision of the system. As in [6], WAT offers the possibility to use the legacy $\rho(a)$ threshold mechanism to prune non-coherent annotations. The ρ is a synthetic metric of a given annotation $a = (m, e)$ that is the average between the link probability $lp(m)$ of mention m and the coherence (average relatedness) of e with its surrounding annotations. It can be expressed as $\rho(a) = \frac{lp(m)+coherence(a)}{2}$ with the coherence being: $\sum_{a' \in W(a)} \frac{rel(e',e)}{|W(a)|}$. The relatedness used is the so called MW-function [14].

In addition to the classic ρ threshold mechanism, WAT offers the possibility to plug in the pruner a binary classifier (a SVM binary classifier with either a linear or RBF kernel) trained on a diverse set of features, both involving the mention and the top-scoring entity. The set of available features is shown in Table 4.3 (the entity specific features are reported in the table on the right). Features with FID between 21 and 26 are local features and they are relative to the entity only. Specifically *pageRank* is the PageRank score of that entity (article) in the Wikipedia graph, *pageHits* is the WikiLinks score extracted from the WikiLinks corpora [4]. The degree features (*inDegree, outDegree*) capture the number of in/out links of that entity. *numRedirects* denotes the number of redirect pages pointing to e, while the *commonness* feature is the prior probability $Pr(e|m)$. Other features worth an explanation are *localCoherence*, *contributions* and *contributionsRatio*. The former differs from the classical coherence in the denominator (called *contributions*) which just counts the number of annotations around a whose relatedness (with a) are non-zero. The same kind of reasoning is behind *contributionsRatio*, which is defined to be the ratio between *contributions* and $|W(a)|$. The feature *pointsTo* denotes the number of annotations around a that are pointed by e, while *pointedBy* is defined as the number of annotations around a that points to e. The *pointsToScore* and *pointedByScore* features are the scaled versions of the previous ones, in which a "point-to/pointed-by" relationship does not count as 1 but rather $\frac{1}{out(e)}$. The last feature, *trigramSimilarity*, is the trigram similarity based on the Jaccard-score between $title(e)$ and the mention m.

4.4 Relatedness functions

Aware of the possible limitations of the relatedness function being used by TAGME, as found in [3], we tried to devise and implement new measures $rel(e_1, e_2)$ which try to quantify how much two entities e_1 and e_2 are related. The function returns a score between 0 and 1, with 1 denoting a strong relation between the two entities and 0 indicating that the two entities are not related at all. For our experiments we considered four different relatedness functions (others have been experimented with lower success and are therefore dropped from this description, but they are available in the WAT framework):

1. Jaccard is the ratio between the intersection and the union of the in-links of e_1 and e_2: $\frac{|in(e_1) \cap in(e_2)|}{|in(e_1) \cup in(e_2)|}$
2. Milne-Witten [14] defined as:
$$1 - \frac{max(log|in(e_1)|, log|in(e_2)|) - log|in(e_1) \cap in(e_2)|}{|W| - min(log|in(e_1)|, log|in(e_2)|)}$$
3. LSI vector cosine similarity between two LSI vectors extracted using the `gensim` [5] library.
4. A special (empirical) measure obtained by precomputing the intersection $|in(e_1) \cap in(e_2)|$ for all the possible pairs of entities (e_1, e_2). The intersection frequency distributions were stored in logarithm buckets identified by the pair $\langle log_2(e_1), log_2(e_2)\rangle$, with $|in(e_1)| < |in(e_2)|$. We then dropped from each distribution the first 80% pairs thus focusing our attention on the long tail. The relatedness is then defined to be the value of CDF at point $i = |in(e_1) \cap in(e_2)|$ of the distribution indexed by $\langle log_2(e_1), log_2(e_2)\rangle$.

FID	Feature
1	*allUpper*
2	*isCapitalized*
3	*numCapWords*
4	*ratioCapWords*
5	*mentionLength*
6	*numTokens*
7	*averageTokenLength*
8	*linkProb*
9	*boostedLinksRatio*
10	*ambiguityRatio*
11	*documentProb*
12	*mutualDependency*
13	*isPerson*
14	*isLocation*
15	*isOrganization*
16	*insidePersonSpans*
17	*insideLocationSpans*
18	*insideOrganizationSpans*
19	*hasSuperior*
20	*isWhitelisted*
37–	*NLP tags*

FID	Feature
21	*pageRank*
22	*pageHits*
23	*inDegree*
24	*outDegree*
25	*numRedirects*
26	*commonness*
27	*rho*
28	*localCoherence*
29	*coherence*
30	*contributions*
31	*contributionRatio*
32	*pointsTo*
33	*pointedBy*
34	*pointsToScore*
35	*pointedByScore*
36	*trigramSimilarity*

Table 2: Mention and Entity feature sets

5. EVALUATION

In order to have a clear picture of the performance of various instantiations of our system, we devised three different tests (inspired to [4]). The first one, called *Spotting coverage*, determines the level of coverage of our spotter with respect to the gold-standard dataset: namely, the mentions in the gold-standard are compared with the mentions our system identifies in the input text. The second test, called *D2W*, evaluates the performance of our disambiguation algorithms: namely, the mentions of the gold-standard are sent as input to our system that, by running one of the proposed disambiguation algorithms, returns a mention-entity assignment for each (gold) mention. The assignments are compared with the ones in the gold-standard and F1/Precision/Recall

[4]`https://code.google.com/p/wiki-links/`

[5]`http://radimrehurek.com/gensim/` with $k = 400$

measures are derived. The last test deals with the performance of the system as a whole. The input text is annotated via the entire annotation pipeline, and F1/Precision/Recall measures are derived by comparing the detected mention-entity annotations with the ones in the gold-standard.

For the sake of clarity we now define the concept of *weak* and *strong* annotation match between two annotations a_1 and a_2, as in [4]. The former, denoted by $M_w(a_1, a_2)$, is true *iff* the mentions m_1 and m_2 textually overlap and e_1 equals e_2, while the latter denoted by $M_s(a_1, a_2)$ is true *iff* a perfect text-alignment occurs between the two mentions m_1 and m_2 in addition to the entity equality property.

5.1 Datasets

In this work we will concentrate our attention onto the ERD dataset, but the conclusion and results can be extended to all the available datasets such as (AIDA/CoNNL, AQUAINT, IITB and MSNBC). We do not report a detailed analysis of those datasets for the sake of space, details are deferred to a future paper. The datasets we considered are:

ERD-50 contains a subset of the original ERD-100 corpus, consisting of 50 annotated documents as described on the ERD challenge site [6].

ERD-100 is the complete development dataset. We didn't have access to it but it was possible to test a given WAT configuration getting the macro averaged F1, Precision and Recall.

5.2 Spotting

Under the assumption of strong/weak annotation match, the benchmark for the spotting can be easily defined. We consider as *true positives* the mentions reported in the gold-standard that are also found by the annotator (ignoring the entity equality property that, for this test, is not defined). *False positives* are the mentions found by the system that are not in the gold-standard. Conversely *false negatives* are the mentions that appear in the gold-standard but are not reported by the annotator. Given these definitions we can derive the classical IR metrics: Precision, Recall and F1 for each type of match: either strong or weak.

Being the first stage of the processing pipeline of an annotator, we can tune the performance of spotting in three different ways. We can (a) maximize the recall of the spotter, leaving to the pruner the burden to remove spurious mentions at the end of the pipeline. The second option, under the assumption of having a not-so robust disambiguator and a very fragile pruner, is to (b) maximize the precision of the spotter. The third option is (c) maximizing the F1, even if it might be not the best choice in most cases because it assumes the presence of an adequate disambiguator (that is robust to limited noise in the spotting procedure) and a very simple pruner. Therefore if a spotter does not return an F1 near to 1 it is usually better to optimize the recall.

We report the performance of WAT's spotter against the Stanford NER system in Table 3 and Table 4. The comparison against Stanford NER is important since it is currently used as a spotter module of several annotators like AIDA [8]. Indeed it is the one offering the best F1, but surprisingly we argue that using a spotter that is based on Wikipedia, like the one provided by WAT, is the best choice with respect

[6]`http://web-ngram.research.microsoft.com/erd2014/`

Dataset	Annotator	Thr.	P	R	F1
ERD-50	WAT	.266	.557	.641	.564
ERD-50	WAT	.000	.090	**.986**	.160
ERD-50	STANFORD		.560	.673	**.573**

Table 3: Spot experiment using the weak match $M_w(a_1, a_2)$, with threshold over $lp(m)$.

Dataset	Annotator	Thr.	P	R	F1
ERD-50	WAT	.266	.467	.588	**.493**
ERD-50	WAT	.000	.076	**.919**	.136
ERD-50	STANFORD	-	.296	.381	.312

Table 4: Spot experiment using the strong match $M_s(a_1, a_2)$, with threshold over $lp(m)$.

to efficiency and flexibility. Indeed Stanford NERs and others are orders of magnitude slower, taking seconds rather than milliseconds. Conversely the WAT's spotter is much faster and capable of handling a diverse set of datasets either by a simple threshold mechanism or by introducing a more sophisticate spotting procedure based on the features introduced in Section 4.1. This conclusion can be drawn by just taking a closer look to the Recall metric (in case no threshold is applied). We argue that by properly tuning the spotter model we can reach a good compromise between mention coverage and introduction of spurious noise that might affect the successive steps of the disambiguation pipeline.

5.3 Disambiguation

We tested the performance of each disambiguation algorithm in isolation, using the D2W benchmark. The D2W benchmark can be described as follows. The input document and the set of correct mentions are passed to the annotator, which is asked to only disambiguate the gold-mentions. The annotator is free to choose whether to use other sources of information in the input text, such additional mentions or lexical features in the input text (*full D2W*), or not (*light D2W*). The results of the light-D2W experiment are present in Table 5, which shows the performance upper bound of the system under the assumption of a perfect spotting procedure. We set the parameter *epsilon* to 0 in order to obtain results that just depend onto the disambiguation algorithm procedure (and not on the other steps).

As the reader can observe, the vast majority of the disambiguators report a close F1's value (between $0.83 - 0.84$), thus suggesting that the problem is not the disambiguation algorithm itself, but rather the spotting phase. In order to further prove our point and in order to test the robustness of each algorithm in presence of noise (derived from a not so precise spotting function), we can pass to the system, in addition to the mentions that are present in the gold-standard, other mentions derived from a spotting procedure of the input text. The peculiarity of this test is that *false positives* relative to mentions that are not found in the gold-standard are not taken in consideration but they are just used to influence the disambiguation algorithm providing a sort of (useful or not) contextual information. The results of this experiment are reported in Table 6 for the top scoring configurations previously highlighted. The performance loss in F1 with respect to the best scoring algorithms of Table 5 is reported in parenthesis.

Dataset	Method	P	R	F1
ERD-50	`base`	.871	.781	.817
ERD-50	`base-t`	.891	.793	**.831**
ERD-50	`base-nt`	.884	.787	.824
ERD-50	`base-n`	.871	.781	.817
ERD-50	`pr`	.862	.773	.808
ERD-50	`ctx-pr`	.839	.766	.795
ERD-50	`comm-pr`	.874	.784	.820
ERD-50	`ppr`	.866	.776	.812
ERD-50	`ctx-ppr`	.839	.765	.795
ERD-50	`comm-ppr`	.893	.801	**.838**
ERD-50	`ppr-uniform`	.862	.774	.809
ERD-50	`ctx-ppr-uniform`	.837	.764	.793
ERD-50	`comm-ppr-uniform`	.890	.799	.835
ERD-50	`hits-auth`	.894	.797	**.834**
ERD-50	`ctx-hits-auth`	.802	.722	.752
ERD-50	`comm-hits-auth`	.894	.792	.830
ERD-50	`hits-hub`	.843	.754	.789
ERD-50	`ctx-hits-hub`	.851	.776	.806
ERD-50	`comm-hits-hub`	.833	.748	.782
ERD-50	`salsa-auth`	.903	.805	**.843**
ERD-50	`ctx-salsa-auth`	.808	.725	.756
ERD-50	`comm-salsa-auth`	.878	.778	.816
ERD-50	`salsa-hub`	.873	.783	.819
ERD-50	`ctx-salsa-hub`	.874	.797	.828
ERD-50	`comm-salsa-hub`	.873	.783	.819

Table 5: Light D2W results on ERD-50 dataset

Dataset	Method	P	R	F1
ERD-50	`base-t`	.731	.731	.731 (-.100)
ERD-50	`comm-ppr`	.811	.812	.811 (-.027)
ERD-50	`hits-auth`	.797	.800	.798 (-.036)
ERD-50	`salsa-auth`	.806	.809	.807 (-.036)

Table 6: Full D2W results on ERD-50 dataset

All the algorithms suffer from spurious noise being introduced by a loose spotter, but the algorithms based on the mention entity graph seem to be more robust. The drop is more pronounced for the `base-t` scheme (10% drop), while the other methods are less affected (3–4% drop). The results prove that our algorithms are reasonably robust in that they can handle incorrect mentions without the risk of topic drift. However the use of a not-so-precise spotter must be taken with care because the introduction of more mentions makes the decision of the pruner more difficult. In fact the disambiguator will always try to find a mention-entity binding that makes sense to the overall text, or a window of it, and the spurious mentions could wrongly affect the disambiguation of the correct ones.

Dataset	Method	P	R	F1
ERD-50	TAGME	.762	.698	.725
ERD-50	`base-t`	.731	.731	.731 (-.100)
ERD-50	`comm-ppr`	.811	.812	.811 (-.027)
ERD-50	`hits-auth`	.797	.800	.798 (-.036)
ERD-50	`salsa-auth`	.806	.809	.807 (-.036)

Table 7: SA2W to D2W results on ERD-50 dataset

We finally introduce a third experiment, the so called *SA2W to D2W reduction* experiment, that is also useful to understand the improvement in disambiguation performance with respect to classic TAGME. In this case a document t is fully annotated without any auxiliary information being provided to the system and then only the annotations matching the ones in the gold-standard (using $M_w(a_1, a_2)$) are considered for the final evaluation. The results of this final test are reported in Table 7, which are in complete accordance to what we have found with the previous test. The improvement over TAGME is in the range 1–9%, with `comm-ppr` being the top-scoring disambiguation algorithm on this test.

5.4 SA2W benchmark

In this section we report the results of the SA2W experiment (using $M_w(a_1, a_2)$) under the assumption of a spotting procedure that feeds all the possible spots to the annotator's pipeline. The results and the performance loss in F1 with respect to the results obtained in the previous test (in parenthesis) are reported in Table 8.

Dataset	Method	Thr.	P	R	F1
ERD-50	TAGME	.242	.481	.513	.455 (-.270)
ERD-50	`base-t`	.180	.466	.536	.456 (-.275)
ERD-50	`comm-ppr`	.180	.493	.580	.489 (-.322)
ERD-50	`hits-auth`	.172	.480	.551	.475 (-.323)
ERD-50	`salsa-auth`	.156	.466	.586	.477 (-.330)

Table 8: SA2W experiment using $M_w(a_1, a_2)$, with threshold over ρ

You can see how dramatic the drop in performance is and how the use of all possible mentions makes the methods almost identical in performance. The only possible way to intervene is either by plugging a more sophisticated pruning module or a more clever spotter module with the aim of minimizing the chance of introducing false positives.

6. ERD TUNING

All the choices made during the tuning of the system were done by using the leaderboard system before the ERD-50 dataset leaked in the mailing list. Thus some of the choices made in this phase are not in accordance to what detailed in the previous sections and in general might have been non-optimal.

Aware of the possible limitations of the spotter and the pruner modules we started doing a brute-force search over the parameters $lp(m)$ and $\rho(a)$ with reasonable step sizes using all the disambiguation algorithms. The three top-scoring algorithms are shown below:

Method	Thr. $lp(m)$	Thr. $\rho(a)$	P	R	F1
`base-t`	.16	.16	.802	.651	.718
`comm-ppr`	.16	.16	.816	.657	.727
`hits-auth`	.16	.16	.803	.651	.719

The `base-t` method was selected as disambiguation algorithm since the difference between the methods was minimal with such a strict thresholding mechanism. We then evaluated the performance of the system using a diverse set of relatedness functions. The results were in favor of the Jaccard measure:

Relatedness	P	R	F1
jaccard	.802	.651	.718
empirical	.808	.644	.717
lsi	.802	.645	.715
mw	.793	.644	.710

We then analyzed how both ϵ and the sorting by either *commonness* or *PageRank* might have impacted the final results. The results revealed the top scoring configuration to be the one with $\epsilon = 0$ hence with the ϵ-trick disabled, bringing up the F1 to 72.5%. At this point we took a look at errors in the logs revealing a set common error patterns in the annotated documents. We then came up with a filter implementing four different rules:

1. Remove all the annotations that link to entities not recognized as valid by the ERD Challenge
2. Remove all the annotations with mentions that are lower case
3. Remove all the ambiguous annotations that have *contributions* equal to 0
4. Re-enable all the pruned annotations that have the same mention of a currently active annotation (resulting in a primitive co-reference resolution mechanism).

The use of rule system and the enabling of the *optimizer* allowed us to improve the precision bringing the F1 score to 74.8%. This was the final configuration we submitted to the challenge that registered a final F1 67.2% score, with an impressive precision of 87.6% but very low recall 54.5%.

In order to increase the recall of the system we tried to intervene on the spotting module by plugging a binary classifier trained on different publicly available datasets. The same was done for the pruner module. Unfortunately the use of different datasets didn't provide the generality (they actually worsen the overall performance of the system) we were hoping for and therefore we discarded the option of using machine learning techniques to improve the performance the system, since the risk of over-fitting was too high.

Just for the sake of completeness we document our overfitted configuration (using the ERD-50 dataset), derived by training a linear SVM with L2 loss function and L1 penalty both for the spotter and the pruner module. The system scored 80% (91.5% Precision and 71.2% Recall) on the test dataset using the `comm-ppr` method.

7. CONCLUSIONS

In recent years the literature around entity linking and entity annotators has focused its attention on the disambiguation step of the annotation pipeline, thus ignoring the issues raised by mentions recognition and annotations pruning (see e.g.[3, 8, 7, 18]). At the end we can confess that in attacking the ERD challenge we did the same mistake, spending lot of time and efforts in trying to improve and implement new disambiguation algorithms based on the recent trends in the literature [7, 18]. The main conclusion of our extensive experiments, that we will detail in a future paper, is that the spotter and the pruner modules need a complete redesign. They are responsible of the introduction of many false positives, which provoke the significant gap in the WAT's performance witnessed by the figures reported in Table 8 versus Table 6.

8. ACKNOWLEDGEMENTS

Part of this work has been supported by Google Faculty research award 2013 and MIUR PRIN grant ARS TechnoMedia.

9. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, Z. Ives DBpedia: A nucleus for a web of open data. In *Proc. ISWC*, 722–735, 2007.

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

[3] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, S. Trani Learning Relatedness Measures for Entity Linking. In *Proc. ACM CIKM*, 139–148, 2013.

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

[5] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, 708–716, 2007.

[6] P. Ferragina, U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by Wikipedia entities). In *Proc. CIKM*, 1625–1628, 2010.

[7] X. Han, L. Sun, J. Zhao Collective Entity Linking in Web Text: A Graph-Based Method In *Proc. SIGIR*, 765–774, 2011.

[8] 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.

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

[10] R. Lempel, S. Moran SALSA: The Stochastic Approach for Link-structure Analysis. In *ACM Trans. Inf. Syst.*, 131–160, 2001.

[11] E. Meij, K. Balog, D. Odijk. Entity linking and retrieval for semantic search. In *Proc. WSDM*, 683–684, 2014.

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

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

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

[15] M. Sozio, A. Gionis The Community-search Problem and How to Plan a Successful Cocktail Party. In *KDD*, 939–948, 2010.

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

[17] A. Thanopoulos, N. Fakotakis, G. Kokkinakis, Comparative Evaluation of Collocation Extraction Metrics. In *Proc. LRE*, 620–625, 2002.

[18] R. Usbeck, A. Ngomo, M. Röder, D. Gerber, S. Coelho, S. Auer, A. Both AGDISTIS-Agnostic Disambiguation of Named Entities Using Linked Open Data. In *European Conference on Artificial Intelligence (Poster)*, 2014.