TAGME: on-the-fly annotation of short text fragments (by Wikipedia entities)

Ferragina Paolo Scaiella Ugo

Dipartimento di Informatica
University of Pisa, Italy
{ferragina,scaiella}@di.unipi.it

Abstract

In this paper we address the problem of accurately and efficiently cross-referencing text fragments with Wikipedia pages, in a way that structured knowledge is provided about the (unstructured) input text by resolving synonymy and polysemy. We take inspiration from the invited talk of Chakrabarti at WSDM 2010, and extend his proposed scenario from the annotation of entire documents to the annotation of short texts, such as snippets of search-engine results, tweets, news, etc.. These short and poorly composed texts pose new challenges in terms of efficiency and effectiveness of the annotation process, that we address by proposing TAGME, the first system that performs an accurate and on-the-fly annotation of these short textual fragments. A large set of experiments shows that TAGME significantly outperforms state-of-the-art algorithms [11, 15] when they are adapted to work on short texts, and surprisingly, it results competitive (if not superior!) on long texts with the *plus* of being faster.

1 Introduction

The typical IR-approach to indexing, clustering, classification and retrieval, just to name a few, is that based on the bag-of-words paradigm [3]. In recent years, a good deal of work attempted to go beyond this paradigm with the goal of improving the search experience on unstructured textual data as well as on structured or semi-structured data. In his invited talk at WSDM 2010, S. Chakrabarti surveyed this work categorizing it in three main classes: (a) adding structure to unstructured data, (b) adding structure to answers, and (c) adding structure to queries while avoiding the complexity of elaborate query languages that demand extensive schema knowledge. In this paper we will be concerned with the first issue, which consists of *identifying* a sequence of terms (also called *spots* in [11]) in the input text and of *annotating* them with un-ambiguous entities drawn from a catalog. The choice of the catalog is obviously crucial for the success of the approach; currently several systems adopt Wikipedia pages or derived concepts as entities (see e.g. [11, 15] and Sect. 3 for more details), because of their ever-expanding number (more than 3 million English pages, and more than 500K pages in each major European language) and the fact that Wikipedia offers the best trade-off between a catalog with a rigorous structure but with low coverage (like the one offered by the high-quality entity catalogs s.t. `WordNet`, `CYC`, `OpenCYC`, `TAP` [8]), and a large text collection with wide coverage but unstructured and noised content (like the whole Web). These systems have already shown that this annotation process provides a stunning contextualization of the input text helpful for improving subsequent searching, classification or clustering tasks.

In this paper we add to this flow of work the specialty that the input texts to be annotated are

very short, namely composed of few tens of terms. The context of use we have in mind is the annotation of either the results of a search engine (a.k.a. web-snippets), or the tweets of a Twitter channel, or the items of a news feed, or the posts of a blog, etc.. These poorly composed texts pose new challenges in terms of efficiency and effectiveness: (1) the annotation should occur on-the-fly, because in those contexts data may be retrieved at query time and thus cannot be pre-processed; (2) the annotation needs new algorithms because the input texts are so short that it is difficult to mine significant statistics and rely on other good contextual-information which would be surely available in the case of the "entire document".

Given these goals, we have designed TAGME,[1] a system that is able to accurately and efficiently cross-reference short text fragments with Wikipedia pages (a.k.a. senses/concepts), in a way that it resolves synonymy and polysemy (in a language-independent manner).[2] This is extremely informative, with implications which go far beyond the enrichment of a text with explanatory links (as also pointed out by Chakrabarti in his WSDM10's talk). In fact, any task that is currently addressed using the bag-of-words model, or with knowledge obtained from less comprehensive knowledge bases (such as the ones mentioned above), could benefit from using TAGME to draw upon (the millions of) Wikipedia topics instead.

Technically speaking, TAGME follows the approach proposed in [11, 15] and thus uses Wikipedia *anchor texts* as entities and their (potentially many) *pages* linked in Wikipedia as possible annotations for them. TAGME receives a *short* text in input, detects the anchors occurring in it, and then tries to annotate them in a unique way by possibly discarding some of these anchors if they are considered un-important or un-meaningful for the topics discussed in the input text. TAGME aims, like prior work [11], at the collective agreement among all anchor annotations by deploying a powerful relatedness function among concepts (proposed in [14]) together with some other statistics drawn from Wikipedia (see Sect. 2). However, unlike prior works [11, 15], TAGME deploys this information within *new scoring functions* which take into account the *sparseness* of the anchors in the short input text, and allow to disambiguate and possibly discard some of the candidate annotations in a way that is fast and effective in the quality of the final assignment.

The performance of TAGME has been validated through a large set of experiments which involve standard datasets, as the ones offered by IITB [11] and consisting of over a hundred manually annotated Web pages and tens of thousands of text entities, as well as new datasets that we have constructed and make available to the community (see Sect. 5.3). Our datasets consist of millions of short text fragments drawn from Wikipedia and Tweeter. Overall our experiments show that TAGME significantly outperforms state-of-the-art algorithms [11, 15] when they are adapted to work on short texts, and surprisingly, it results also competitive (if not superior!) on long texts with the *plus* of being much faster.

2 Notation and terminology

A *(text) anchor* (referred also as *spot* in [11]) for a page p is a text used in another page to point to p. This concept is borrowed from the Web search, where anchors were used to enrich Web-page descriptions for indexing purposes. In the Wikipedia context, an anchor text pointing to p can be the title of p, one of its synonyms or acronyms, or even long phrases which might be (much) different from a syntactic point of view from p's title. As an example, an anchor text for the page "Nintendo DS" is the acronym "nds" as well as the phrase "Gameboy ds" or "Nintendo Dual Screen". For coverage purposes, we assume to enrich all anchors of p with other descriptive phrases such as the title of the Redirect pages that link to p. Overall, this approach derives a total of about 8M distinct anchors from Wikipedia (English).

Because of polysemy and variant names, the same anchor a may occur in Wikipedia pointing to many different pages. So we denote by $P(a)$ the set of *all pages* linked by a, and use some well-known *basic*

[1] A preliminary version of the software can be tested at `http://tagme.di.unipi.it`.

[2] TAGME runs on English-Wikipedia but, of course, nothing prevents to make it work on other languages present in Wikipedia.

scoring functions to evaluate the "significance" of an occurrence of a as an anchor or the "significance" of annotating anchor a with the page $p \in P(a)$. In particular, we set: $freq(a)$ as the number of times the text a occurs in Wikipedia (as an anchor or not); $link(a)$ as the number of times the text a occurs as an anchor in Wikipedia (of course $link(a) \leq freq(a)$); $lp(a) = link(a)/freq(a)$ denotes the probability that an occurrence of a has been set as an anchor (a.k.a. *link probability* [15]).

Given a page $p \in P(a)$, we call the *prior probability* $\Pr(p|a)$ as the *commonness* of page p among the senses of a. This value denotes the probability that, given an occurrence of a as anchor, this is an anchor to page p. It is clear that the entropy of $\Pr(p|a)$, computed over all $p \in P(a)$, would give us a measure of the ambiguity of a, in terms of "related" Wikipedia senses, and thus it would measure the difficulty to annotate a with some page $p \in P(a)$. This annotation will be denoted by $a \mapsto p$. Often a has more senses, thus $|P(a)| > 1$, so we call *disambiguation* the process of selecting one of the possible senses of a from $P(a)$. It goes without saying that not all occurrences of text a should be considered as text anchors: think to the occurrence of the single term "act", sometimes it refers to meaningful entities like Act of Parliament, act of a drama, or is the acronym of the Australian Capital Territory; some other times it is used as a verb. Thus, by following [11] we introduce a *fake page* NA and use the annotation $a \mapsto \text{NA}$ in order to denote that an occurrence of anchor a has been *discarded*, which means "no assignment" for that occurrence of a in the input text.

Finally, we will denote by $\mathcal{A}_T$ the set of all anchors detected in a text T, and use $\mathcal{M}(\mathcal{A}_T) = \{a \mapsto p \,|\, a \in \mathcal{A}_T, p \in P(a) \cup \{\text{NA}\}\}$ to denote the set of *all selected assignments* for the anchors in T. The *goodness* of an assignment process depends on the significance of the mappings in $\mathcal{M}(\mathcal{A}_T)$. The main goal of this paper will be to *efficiently* compute a *good* assignment $\mathcal{M}(\mathcal{A}_T)$ for input texts T which are *short*.

3 Related Works

The literature offers two main approaches to *contextualize* a (possibly short) text in order to empower subsequent IR-steps such as clustering, classification, or mining.

One approach consists of extending the classic term-based vector-space model with additional dimensions corresponding to *features (concepts)* extracted from an external knowledge base, such as DMOZ [5, 6], Wikipedia [7, 1, 10], or even the whole Web (such as the Google's kernel [17]). Probably the best achievements have been obtained by querying Wikipedia (titles or entire pages) by means of short phrases (possibly single terms) extracted from the input text to be contextualized. The result pages (typically restricted to the top-k) and their scores (typically *tf-idf*) are used to build a vector that is considered the "semantic representation" of the phrase. Vectors are then combined (via centroid or other mining approaches) to derive the "semantic representation" of the input text, which is finally used in classification [7], clustering [1, 10], or searching [21] processes. The pro of this approach is to extend the bag-of-words scheme with more concepts, thus possibly empowering the identification of *related* texts which are syntactically far apart. The cons is the contamination of these vectors by un-related (but common) concepts retrieved via the syntactic queries.

In order to overcome these difficulties, more and more authors have recently tried to *annotate* only the *salient* entity references present in the input text, without resorting to the vector-space model. Their key idea is to identify in the text short-and-meaningful sequences of terms and connect them to an *unambiguous sense (entity)* drawn from a catalog. Entities can be either Named Entities (see e.g. [19, 22]) drawn from a small set of specifically recognized types—most often People and Locations— or they can be *short phrases* (see e.g. [11, 15]) drawn from a large knowledge base, such as Wikipedia. In the former case, substantial training or human effort is needed in knowledge representation and linguistics which eventually provides a very *"coarse"* annotation. A typical example consists of two texts that contain the entity *Michael Jordan*; these sys-

tems would certainly recognize that it is the name of a person, but they would miss to *disambiguate* which *Michael Jordan* the occurrence is referring to (e.g. Wikipedia reports at least seven VIPs with that name).

More recently, some authors tried to deploy *much larger* catalogs composed of millions of *senses*, typically represented by Wikipedia pages or derived concepts. They are promising because of the ever-expanding size of Wikipedia (more than 3 million English pages, and more than 500K pages in each major European language) and because it offers the best trade-off between a rigorous structure with low coverage (like the one offered by the high-quality entity catalogs such as `WordNet`, `CYC`, `OpenCYC`, `TAP` [8]), and the large coverage of collections composed by unstructured and noised texts (like the whole Web). Wikipedia is probably the most impressive example of the Web-2.0 trend in which tagging and cataloguing of knowledge has been opened to the masses: it has 340K categories and 3M pages, and keeps up with worlds event on an hourly or daily basis. The negative side of using Wikipedia as an entity catalogue is that there is little schema, the authorship is missing, and its categories are "haphazard, redundant, incomplete and inconsistent" [12].

This is the reason why more and more authors are trying to extract some *useful structured knowledge* from its content and links. For example, [9] derived *concept relations* by exploiting some special pages that are present in Wikipedia like Disambiguation pages (for polysemy), Redirect pages (for synonymy) and Category links (for hyperonymy). Other authors (such as [14]) proposed to deploy the links between Wikipedia pages to infer other *concept relations*, and [11] showed that this improves the precision and recall of what is achievable via just Wikipedia categories. Yet other authors tried to build Wikipedia-derived ontologies: such as YAGO [18], Kylin [20] and DBpedia [3], in order to provide a machine readable and consistent representation of the entities and facts contained in Wikipedia. More recently, some research groups proposed a new approach which consists of *annotating* an input text by cross-referencing some of its fragments with proper Wikipedia articles. As observed in the Introduction and in [11, 15], this approach has implications that go far beyond the enrichment of texts with explanatory links because it can provide structured knowledge about any unstructured fragment of text. This way, any task that is currently addressed with bags of words-indexing— such as clustering, retrieval, and summarization, just to name a few— could use these techniques to draw on a vast network of concepts and semantics (refer to Chakrabarti's invited talk).

A common feature among all these approaches is that they exploit few *million anchor texts* in Wikipedia as spots to be annotated via two main steps: *anchor disambiguation* and *anchor pruning*. Disambiguation is the task that selects the best *sense* for an anchor (i.e. assigns an anchor to a Wikipedia page that best describes its meaning); pruning is the process that possibly discards some detected anchors because not interesting for the *contextualization* of the input text. Recall that (see Sect. 2), this means associating an anchor to the *fake page* NA.

To our knowledge the first work that addressed the problem of linking anchor texts to Wikipedia pages was WIKIFY [13]. Here, disambiguation is performed by comparing the context of the anchor (i.e. terms that surround it in the input text) with the context of all its possible senses (i.e. terms that surround citations of those senses/pages in Wikipedia); anchor pruning is performed by using a threshold on the link probability of the selected annotation (see Sect. 2). This approach yields a disambiguation precision of 93% (recall is 83%), but it may be slow in that it requires the comparison among many term-vectors that cannot be kept in main memory. This clearly prevents its use in our *on-the-fly* annotation process.

In the same year, Cucerzan [4] proposed an annotation process that in some sense mixes the two approaches above— feature-based vs entity-based— and was the first to recognize the importance of the *inter-dependence* between entity annotations. It first performs anchor pruning by using NE-techniques that exploit Wikipedia and Web statistics; then it does disambiguation by representing each page p with an high-dimensional feature vector $f(p)$ that exploits context terms (of the anchors pointing to p) and cat-

[3] `http://dbpedia.org`

egory links of p. Then a vector $g(\mathcal{P})$ of the same type is built for the entire input text T by considering terms and category links *drawn from the set of all possible senses of the anchors* occurring in T: namely, $\mathcal{P} = \{p \mid p \in \bigcup_{a \in \mathcal{A}_T} P(a)\}$. Then the score of an annotation $a \mapsto p$ is computed as the dot-product between $f(p)$ and $g(\mathcal{P} \setminus \{p\})$. This takes into account the candidate annotations of the other entities in T, but presents two major limitations: it is based on the Wikipedia category hierarchy (see above), and $g(\mathcal{P} \setminus \{p\})$ is contaminated with all possible disambiguations of all anchors, so its use for "agreement with the majority" may be misleading.

Recently Milne and Witten [15] proposed an approach that yielded considerable improvements by hinging on three main ingredients: (i) the identification in the input text of the set C of so-called *context-pages*, namely pages drawn from anchors that are not ambiguous (because they link to just one page/sense); (ii) a measure $rel(p_1, p_2)$ of *relatedness* between two pages p_1, p_2 based on the overlap between their in-linking pages in Wikipedia; and finally (iii) a notion of *coherence* of a page p based on the relatedness of p to the other context pages $c \in C$. Given these, the disambiguation of an anchor a is then obtained by using a classifier that exploits for each sense $p \in P(a)$: the commonness $\Pr(p|a)$ of the annotation $a \mapsto p$, the relatedness $rel(p, c)$ between the candidate sense p and all context-pages $c \in C$, and the coherence of each c with respect to the entire input text. Then anchor pruning is performed by using another classifier that mainly exploits the location and the frequency of the anchor in the input text, the link probability, the confidence of disambiguation (assigned by the classifier at the previous step) and the relatedness of the disambiguated sense with respect to the un-ambiguous pages $c \in C$. In [15] the authors showed an impressive precision of 97% for disambiguation and an F-Measure of 74.8% for pruning. It must be said that this approach is designed to deal with a text that is *"reasonably long and focused"*[4], so it seems unsuitable for our context of annotation where C might be empty or too small in short texts. Experiments in Sect. 5.6 validate such negative expectations!

Last year, Chakrabarti and his group [11] proposed a method based on two main novelties. The first one was to score an annotation $a \mapsto p$ with two terms: one *local* to the occurrence of a and the other *global* to the text fragment. The local score involves the context terms of a in the input text and 12 features built upon the context terms of p and some similarity functions. The global score involves all the other annotations $a' \mapsto p'$ detected for the input text and, inspired by [14], this score is computed as the sum of the relatedness between p and the other pages p'. The average of these two scores provides *the* score for an assignment $a \mapsto p$, given the others. The second novelty of this paper was to model the annotation process as a search of the mapping of all anchors that *maximizes* the sum of their scores. Actually Chakrabarti et al. fused the two steps— anchor disambiguation and pruning— by introducing the fake entity NA (whose relatedness with other pages is zero), and a parameter ρ_{NA} in the objective function that balances precision vs recall. Extensive experiments showed that this approach overcomes Cucerzan's algorithm, yields a precision score comparable to Milne&Witten's system, but reaches a considerable higher recall. Unlike the others, this approach does not offer any evidence that it could work badly on short fragments: actually, the only *local score* achieved high performance in [11]. However, as it is shown in Figure 13 of [11], the system is not fast since it takes > 2 seconds over texts of about 15 anchors. This is acceptable for an off-line setting, like the one considered in [11], but it is unsuitable for our setting where we wish to annotate *on-the-fly* many text fragments (think to the text snippets of a search engine, or a tweet flow).

In the light of this literature, we will compare in Sect. 5 our system TAGME against the two best-known systems: namely, Chakrabarti's and Milne&Witten's one, together with some other *baseline* systems that will be fast and behave surprisingly well on our datasets.

[4] `http://wikipedia-miner.sourceforge.net` This is the response message of the system by Milne&Witten when the input text is too short.

4 Our Proposal

In this section we describe TAGME, a software system that annotates short fragments of text *on-the-fly and with high precision* by cross-referencing some meaningful text spots (i.e. anchors drawn from Wikipedia) with pertinent concepts (i.e. Wikipedia pages). This is obtained in two main phases, namely anchor disambiguation and anchor pruning. Disambiguation will be based on finding the *"best agreement"* among the concepts assigned to the anchors detected in the input text, while pruning will evaluate the *"significance"* of an anchor-sense annotation by considering the relatedness among the assigned senses and their link probability, in order to possibly drop the annotations which result non-pertinent with the topics the input texts talks about.

So the structure of TAGME mimics the one of [11, 15]'s systems and, as done in these papers, it aims at the collective agreement among all anchor annotations. However, unlike prior works (and as detailed in the next sections), TAGME deploys some *new scoring functions* that evaluate the *"significance"* of an anchor annotation in a way that is fast and effective in the quality of the final assignment. The following subsections will detail our algorithmic choices, and then experiment TAGME showing that it achieves the best known precision/recall with *on-the-fly* speed of annotation over short text fragments. When TAGME is adapted to work on long texts, our experiments will show that it competes favorably against the best-known systems resulting superior either in annotation accuracy (wrt. [15]) or much faster (wrt. [11]).

4.1 Preprocessing

TAGME indexes some distilled, but useful, information drawn from the Wikipedia snapshot of November 6, 2009.

Anchor dictionary. We took all anchors present in Wikipedia pages, augmented them with the titles of Redirect pages plus some variants of the page-titles, as suggested in [4]. We then removed the anchors composed by one character or just numbers, and also discarded all anchors a whose absolute frequency ($link(a) < 2$) or its relative frequency ($lp(a) < 0.1\%$) is small enough to argue that they are unsuitable for annotation and probably misleading for disambiguation. The final dictionary contains about 3M of "valid" anchors, and it is indexed by Lucene[5].

Page catalog. We took all Wikipedia pages and discarded disambiguation pages, list pages, and redirect pages, because un-suitable as concepts/senses. About 2.7M pages remained, they were indexed with Lucene by building documents consisting of two fields: the first one contains the body of the page, the second one contains all anchors and context terms that are used as citations to that page.

In-link graph. This is a directed graph whose vertices are the pages in the Page Catalog, and the edges are the links among those pages as derived from the Wikipedia-dump called "Page-to-page link records". This dump consists not only of the links inserted in the page body, but also of links that are added by template expansions. This expands the "in-body links" of a factor 2, thus obtaining a graph of about 147M edges. This graph is indexed in internal-memory by Webgraph[6], taking 190Mb.

4.2 Anchor parsing

TAGME receives a short fragment of text as input, tokenizes it, and then detects the text anchors by querying the *Anchor dictionary*. Since anchors may overlap or be substring one of another, we need to detect their boundaries. We simplified the approach of [4] in the following way: if we have two anchors a_1, a_2 s.t. a_1 is a substring of a_2, we drop a_1 only if $lp(a_1) < lp(a_2)$. This is because a_1 is typically more ambiguous than a_2 (being one of its substrings), and editors like to link more specific (i.e. longer) term sequences. Therefore, we prefer to discard a_1 in order to ease the subsequent disambiguation task. As an example, consider a_1 = "jaguar" and a_2 = "jaguar cars": in this case if we didn't discard a_1, disambiguation task would uselessly handle all possible senses of "jaguar" thus slowing down the process and making it more cumbersome.

[5] `http://lucene.apache.org`
[6] `http://webgraph.dsi.unimi.it`

On the other hand, it might be the case that $lp(a_1) > lp(a_2)$. Now since $freq(a_1) \geq freq(a_2)$, this may occur because $link(a_1) \gg link(a_2)$. This is the case when a_2 adds a non-meaningful word to a_1 that nonetheless identifies some senses. As an example, consider a_1 = "act" and a_2 = "the act" for which it is $lp(a_1) > lp(a_2)$: in fact, "act" refers to a huge amount of possible senses (Act of parliament, Australian Capital Territory, Act of a drama, Group Action, etc. etc.), while "the act" is the name of a band and the title of a musical with a consequent small number of link occurrences. In this case we keep both anchors because, at this initial step of the annotation process, we are not able to make a principled pruning.

4.3 Anchor disambiguation

This phase takes inspiration from the algorithmic principles and the relatedness function deployed in previous works [11, 14, 15], nonetheless we specialize and/or enrich them to work in our context of short text fragments and to achieve *on-the-fly* annotation. More precisely, as in [11], we aim for the collective agreement among all anchors detected in the (short) input text, and thus deploy among their associated senses/pages the relatedness function proposed in [14]; moreover, as in [15], we deploy the context-anchors (if any) to boost the relatedness among the detected un-ambiguous senses. However, unlike these two approaches, we propose *new disambiguation scoring-functions* that are fast to be computed and take into account the sparseness of the anchors which is typical in the short input texts. Although simple, our scoring functions allow to improve precision-and-recall of the known approaches on short text fragments and, surprisingly, obtain competitive performance on the long ones (see Sect. 5.5).

Given a set of anchors $\mathcal{A}_T$, detected in the short input fragment T, TAGME tries to disambiguate each anchor $a \in \mathcal{A}_T$ by computing a score for each possible sense p_a of a (hence $p_a \in P(a)$) that is based on a *new notion* of "collective agreement" between p_a and the possible senses of all other anchors detected in T. To do this, we introduce a *voting scheme* that computes for each other anchor $b \in \mathcal{A}_T \setminus \{a\}$ its *vote* to the annotation $a \mapsto p_a$. Given that b may have many senses (i.e. $|P(b)| > 1$) we will compute this vote as the *average relatedness* between each sense p_b of b and the sense p_a we wish to associate to a. Since not all possible senses of b have the same *(statistical) significance*, we weight the contribution of p_b in this voting scheme by means of its commonness (i.e. $\Pr(p_b|b)$). The formula is:

$$vote_b(p_a) = \frac{\sum_{p_b \in P(b)} rel(p_b, p_a) \cdot \Pr(p_b|b)}{|P(b)|}$$

We notice that if b is un-ambiguous, it is $\Pr(p_b|b) = 1$ and $|P(b)| = 1$, so we have $vote_b(p_a) = rel(p_b, p_a)$ and hence we fully deploy the unique senses of the un-ambiguous anchors (as it occurred in [15]). But if b is polysemous, only the senses p_b related to p_a will mainly affect $vote_b(p_a)$ because of the use of the relatedness score $rel(p_b, p_a)$. This is the *key difference* with the scoring proposed by Milne&Witten, based only on un-ambiguous anchors (here possibly missing) and by Cucerzan, which does not use any weighting and thus results contaminated by un-meaningful senses (see Sect. 3). The net result is a significant improvement in Precision/Recall bounds, as shown in Sect. 5.5. As for [11], we do not use vectors over terms and for all involved senses (pages) with consequently slowdown in efficiency, but we implicitly used few and short vectors, one per anchor and one-dimension per sense (possibly pruned, see below).

Finally, the score for the annotation $a \mapsto p_a$ is computed as the sum of the votes given by all other anchors b detected in T: $rel_a(p_a) = \sum_{b \in \mathcal{A}_T \setminus \{a\}} vote_b(p_a)$. We use this score in combination with the commonness $\Pr(p_a|a)$ to disambiguate the anchor a and thus select the *best annotation* $a \mapsto p_a$. To do this we developed two different ranking algorithms: Disambiguation by Classifier (shortly DC) and Disambiguation by Threshold (shortly DT). DC uses a classifier that takes into account these two values as features and computes a *"probability of correct-disambiguation"* for all senses $p_a \in P(a)$. Among all $p_a \in P(a)$, DC selects the one reporting the highest classification score. On the other hand, DT recognizes a roughness in the value of rel_a among all $p \in P(a)$, so it computes the best sense p_{best} that

achives the highest relatedness value with a and then identifies the set of other senses in $P(a)$ that yield about the same value of $rel_a(p_{best})$, according to some fixed threshold ϵ. Finally DT annotates a with the sense that obtains the *highest commonness* among these *top*-ϵ senses.

Given that speed is a main concern in our context, both DC and DT discard from the above computation all senses whose prior-probability is lower than a properly set threshold τ. In fact, as illustrated in [15], the distribution of $\Pr(p|a)$ follows a power law so we can safely discard pages at the tail of that distribution. The setting of τ clearly affects the precision of the disambiguation process: if τ is too large, precision decreases because we would discard many valid senses; if τ is too small, speed and recall decrease. In Sect. 5.5 we will perform a wide set of experiments to evaluate these two algorithms and their parameter settings.

A comment is in order at this point. Previous works [4, 11, 13] deployed also the text surrounding anchors for boosting the efficacy of the disambiguation process. We tested these features in the design of TAGME but we either got worse accuracy or slower speed of annotation. For the lack of space we cannot report these numbers, but we foresee to investigate further the use of these terms (see Sect. 6).

4.4 Anchor pruning

The disambiguation phase produces a set $\mathcal{M}(\mathcal{A}_T)$ of candidate assignments, one per anchor detected in the input text T. This set has to be *pruned* in order to discard the possible un-meaningful anchors a. These "bad anchors" are detected via a *novel and efficiently-computable* scoring function that takes into account only two features: the link probability $lp(a)$ of the anchor a and the *coherence* of its candidate annotation $a \mapsto p_a$ (assigned by the Disambiguation Phase) with respect to the candidate annotations for the other anchors in $\mathcal{M}(\mathcal{A}_T)$. The link probability is a simple yet effective feature for detecting significant anchors, as it was shown in [15]. The coherence with the *un*-ambiguous anchors was also shown to be effective in [15], TAGME extends this notion to all anchors present in T and computes it as the average relatedness between the candidate sense p_a for a and the candidate senses p_b for all other anchors b. More precisely, let us define S as the set of distinct senses assigned to the anchors of T after the Disambiguation Phase, we compute:

$$coherence(a \mapsto p_a) = \frac{1}{|S|-1} \sum_{p_b \in S \setminus \{p_a\}} rel(p_b, p_a)$$

So the principle is to keep all anchors whose link probability is high or whose assigned sense (page) is coherent with the senses (pages) assigned to the other anchors. We implemented this idea by combining lp and *coherence* in one unique score, denoted by $\rho(a \mapsto p)$, obtained in two simple ways: either we average the two scores as $\rho_{\texttt{AVG}}(a \mapsto p_a) = (lp(a) + coherence(a \mapsto p_a))/2$; or we consider a linear combination of the form $\rho_{\texttt{LR}}(a \mapsto p_a) = \alpha \cdot lp(a) + \beta \cdot coherence(a \mapsto p_a) + \gamma)$, where the 3 parameters are trained via linear regression.

We also used some classifiers— such as C4.5, Bagged C4.5, Support Vector Machine— in order to provide a "probability of not-pruning" for an candidate annotation $a \mapsto p$. We finally set a threshold ρ_{NA} so that if $\rho(a \mapsto p) < \rho_{\text{NA}}$ then we remove the annotation for a setting $a \mapsto \text{NA}$. The parameter ρ_{NA} allows to balance recall vs precision, and its impact will be experimentally evaluated in Sect.5.6 where we show that, although much simple in using just two features, our pruning is effective and overcomes the other (more complicated) state-of-the-art annotators on short texts, still resulting competitive (if not superior) on long texts.

5 Evaluation

In the following subsections we will address some key questions that pertain with the efficiency and efficacy of TAGME:

- How much is the coverage of Wikipedia's anchors in short text fragments like the ones occurring in web-search snippets and tweets ? This is crucial to understand how much useful can be the usage of Wikipedia anchors as entity catalog. (See Sect. 5.1.)

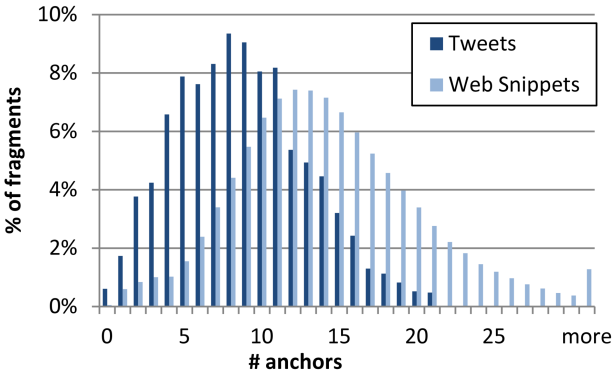

Figure 1: Number of Wikipedia anchors found in web snippets and tweets.

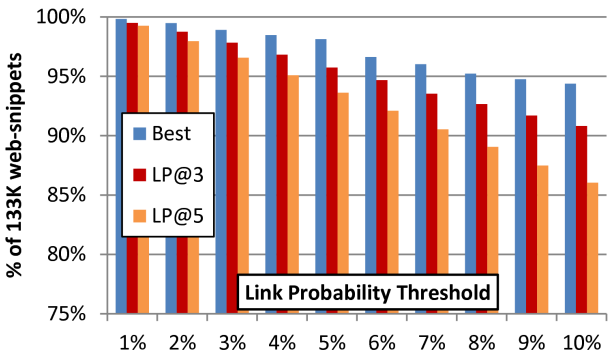

Figure 2: Distribution of Wikipedia anchors in web-snippets by considering their link probability.

- How do best know annotators (such as [11, 15]) behave on short text fragments ? (See Sect. 5.4.)
- How much effective is the disambiguation (Sect. 5.5) and pruning phases (Sect. 5.6) of TAGME with respect to these best-known systems on short and long texts?
- Is TAGME fast in annotating short text fragments, and how its speed compares with the other known systems? (See Sect 5.7.)

5.1 Coverage by Wikipedia anchors

First we want to evaluate the coverage of Wikipedia as entity catalog in the context of short fragments of text drawn from the web. We consider two types of text fragments: web snippets and micro-blogging (namely, tweets), which constitute a *worst-case setting* for an annotator because of their poor and much short textual composition. We derived these datasets by parsing about 5K tweets (of 10 terms on average) and about 133K web snippets (of 20 terms on average)[7].

[7]Tweets and web snippets are gathered by using the "The 1000 most frequent web search queries issued to Yahoo! Search". We randomly selected 300 queries from this dataset, performed searches on Tweeter and collected first 20 results. For web snippets, we used almost all queries in that dataset and we collected top 200 results for each performed query from a web search engine.

Figure 1 reports statistics on the number of Wikipedia anchors found in those fragments: notice that more than 93.9% tweets and 98.5% web-snippets have at least 3 anchors. So Wikipedia offers an unexpected large coverage of senses even in these challenging scenarios.

However this is just a quantitative test in that the presence of an anchor is not a witness of its "significance". So Figure 2 uses the link probability of an anchor as an estimate for its meaningfulness, as suggested in [13, 15], and plots the distribution of anchor's link-probability among web-snippets (results for Tweets are slightly lower, and not reported for the lack of space). In particular, the Figure plots for each lp-threshold the percentage of web-snippets that have the top-lp above that threshold (*Best*), the average among the top-3-lps above that threshold (*LP@3*), and the average among the top-5-lps above that threshold (*LP@5*). We notice that for a link-probability larger than 6.5%, which [15] considered a strong indication of a significant anchor, we get *Best* ¿ 95%, and *LP@3/5* ¿ 90%. (For the tweets we get, *Best*= 96%, $LP@3 = 89\%$, $LP@5 = 83\%$.) These results support our hypothesis that Wikipedia is a significant entity catalog also for short text fragments drawn from the Web.

5.2 Evaluation Measures

In order to evaluate the performance of the Disambiguation Phase in TAGME, we use standard precision and recall scores: $precision = tp/(tp + fp)$ and $recall = tp/(tp + fn)$, where tp, fp and fn are the number of true-positive, false-positive and false-negative cases.

In order to evaluate both Disambiguation and Pruning Phases, hence the whole TAGME system, we follow [11] and thus focus on the precision/recall measures that are computed only on the set of anchors which are annotated in the ground truth (see Sect. 5.3 for a description of the corpora). We adopt the following notation: $A \rightarrow A$ is the set of anchors annotated in the ground truth that were correctly annotated by TAGME, $A \nrightarrow A$ is the set of anchors annotated in the ground truth but wrongly annotated by TAGME, $A \rightarrow \text{NA}$ is the set of anchors annotated in the ground truth but pruned by TAGME (i.e. annotated with NA), $\text{NA} \rightarrow A$ is the set of anchors that were *not* annotated in the ground truth but were annotated by TAGME. Given this notation, we can compute the precision of the annotation and the recall of the annotation as follows:

$$P_{ann} = \frac{|A \rightarrow A|}{|A \rightarrow A| + |A \nrightarrow A| + |\text{NA} \rightarrow A|}$$

$$R_{ann} = \frac{|A \rightarrow A|}{|A \rightarrow A| + |A \nrightarrow A| + |A \rightarrow \text{NA}|}$$

Note that the denominator of P_{ann} denotes the number of anchors that got annotated (correctly or wrongly) by TAGME, whereas the denominator of R_{ann} denotes the number of anchors that were annotated in the ground truth.

The previous evaluation-measures are much demanding because they ask for a *perfect match* between the annotation in the ground truth and the annotation obtained by TAGME. If the goal of an annotator is to identify *topics* in the text fragment, then it doesn't matter which anchors got annotated, but which senses (pages of Wikipedia) annotated that text. So, given a text fragment f, let $\mathcal{T}(f)$ be the set of senses pointed to by f's anchors in the ground truth, and $\text{TAGME}(f)$ be the set of senses identified by TAGME in f. So as in [15], we define a topic-based notion of precision (P_{topics}) and recall (R_{topics}) over a set of fragments $\mathcal{F}$, as follows:

$$P_{topics} = \frac{\sum_{f \in \mathcal{F}} |\mathcal{T}(f) \cap \text{TAGME}(f)|}{\sum_{f \in \mathcal{F}} |\text{TAGME}(f)|}$$

$$R_{topics} = \frac{\sum_{f \in \mathcal{F}} |\mathcal{T}(f) \cap \text{TAGME}(f)|}{\sum_{f \in \mathcal{F}} |\mathcal{T}(f)|}$$

5.3 Evaluation corpora

In our experiments we considered three datasets. The first one is derived from the manually annotated dataset introduced in [11], called IITB dataset. It consists of about 100 documents and 19K anchors (40% of them are annotated with NA). For the experiments we split each IITB document into fragments of 30 words each: this way on avg each short text contains 20 non-*stop-words*, as it occurs for web-snippets.

The second and third datasets were derived from Wikipedia, as done in [15], and will be used for evaluating the disambiguation phase (WIKI-DISAMB30) and the overall system (WIKI-ANNOT30). The former dataset consists of 1.4M short fragments randomly selected from Wikipedia pages. Each fragment consists of about 30 words. To avoid any advantage to TAGME, we were careful in selecting fragments that surely contain at least one ambiguous anchor-text (i.e. $|P(a)| > 1$). The latter dataset consists of 150K fragments whose set of anchors is expanded by detecting all their un-annotated anchors (say $\mathcal{N}$) and annotating them with one of the senses (say $\mathcal{T}$) pointed out by (possibly other) anchors that occur in the same page from which that fragment was drawn. We did this because Wikipedia-contributors usually link only the first occurrence of an anchor-text in a page, so if the short fragment contains a subsequent occurrence of that anchor, this occurrence could be not annotated in the ground truth. Therefore we expand WIKI-ANNOT30 by annotating an anchor $a \in \mathcal{N}$ with the page p that has the largest commonness $\Pr(p|a)$ among the ones in $P(a) \cap \mathcal{T}$ (if $\neq \emptyset$). After this expansion, WIKI-ANNOT30 contains about 1.5M annotated anchors, 63% of them annotated by NA (i.e. not annotated). So, each of the 150K fragments has on average 10 anchors, about 4 of them got annotated.

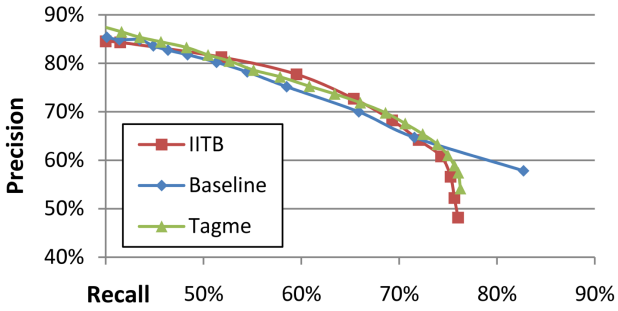

Figure 3: Performance of Baseline, IITB's system and TAGME over the IITB dataset.

We point out that we need two distinct datasets for the two phases (i.e. Disambiguation and Pruning) because in the former phase we need to concentrate on ambiguous anchors and thus force the dataset to contain them; in the other phase, fragment may or may not contain ambiguous anchors and also we wish to further stress TAGME providing in input a different dataset, thus further testing the Disambiguation Phase (which was trained on WIKI-DISAMB30).

5.4 Baseline and competitors

We developed a trivial (yet surprisingly effective) baseline system that performs the disambiguation of anchor a by selecting its *most common sense*, namely the page $p \in P(a)$ that has the highest commonness $\Pr(p|a)$. This system, inspired by WIKIFY's pruner [13], then implements the subsequent pruning phase via a threshold on the link probability of anchor a. It goes without saying that Precision and Recall of this Baseline can be balanced by changing this threshold.

Figure 3 compares the performance of Baseline, the system in [11], and our TAGME over the IITB dataset chopped into short texts (see above), using *annotation-based* precision and recall. Since we could not get access to IITB's system,[8] we extracted its performance from Figure 14 in [11]. Note that this extrapolation gives advantage to [11]'s system wrt TAGME because they used the entire document whereas TAGME processes it one short fragment at a time. Overall Figure 3 is interesting because it shows that Baseline is very close or, even, superior to [11] at the extremes of the Recall-range; on the other hand, TAGME is always above the Baseline up to a recall of 75%, and it is competitive (if not superior!) to IITB's system even if TAGME operates on short fragments.

We do not want to draw any conclusions upon the elegant algorithm in [11], actually we foresee to download their software if it'll be made available in the future[8] or re-implement their scoring functions upon TAGME's architecture. Moreover, since the IITB dataset awards too much the most-common sense, as one can argue given the excellent performance of Baseline, we decided to drop it from the following experiments on short texts, and hence concentrate our following experiments on the two datasets WIKI-DISAMB30 and WIKI-ANNOT30. For the tested systems, we will consider the Baseline and Milne&Witten's system [15] because it is freely available[9] and offers competitive performance in annotating *long* documents (recall that no known software is designed to annotate short text fragments, which is properly the specialty of TAGME).

5.5 Evaluation of the disambiguation phase

We split WIKI-DISAMB30 into two datasets: one contains 400K anchors and is used for training, the other contains the remaining 1M anchors. We refer the reader to Sect. 4.3 for details on our two approaches to disambiguation, here we comment their experimental results.

Approach DC. We trained a C4.5 classifier that was shown to achieve the best results for disambiguation in [15]. We trained it using different values of τ, and discovered that $\tau = 0.5\%$ got the best results: larger values didn't gain precision while recall decreased significantly.

Approach DT. Recall that this approach depends on two parameters: τ and ϵ. Setting ϵ close to zero leads DT to always select the sense (page) that achieves the highest value of rel_a (i.e. the most related sense),

[8] Chakrabarti's personal communication.

[9] `http://wikipedia-miner.sourceforge.net` We set this system with the same snapshot of English Wikipedia used by our TAGME.

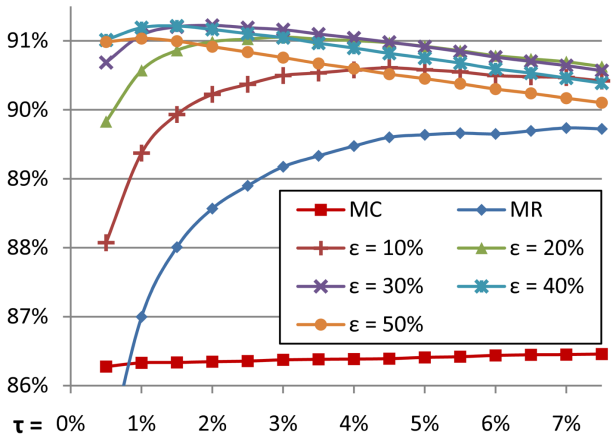

Figure 4: Performance of DT with respect to ϵ and τ over the training dataset, where `MC` and `MR` plot the performance for the choice of the Most Common (i.e. $\epsilon = 100\%$) and the Most Related (i.e. $\epsilon = 0\%$) sense, respectively.

while ϵ close to 1 leads DT to select the sense with the highest commonness (i.e. the most common sense). It could be the case that two senses (pages) yield about the same values of rel_a and commonness (in our experiments this occurs for 0.2% of the disambiguated anchors): in this case DT chooses the sense at random. Also, if all senses get $rel_a = 0$, we decided to set $a \mapsto$ NA: this gained just 0.04% in Precision and a drop of 0.6% in Recall, but doing this, disambiguator is more consistent when used in the whole system, because it can help the following pruning phase (see Sect. 5.6). Figure 4 plots F-Measure with respect to τ and ϵ. To clarify it, we didn't plot the performance of DT for values of $\epsilon > 50\%$: because for these values its performance approaches the *most common* scheme (*MC*). Overall, lower values of ϵ are better; experiments lead us to choose $\tau = 2\%$ and $\epsilon = 30\%$.

Given these parameter settings for DC and DT, we compared them against Milne&Witten's disambiguator over the 1M anchors of Wiki-Disamb30. To make the comparison wider, we consider also two other (simple) approaches to disambiguation: one selects always the most-common sense from $P(a)$ (i.e. statistically-driven choice), and the other randomly selects a page from $P(a)$ (i.e. oblivious choice).

	Precision	Recall	F-Measure
Random	32.2	32.2	32.2
Most Common	85.8	86.8	86.3
Milne&Witten	**92.3**	84.6	88.3
DC	91.7	89.9	90.8
DT	91.5	**90.9**	**91.2**

Table 1: Performance of disambiguation algorithms over Wiki-Disamb30.

Performances are shown in Table 1. The F-measure of both our algorithms is (significantly) better than all other approaches; this is also true for recall, whereas precision of the Milne&Witten's system is slightly better than ours (a difference lower than 0.6%). The key difference between Tagme and Milne&Witten's system is that we are deploying relatedness not only with the senses of un-ambiguous anchors, but also with all the other candidate senses in the fragment, via our novel voting scheme described in Sect. 4.3. This is crucial in our scenario of application because the input texts are short and thus often do not contain un-ambiguous anchors.

As for the comparison between DC and DT, although precision and recall are very close, we decided to choose DT as winning approach because of three main reasons: (i) it has a better F-Measure, (ii) the best F-Measure is obtained with $\tau = 2\%$ that let us to gain much speed, as explained in Sect. 4.3, (iii) DT depends on a threshold ϵ that gives much flexibility: we can increase ϵ if the input texts are too ambiguous (and thus choose more often the most-common sense) or decrease ϵ if the input texts are longer and more focused (and thus choose more often the most-related sense).

As a final check for fairness, let us compare the results obtained on our datasets against the ones achieved by Milne&Witten in [15]. In that paper, the authors evaluated their system over a collection of 100 full-articles of Wikipedia, each containing at least 50 links (for a total amount of 11,000 anchors). Their system yield an overall performance on disam-

biguation of about 96%, which is larger than the 88% we obtain for their system on our datasets. The reason is that our datasets are more difficult to be disambiguated because texts are short and more ambiguous as witnessed by the following numbers: on our datasets the choice of a random sense gets $F \approx 32\%$ and the choice of the most-common sense gets $F \approx 86\%$; whereas on Milne&Witten's dataset, these numbers were 53% and 90% respectively. This remarks further that the performance yield by TAGME in the disambiguation phase is much effective, with the *plus* of being computed fast!

5.6 Evaluation of the whole system

As detailed in Sect. 4.4, the pruning step hinged onto two features, *lp* and *coherence*. We tested two combination of these features (AVG and LR) and three different classifiers deploying these features (C4.5, Bagged C4.5 and Support Vector Machine). We trained these classifiers over 50K short-texts extracted from WIKI-ANNOT30. We consider each anchor contained in these fragments as a test case: if it is linked in the ground truth it is a positive case, otherwise it is a negative case, for a total amount of 500K test cases. However, since the result of the disambiguation phase affects the value of *coherence* (given that it defines S in the formula of Sect. 4.4) a wrongly disambiguated anchor could be misleading in the training step because it provides a positive case for the classifier but its *coherence* value would be a negative example. So we removed these disambiguation errors from the training. Moreover, to train the three parameters of the approach based on linear regression, we transformed the boolean class of the ground truth (linked or not linked) into a numeric value: we set $\rho_{\texttt{LR}} = 1$ for positive cases (i.e. linked anchors) and $\rho_{\texttt{LR}} = 0$ for negative cases.

After training, we evaluated our approaches over the remaining 100K fragments of WIKI-ANNOT30, obviously without removing the wrongly disambiguated anchors. We performed several evaluations for different values of the parameter ρ_{NA} which controls the *sensibility* of our annotation process. So to gain insights on its impact, we evaluated all approaches by varying ρ_{NA} in $[0, 1]$ using a step of 0.01.

	P_{ann}	R_{ann}	F-Measure
Milne&Witten	69.32	69.52	69.42
Baseline	74.98	69.94	72.37
Only *lp*	75.5	72.01	73.71
AVG	76.27	76.08	76.17
LR	76.49	75.74	76.1
C4.5	**76.72**	**76.22**	**76.47**
Bagged C4.5	76.54	**76.22**	76.38
SVM	76.25	75.96	76.11
	P_{topics}	R_{topics}	F-Measure
Milne&Witten	69.6	69.8	69.7
Baseline	77.16	74.09	75.59
Only *lp*	76.85	76.65	76.75
AVG	78.41	77.48	77.94
LR	78.42	77.03	77.72
C4.5	76.78	**79.69**	**78.21**
Bagged C4.5	**79.13**	77.12	78.11
SVM	78.91	77.13	78.01

Table 2: Performance of annotators over WIKI-ANNOT30, using either *annotation* or *topics* metrics.

In these experiments we included another simple method that we called "Only *lp*": it disambiguates the anchors by using DT and performs anchor pruning by using only the link probability, like Baseline explained above (see Sect. 5.4). By comparing this approach against TAGME, we can evaluate the significance of using the feature *coherence* in addition to the link-probability in order to perform the pruning step.[10]

Table 2 summarizes all experimental results by reporting the numbers only for the setting of ρ_{NA} that yield the highest F-Measure, using 2-fold cross validation. As we could expect, *annotation* measures are more severe than *topics* measures. Anyway, there are dependencies between them, and indeed the ranking of the experimented systems is the same for both of them.

The system by Milne&Witten performed poorly, overcame even by Baseline. This is worse than ex-

[10]We tested also the case of *coherence-only* feature but performance was worse than Milne&Witten's one. They are not reported to ease the reading of the Table.

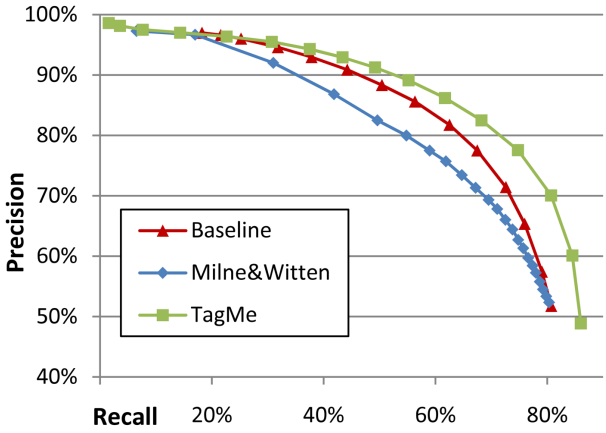

Figure 5: Performance of annotators by varying the value of ρ_{NA}.

pected, but not surprising, because many features used by their pruning method are not effective when dealing with short fragments of text: indeed, they considered features like location and frequency of anchors (which may be "undefined" or even misleading on short texts), as well as they considered only the un-ambiguous anchors to compute a coherence-score (and these are often absent in short texts, as we commented above).

The overall performance of all our pruning approaches is very close to each other, and all of them overcome the Baseline quite clearly. Results also show that "Only *lp*" is surpassed by all other approaches that deploy *coherence*, which confirm the significance of this feature in the pruning phase. As a result, we decided to implement in TAGME the simplest pruning method based on $\rho_{\texttt{AVG}}$. This is because SVM was very slow, the others were as fast as AVG but they needed a training step that we prefer to avoid in the Web context. In some sense our choice of the simple AVG-pruning was driven by the *Occam Razor* principle!

Finally Figure 5 reports the comparison among the three systems— TAGME, in which we set $\tau = 2\%$ and $\epsilon = 30\%$, Milne&Witten's system and the Baseline. We notice that the performance of all annotators has a uniform trend as ρ_{NA} varies in $[0, 1]$, and TAGME overcomes the other approaches even by pushing the value of ρ_{NA} to its boundaries. The value of ρ_{NA} that gives the highest F-measure is 0.2, anyway ρ_{NA} can be set properly in order to balance precision vs recall.[11]

5.7 Last issues

How does TAGME *perform on long text?* In the context of long text, TAGME operates by shifting a text window of about 10 anchors over the long text in input: this way we do not change the software and system scales linearly with the number of anchors in the input text (see below). It is clear that this approach gives advantage to both Chakrabarti's and Witten&Milne's systems in terms of precision/recall of the annotation, because they deploy the full input text (and thus probably more than 10 anchors!). Nevertheless, we decided to stick on this unfavorable setting for TAGME in order to stress its performance and keep fast its annotation speed. Figure 3 already showed that TAGME over the IITB dataset is competitive wrt Chakrabarti's annotator with the *plus* of being one-order of magnitude faster. Moreover, Table 2 and the way we built WIKI-ANNOT30 allow us to extrapolate a comparison between TAGME and the Milne&Witten's system over full Wikipedia articles, and safely conclude that TAGME's F-Measure can be estimated at about 78% (for P_{topics} and R_{topics}) which surpasses the 74% reported in [15] for long and highly linked Wikipedia full-documents (whose structure is much better than our WIKI-ANNOT30, as we commented above)!

Since last experiments are based over datasets drawn from Wikipedia, does TAGME *achieve the same effective performance* in the wild*?* There are two issues that let us argue positively about this: (1) the IITB dataset (see Sect. 5.4) is a manually annotated set of news stories drawn from web, and there TAGME is competitive, if not superior, to the state-of-the-art system proposed in [11]; (2) the user-study conducted in [15] confirmed that perfomance yielded over large datasets drawn from Wikipedia are good predictors of performance *in the wild*. In addition to these two positive witnesses, we are currently setting up a much

[11] The on-line version of TAGME offers this feature to the user.

larger user-study over Mechanical Turk[12] in order to yield a further feedback on the efficiency and effectiveness of our approach.

What about time efficiency of TAGME*?* We notice that the most time consuming step is the calculation of the relatedness score, because anchor detection and other scores require time *linear* in the length of the input text T. If n is the number of anchors detected in T, s is the average number of senses potentially associated with each anchor, and d_{in} is the average in-degree of a Wikipedia page, then the time complexity of the overall calculation is $O(d_{in} \times (n \times s)^2)$. In practice, for our short text fragments (taken from the Web, see Sect. 5.1) it is $n \approx 10$, $s \approx 5$ and $d_{in} \approx 50$, so that our current implementation of TAGME, although not much engineered, takes less than 10ms per input text fragment on a commodity PC[13]. This is more than an order of magnitude faster than the time performance reported by [11] for about 15 anchors. Additionally, when TAGME is applied on long texts of L anchors ($L \gg 10$), it can process them by considering overlapping windows of about w anchors each (e.g. $w = 10$). This way, when shifting the window over text anchors to the right, we can re-compute incrementally the scores, so actually paying $W_{cost} = O(d_{in} \times w \times s^2)$ time per shift. This is $O(L \times W_{cost})$ in total and thus linear with number of anchors in the input text. Conversely, [11]'s system scales "mildly quadratically" in L.

6 Conclusion and future works

Our datasets are freely available as well as is available TAGME as a web-service [14]. We believe that TAGME, like the systems of [11, 15], has implications which go far beyond the enrichment of a text with explanatory links. Currently, we are investigating TAGME's annotation of short texts in the on-the-fly labeled clustering of search-engine results of our past system SNAKET[5], which was based just on syntactic features (as most of its similar competitors, see e.g. [2, 16]). Furthermore, we are studying how other by-products of Wikipedia— such as `DBpedia.org`, `Freebase.com`, Kylin [20] or YAGO [18]— could be used to better relate concepts and/or assigning senses to text spots. Finally, as commented in Sect. 5.6, we are setting up a large user-study based on Mechanical Turk with the goal of extending our experimental results and thus check TAGME *in the wild*.

We conclude by mentioning another quote from the talk given at WSDM 2010 by S. Chakrabarti: "How can they [users] take advantage of the new type-entity-snippet composite data model?". Well, we have not yet a principled answer to this question, but we think that our paper has added a new info pathway between the *micro-docs of Web 2.0* and the semi-structured knowledge provided by Wikipedia (another Web 2.0 product!).

[12] `http://www.mturk.com`

[13] In detail, anchor parsing takes about 3.5ms, disambiguation and pruning about 6.5ms per fragment. Of course algorithm engineering could speed-up it, and this will be addressed in the future.

[14] `http://tagme.di.unipi.it`

References

[1] S. Banerjee, K. Ramanathan, and A. Gupta. Clustering short texts using wikipedia. In *Proc. ACM SIGIR*, 787–788, 2007.

[2] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):1–38, 2009.

[3] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2002.

[4] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. *Proc. of Empirical Methods in NLP*, 2007.

[5] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Proc. WWW*, 801–810, 2005.

[6] E. Gabrilovich and S. Markovitch. Feature generation for text categorization using world knowledge. In *Proc. IJCAI*, 1048–1053, 2005.

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

[8] R. V. Guha and R. McCool. Tap: A semantic web test-bed. *J. Web Sem.*, 1(1):81–87, 2003.

[9] J. Hu, L. Fang, Y. Cao, H.-J. Zeng, H. Li, Q. Yang, and Z. Chen. Enhancing text clustering by leveraging wikipedia semantics. In *Proc. ACM SIGIR*, 179–186, 2008.

[10] X. Hu, N. Sun, C. Zhang, and T.-S. Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proc. ACM CIKM*, 919–928, 2009.

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

[12] O. Medelyan, D. Milne, C. Legg, and I. H. Witten. Mining meaning from wikipedia. *Int. J. Hum.-Comput. Stud.*, 67(9):716–754, 2009.

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

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

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

[16] S. Osinski. Improving quality of search results clustering with approximate matrix factorisations. In *Proc. ECIR*, LNCS vol. 3936, 167–178, 2006.

[17] M. Sahami and T. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proc. WWW*, 377–386, 2006.

[18] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge. In *Proc. WWW*, 697–706, 2007.

[19] C. Whitelaw, A. Kehlenbeck, N. Petrovic, and L. Ungar. Web-scale named entity recognition. In *Proc. ACM CIKM*, 123–132, 2008.

[20] F. Wu and D. Weld. Automatically Semantifying Wikipedia. In *Proc. ACM CIKM*, 41–50, 2007

[21] Y. Yang, N. Bansal, W. Dakka, P. Ipeirotis, N. Koudas, and D. Papadias. Query by document. In *Proc. ACM WSDM*, 34–43, 2009.

[22] H. Zaragoza, J. Atserias, M. Ciaramita, and G. Attardi. Semantically annotated snapshot of the English Wikipedia. `http://www.yr-bcn.es/semanticWikipedia`, 2007.