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

2 authors:

Paolo Ferragina
University of Pisa
230 PUBLICATIONS **10,696** CITATIONS

SEE PROFILE

Ugo Scaiella
University of Pisa
9 PUBLICATIONS **1,285** CITATIONS

SEE PROFILE

TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities)[*]

Paolo Ferragina
Dipartimento di Informatica
University of Pisa, Italy
ferragina@di.unipi.it

Ugo Scaiella
Dipartimento di Informatica
University of Pisa, Italy
scaiella@di.unipi.it

ABSTRACT

We designed and implemented TAGME, a system that is able to efficiently and judiciously augment a plain-text with pertinent hyperlinks to Wikipedia pages. The specialty of TAGME with respect to known systems [5, 8] is that it may annotate texts which are short and poorly composed, such as snippets of search-engine results, tweets, news, etc.. This annotation is extremely informative, so any task that is currently addressed using the bag-of-words paradigm could benefit from using this annotation to draw upon (the millions of) Wikipedia pages and their inter-relations.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*text analysis*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms

Algorithms, Experimentation, Performance.

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. In recent years, a good deal of work attempted to go beyond this paradigm with the goal of improving the search experience on (un-)structured or semi-structured textual data. The key idea is to identify a sequence of terms (also called *spots*) in the input text and to annotate 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 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`, `TAP`), and a large text collection with wide coverage but unstructured and noised content (like the whole Web). This annotation has implications that go beyond the enrichment of texts with explanatory links because it provides structured knowledge about any unstructured fragment of text. So any task that is currently addressed with bags of words-indexing could use these techniques to draw on a vast network of concepts and their inter-relations.

To our knowledge the first work that addressed the problem of linking spots to Wikipedia pages was WIKIFY [6], followed by [2]. Recently Milne and Witten [8] proposed an approach that yielded considerable improvements by hinging on three main ingredients: (i) the identification in the input text of a set C of so-called *context pages*, namely pages linked by spots 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 with the other context pages in C. These ingredients allowed [8] to achieve an F-Measure of 74.8% over long and highly-linked Wikipedia full-articles. Last year, Chakrabarti et al [5] further improved [8] by introducing two main novelties. The first one was to score an annotation with two terms: one *local* to the occurrence of the spot and the other *global* to the text fragment. The second novelty was to model the annotation process as a search for the mapping that maximizes the sum of the (local+global) scores of the selected annotations, via the solution of a quadratic assignment problem. Experiments over a manually annotated dataset showed that Chakrabarti's approach yields a precision comparable to Milne&Witten's system but with a higher recall.

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 snippets of search-engine results, or the tweets of a Twitter channel, or the items of a news feed, 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 that are rather available when texts are long. The systems of [5, 8] are designed to deal with rea-

sonably long texts, so they seem unsuitable in this context because either C is empty [8] or the annotation is slow [5].

Given these limitations we have designed and implemented TAGME a software system that, on-the-fly and with high precision/recall, annotates short fragments of text with pertinent hyperlinks to Wikipedia articles. TAGME uses Wikipedia *anchor texts* as spots and the *pages* linked to them in Wikipedia as their possible senses. TAGME solves ambiguity and polysemy in the potentially many available anchor-page mappings by finding the collective agreement among them via new scoring functions which are *fast* to be computed and *effective* in the produced annotation. TAGME obtains this by taking explicitly into account the sparseness of the anchors in the short input text via the proper combination of the relatedness function among concepts proposed in [7] and some other statistics drawn from Wikipedia. Preliminary experiments show that TAGME outperforms the best known systems [5, 8] when they are adapted to work on short texts, and surprisingly, it results competitive on long texts too. In particular TAGME yields an F-measure of about 78%, with the possibility to balance precision (up to 90%) vs recall (up to 80%), thus being always better than [5, 8]. The time complexity of TAGME's annotation is *linear* in the number of processed anchors (cfr. [5]'s quadratic time complexity), and indeed it can annotate texts in less than 2ms per anchor, so more than one order of magnitudes faster than [5]. TAGME is available at `http://tagme.di.unipi.it` and further details are reported in [4].

2. THE ANATOMY OF TAGME

Notation and terminology. A *(text) anchor* (referred as *spot*) for a Wikipedia page p is the text used in another Wikipedia page to point to p. Because of polysemy and variant names, we denote by $Pg(a)$ the set of all Wikipedia pages linked by a, and set $freq(a)$ as the number of times the text a occurs in Wikipedia (as an anchor or not); whereas we use $link(a)$ to denote the number of times the text a occurs as an anchor in Wikipedia (so $link(a) \leq freq(a)$). Also we use $lp(a) = link(a)/freq(a)$ to denote the *link*-probability that an occurrence of a has been set as an anchor; and use $\Pr(p|a)$ to denote the *prior*-probability (a.k.a. *commonness*) that an occurrence of an anchor a points to $p \in Pg(a)$.

The annotation of an anchor a with some page $p \in Pg(a)$ is denoted by $a \mapsto p$. Often a has more senses, thus $|Pg(a)| > 1$, so we call *disambiguation* the process of selecting one of the possible senses of a from $Pg(a)$. It goes without saying that not all occurrences of the spot a should be considered as text anchors, so we follow [5] and introduce a *fake page* NA that is used in the annotation $a \mapsto$ NA in order to denote that an occurrence of anchor a has been *pruned*.

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

Anchors were drawn from Wikipedia pages, plus we added the titles of redirect pages and some variants of the page-titles as suggested in [2]. From this set, we eventually removed the anchors composed by one character or just numbers, and discarded all anchors whose absolute or relative frequency is small enough to argue that they are unsuitable for annotation and probably misleading for disambiguation.

The final anchor dictionary contains about 3M anchors, and it is indexed by Lucene[1].

Senses were built by taking all Wikipedia pages, and then discarding disambiguation pages, list pages, and redirect pages. At the end remained about 2.7M pages (indexed by Lucene), with a link-graph of about 147M edges (indexed in internal-memory by Webgraph[2]).

Anchor Disambiguation. Let $\mathcal{A}_T$ be the set of all anchors occurring in an input text 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 $p_a \in Pg(a)$) via a new notion of "collective agreement" between p_a and the possible senses of all other anchors detected in T. To do this, TAGME deploys a new *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. $|Pg(b)| > 1$) we compute this vote as the average relatedness between each sense p_b of the anchor b and the sense p_a we wish to associate to the anchor a. Moreover, since not all possible senses of b have the same (statistical) significance, we weight the contribution of p_b by means of its commonness $\Pr(p_b|b)$. Hence the formula is $vote_b(p_a) = \frac{\sum_{p_b \in Pg(b)} rel(p_b, p_a) \cdot \Pr(p_b|b)}{|Pg(b)|}$. We notice that if b is un-ambiguous, it is $\Pr(p_b|b) = 1$ and $|Pg(b)| = 1$, 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 in [8]). 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 $rel(p_b, p_a)$. The total score $rel_a(p_a)$ for the "goodness" of the annotation $a \mapsto p_a$ is computed as the sum of the votes given to it by all other anchors b detected in the input text. This is the key difference with the scoring proposed by Milne&Witten, which was based on un-ambiguous anchors only (here possibly missing). As for [5], we do not use term-vectors for all involved senses, but we use (implicitly) few short vectors: one per anchor and with one-dimension per sense. This allows TAGME to be fast.

To disambiguate a, thus select its best annotation $a \mapsto p_a$, we designed and tested two ranking algorithms: Disambiguation by Classifier (DC) and Disambiguation by Threshold (DT). DC is based on a classifier that uses as features the score $rel_a(p_a)$ and the commonness $\Pr(p_a|a)$ to compute a "probability of correct-disambiguation" for all senses $p_a \in Pg(a)$. Among all $p_a \in Pg(a)$, DC selects the one reporting the highest classification score. On the other hand, DT recognizes a roughness in the value of $rel_a(p_a)$ among all $p_a \in Pg(a)$, so it computes the top-ϵ best senses p' in $Pg(a)$ according to their $rel_a(p')$, and then annotates a with the sense that obtains the highest commonness among them.

Since speed is a main concern, DC and DT discard from the above computation all senses $p \in Pg(a)$ whose prior-probability $Pr(p|a) < \tau$, where τ is properly set (see next).

Anchor Pruning. The set $\mathcal{M}(\mathcal{A}_T)$ of candidate assignments produced by the Disambiguation Phase 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 $a \mapsto p_a$ with respect to the candidate annotations of the other an-

---

[1] `http://lucene.apache.org`
[2] `http://webgraph.dsi.unimi.it`

chors in $\mathcal{M}(\mathcal{A}_T)$. The link probability is a simple and effective feature for detecting significant anchors [8]. The coherence with the *un*-ambiguous anchors was also shown to be effective in [8], TAGME extends this notion to all anchors present in the input text 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 in T. 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 in $\mathcal{M}(\mathcal{A}_T)$. We implemented and tested this idea in two ways (called AVG and LR), each of them produces a score $\rho(a \mapsto p)$ for a candidate annotation $a \mapsto p$ which is then compared with a threshold ρ_{NA} so that if it is lower then that annotation, then it is pruned by setting $a \mapsto \text{NA}$. The parameter ρ_{NA} allows to balance recall vs precision. The two approaches combine *lp* and *coherence* as follows: ρ_{AVG} averages the two scores; whereas ρ_{LR} uses a linear combination trained via linear regression.

3. EXPERIMENTS

In our experiments we considered three datasets. The first one is derived from the manually annotated dataset introduced in [5], 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, so to obtain a *short* input text (mimicking the snippet of a search-engine result). The second and third datasets were derived from Wikipedia, as done in [8], and have been used for evaluating the disambiguation phase (WIKI-DISAMB30) and the overall system (WIKI-ANNOT30). The former dataset consists of 1.4M short texts randomly selected from Wikipedia pages, and consisting of about 30 words. To avoid any advantage to TAGME, we selected fragments that surely contain at least one ambiguous anchor (i.e. $|Pg(a)| > 1$). The latter dataset consists of 150K fragments whose set of anchors is expanded by detecting all anchors that occur not annotated (say $\mathcal{U}$) and annotate them with one of the senses pointed out by (possibly other) anchors that occur in the same page from which that fragment was drawn (say $\mathcal{O}$). 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 $a \in \mathcal{U}$ with the page p that has the largest commonness $\Pr(p|a)$ among the ones in $Pg(a) \cap \mathcal{O}$ (if $\neq \emptyset$). These are the senses associated to a in the same page from which that fragment was drawn (typically one only). After this expansion, WIKI-ANNOT30 contains about 1.5M annotated anchors, 63% of them annotated by NA (i.e. not annotated).

To evaluate the performance of the Disambiguation Phase, we use standard precision and recall scores, whereas for the overall annotation process we follow [5] and thus focus on the precision P_{ann} and recall R_{ann} measures that are computed only on the set of anchors which are annotated in the ground truth (i.e. the corpora above). These last measures are much demanding because they ask for a *perfect match* between the annotation in the ground truth and the one obtained by the tested system. If the goal is to identify *topics* in the text fragment, then it doesn't matter which anchors got annotated but which senses got linked. So, let $\mathcal{G}(T)$ be the senses (pages) associated to the anchors of T in the ground

	Precision	Recall	F-Measure
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 three disambiguators over WIKI-DISAMB30.

	P_{ann}	R_{ann}	F-Measure
Milne&Witten	69.32	69.52	69.42
AVG	76.27	76.08	76.17
LR	76.49	75.74	76.10
	P_{topics}	R_{topics}	F-Measure
Milne&Witten	69.60	69.80	69.70
AVG	78.41	77.48	77.94
LR	78.42	77.03	77.72

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

truth, and let $\mathcal{S}(T)$ be the senses identified by the tested system over T. As in [8], we define a topic-based notion of precision (P_{topics}) and recall (R_{topics}) for the system to be evaluated over a set of fragments $\mathcal{F}$, as follows: $P_{topics} = \frac{\sum_{T \in \mathcal{F}} |\mathcal{G}(T) \cap \mathcal{S}(T)|}{\sum_{T \in \mathcal{F}} |\mathcal{S}(T)|}$ and $R_{topics} = \frac{\sum_{T \in \mathcal{F}} |\mathcal{G}(T) \cap \mathcal{S}(T)|}{\sum_{T \in \mathcal{F}} |\mathcal{G}(T)|}$.

Disambiguation Phase. We split WIKI-DISAMB30 into two datasets: one contains 400K anchors, the other contains the remaining 1M anchors. We used these datasets to train/test DC and DT, and compare them against Milne and Witten's disambiguator. Performances are shown in Table 1.

The recall of DC and DT are significantly better than the one of Milne&Witten's system, precision is slightly worse with a difference lower than 0.6%, so the overall F-measure of TAGME improves Milne&Witten's one by about 3%. 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. This is crucial in our scenario because the input texts are short and thus often do not contain un-ambiguous anchors. We plan to dig into the disambiguation's algorithm in order to further improve these figures.

As for the comparison between DC and DT, although precision and recall are very close, we decided to choose DT as disambiguator in TAGME because it has a better F-Measure, it is faster (with $\tau = 2\%$), and it is also more flexible because of the threshold ϵ: we can increase ϵ if the input text is ambiguous (and thus choose more often the most-common sense) or decrease ϵ if the input text is longer and more focused (and thus choose more often the most-related sense). At the end ϵ was chosen to be 30%.

The overall system: Disambiguation+Pruning. We recall that the pruning step hinges onto two features, *lp* and *coherence*. We trained LR over 50K short-texts extracted from WIKI-ANNOT30, and then evaluated it over the remaining 100K fragments. For both AVG and LR we tried various values for the parameter ρ_{NA} which controls the *sensibility* of our annotation process. Other experiments are reported in [4].
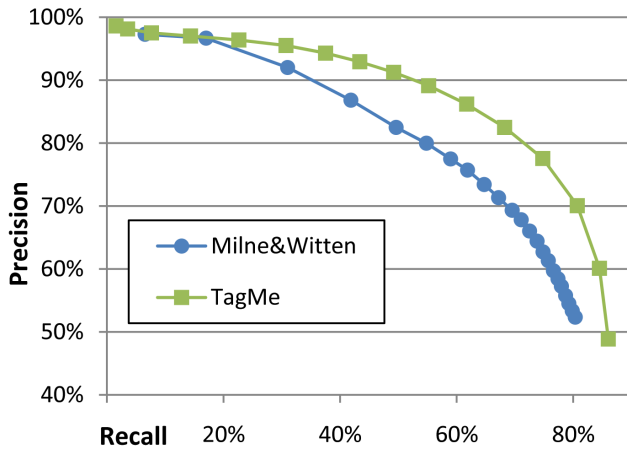
Figure 1: Performance of two annotators by varying the value of ρ_{NA}.

As expected, *annotation* measures are more severe than *topics* measures, although dependencies do exist. We decided to implement in TAGME the simplest method based on ρ_{AVG}, because it achieves a slightly better F-measure with no-need of a training step. The system by Milne&Witten performed poorly, because many features used by their pruning method are not effective when dealing with short texts. In fact they consider features like location and frequency of anchors (which may be "undefined" or even misleading on short texts), as well as they consider only the un-ambiguous anchors to compute a coherence-score (and these are often absent in short texts).

Finally Figure 1 reports the comparison over the dataset WIKI-ANNOT30 between the system of Milne&Witten and TAGME, in which we used ρ_{AVG} for pruning and set $\tau = 2\%$ and $\epsilon = 30\%$. Chakrabarti's system is not included in this comparison because it is un-available[3]. Looking at Figure 1 we notice that the performance of the two annotators has a uniform trend as ρ_{NA} varies in $[0, 1]$, and TAGME overcomes Milne&Witten's approach significantly over the entire range. Parameter ρ_{NA} can be set to balance precision vs recall, and indeed the on-line version of TAGME offers this feature to the user. In the future we plan to investigate other disambiguators/pruners in order to better investigate the impact of the various features and algorithmic choices in TAGME, and possibly draw some performance figures about Chakrabarti's approach.

The most time consuming step in TAGME 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 annotation process is $O(d_{in} \times (n \times s)^2)$. In practice these numbers are very small for short texts, so our current implementation of TAGME takes less than 2ms per anchor on a commodity PC. This is more than one order of magnitudes faster than the time performance reported by [5] (which is > 2 secs for 15 anchors).

---

[3]S. Chakrabarti's personal communication.

For completeness we have adapted TAGME to work on long texts by shifting a text window of about 10 anchors. This way its time complexity grows linearly with the number of detected anchors, which compares favorably against [5]'s system whose time complexity scales "mildly quadratically". Of course, this is an unfavorable setting for TAGME because Chakrabarti's and Milne&Witten's systems deploy the full input text (and thus probably more than 10 anchors). We have preliminary evidence that TAGME is competitive with Chakrabarti's annotator, whereas it improves Milne&Witten's system by achieving an F-Measure of about 78% which surpasses the 74.8% reported in [8] for long and highly linked Wikipedia full-documents. So we can safely conclude that TAGME on long texts is either significantly faster than [5] or more effective than [8].

We are currently investigating the impact of TAGME's annotation onto the performance of our past system SNAKET[3] for the on-the-fly labeled clustering of search-engine results (a lá CLUSTY.COM). In fact SNAKET, as most of its competitors (see e.g. [1]), is based only on syntactic and statistical features and thus, we believe that, it could benefit from TAGME's annotation to improve the effectiveness of the labeling and the clustering phases. Another promising context of application for TAGME could be Web Advertising. The explanatory links and the structured knowledge produced by TAGME could allow the efficient and effective resolution of ambiguity and polysemy issues which often occur when advertiser's keywords are matched against the content of Web pages offering display-ads.

Acknowledgments

4. REFERENCES

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

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

[3] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. *Softw. Pract. & Exper.*, 38(2): 189-225, 2008.

[4] P. Ferragina and U. Scaiella. TAGME: On-the-fly annotation of short text fragents (by Wikipedia entities). Available on http://arxiv.org/abs/1006.3498.

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

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

[7] 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.

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